INTERNATIONAL STANDARD

ISO/IEC 29500-1

Third edition
2012-09-01

Information technology — Document description and processing languages — Office Open XML File Formats —

Part 1: Fundamentals and Markup Language Reference

Technologies de l'information — Description des documents et langages de traitement — Formats de fichier "Office Open XML" —

Partie 1: Principes essentiels et référence de langage de balisage

Reference number
ISO/IEC 29500-1:2012(E)

© ISO/IEC 2012

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2012

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Table of Contents

Foreword viii

Introduction x

1. Scope 1

2. Conformance 2

2.1 Document Conformance 2
2.2 Application Conformance 2
2.3 Application Descriptions 3
2.4 Interoperability Guidelines 5

3. Normative References 6

4. Terms and Definitions 10

5. Notational Conventions 13

6. Acronyms and Abbreviations 14

7. General Description 15

8. Overview 16

8.1 Content Overview 16
8.2 Packages and Parts 16
8.3 Consumers and Producers 16
8.4 WordprocessingML 16
8.5 SpreadsheetML 18
8.6 PresentationML 18
8.7 Supporting MLs 19

9. Packages 21

9.1 Office Open XML's Use of OPC 21
9.2 Relationships in Office Open XML 22

10. Markup Compatibility and Extensibility 27

10.1 Constraints on Office Open XML's Use of Markup Compatibility and Extensibility 27

11. WordprocessingML 28

11.1 Glossary of WordprocessingML-Specific Terms 28
11.2 Package Structure 28
11.3 Part Summary 31
11.4 Document Template 57
11.5 Framesets 58
11.6 Master Documents and Subdocuments 59
11.7 Mail Merge Data Source 60
11.8 Mail Merge Header Data Source 61
11.9 XSL Transformation 62

12. SpreadsheetML 64

12.1 Glossary of SpreadsheetML-Specific Terms 64

©ISO/IEC 2012 – All rights reserved

12.2 Package Structure 65
12.3 Part Summary 67
12.4 External Workbooks 101

13. PresentationML 103
13.1 Glossary of PresentationML-Specific Terms 103
13.2 Package Structure 103
13.3 Part Summary 106
13.4 HTML Publish Location 124
13.5 Slide Synchronization Server Location 125

14. DrawingML 127
14.1 Glossary of DrawingML-Specific Terms 127
14.2 Part Summary 127

15. Shared 140
15.1 Glossary of Shared Terms 140
15.2 Part Summary 141
15.3 Hyperlinks 162

16. Part Overview 164
16.1 WordprocessingML Summary 164
16.2 SpreadsheetML Summary 164
16.3 PresentationML Summary 165
16.4 DrawingML Summary 166
16.5 Shared Summary 166

17. WordprocessingML Reference Material 169
17.1 Table of Contents 169
17.2 Main Document Story 189
17.3 Paragraphs and Rich Formatting 195
17.4 Tables 366
17.5 Custom Markup 478
17.6 Sections 540
17.7 Styles 607
17.8 Fonts 662
17.9 Numbering 684
17.10 Headers and Footers 726
17.11 Footnotes and Endnotes 739
17.12 Glossary Document 772
17.13 Annotations 790
17.14 Mail Merge 921
17.15 Settings 961
17.16 Fields and Hyperlinks 1149
17.17 Miscellaneous Topics 1288
17.18 Simple Types 1299

18. SpreadsheetML Reference Material 1520
18.1 Table of Contents 1520
18.2 Workbook 1539

18.3 Worksheets 1586
18.4 Shared String Table 1711
18.5 Tables 1720
18.6 Calculation Chain 1736
18.7 Comments 1739
18.8 Styles 1746
18.9 Metadata 1794
18.10 Pivot Tables 1808
18.11 Shared Workbook Data 1954
18.12 QueryTable Data 1982
18.13 External Data Connections 1990
18.14 Supplementary Workbook Data 2012
18.15 Volatile Dependencies 2022
18.16 Custom XML Mappings 2027
18.17 Formulas 2035
18.18 Simple Types 2429

19. PresentationML Reference Material 2511

19.1 Table of Contents 2511
19.2 Presentation 2517
19.3 Slides 2551
19.4 Comments 2590
19.5 Animation 2594
19.6 Slide Synchronization Data 2682
19.7 Simple Types 2683

20. DrawingML - Framework Reference Material 2711

20.1 DrawingML - Main 2711
20.2 DrawingML - Picture 3078
20.3 DrawingML - Locked Canvas 3086
20.4 DrawingML - WordprocessingML Drawing 3087
20.5 DrawingML - SpreadsheetML Drawing 3143

21. DrawingML - Components Reference Material 3172

21.1 DrawingML - Main 3172
21.2 DrawingML - Charts 3354
21.3 DrawingML - Chart Drawings 3462
21.4 DrawingML - Diagrams 3483

22. Shared MLs Reference Material 3591

22.1 Math 3591
22.2 Extended Properties 3712
22.3 Custom Properties 3719
22.4 Variant Types 3721
22.5 Custom XML Data Properties 3732
22.6 Bibliography 3735
22.7 Additional Characteristics 3772
22.8 Office Document Relationships 3776
22.9 Shared Simple Types 3777

23. Custom XML Schema References 3791
23.1 Table of Contents 3791
23.2 Elements 3791
Annex A. (normative) Schemas – W3C XML Schema 3795
A.1 WordprocessingML 3795
A.2 SpreadsheetML 3861
A.3 PresentationML 3945
A.4 DrawingML - Framework 3976
A.5 DrawingML - Components 4044
A.6 Shared MLs 4095
A.7 Custom XML Schema References 4119
Annex B. (informative) Schemas – RELAX NG 4121
B.1 WordprocessingML 4122
B.2 SpreadsheetML 4168
B.3 PresentationML 4263
B.4 DrawingML - Framework 4288
B.5 DrawingML - Components 4339
B.6 Shared MLs 4375
B.7 Custom XML Schema References 4392
B.8 Additional Resources 4392
Annex C. (informative) Additional Syntax Constraints 4395
Annex D. (informative) Namespace Prefix Mapping in Examples 4396
Annex E. (informative) WordprocessingML Custom XML Data Extraction 4398
Annex F. (normative) WordprocessingML Page Borders 4401
Annex G. (normative) Predefined SpreadsheetML Style Definitions 4402
G.1 Built-in Table Styles 4402
G.2 Built-in Cell Styles 4458
G.3 Built-in PivotTable AutoFormats 4462
Annex H. (informative) Example Predefined DrawingML Shape and Text Geometries 4478
Annex I. (informative) Bidirectional Support 4479
I.1 Introduction 4479
I.2 Shared (WordprocessingML and DrawingML) 4479
I.3 WordprocessingML 4481
I.4 SpreadsheetML 4484
I.5 PresentationML 4485
I.6 DrawingML 4485
I.7 The Unicode Bidirectional Algorithm and Office Open XML 4485
Annex J. (informative) Accessibility Best Practices 4489
J.1 The Value of Creating an Accessible Office Open XML Implementation 4489
J.2 Needs by Type of Disability 4490
J.3 Best Practices for Developers 4493
J.4 Best Practices for Document and Template Authors 4496

J.5 Best Practices for Customers of Office Open XML Implementations 4509

Annex K. (informative) Root Element Locations 4522

K.1 Grouped by Part Name 4522

K.2 Grouped by Schema Name 4524

Annex L. (informative) Primer 4528

L.1 Introduction to WordprocessingML 4528

L.2 Introduction to SpreadsheetML 4620

L.3 Introduction to PresentationML 4759

L.4 Introduction to DrawingML 4791

L.5 Introduction to VML 4950

L.6 Introduction to Shared MLs 4964

L.7 Miscellaneous Topics 4995

Annex M. (informative) Differences Between ISO/IEC 29500 and ECMA-376:2006 5010

M.1 WordprocessingML 5010

M.2 SpreadsheetML 5013

M.3 PresentationML 5014

M.4 DrawingML 5015

M.5 VML 5016

M.6 Shared 5016

M.7 Custom XML Schema References 5017

Bibliography 5018

©ISO/IEC 2012 – All rights reserved

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 29500-1 was prepared by ISO/IEC JTC 1, Information technology, Subcommittee SC 34, Document description and processing languages.

This third edition cancels and replaces the second edition (ISO/IEC 29500-1:2011), which has been technically revised by incorporation of the Amendment ISO/IEC 29500-1:2011/Amd.1:2012 and the Technical Corrigendum ISO/IEC 29500-1:2011/Cor.1:2012.

ISO/IEC 29500 consists of the following parts, under the general title *Information technology — Document description and processing languages — Office Open XML File Formats*:

- *Part 1: Fundamentals and Markup Language Reference*
- *Part 2: Open Packaging Conventions*
- *Part 3: Markup Compatibility and Extensibility*
- *Part 4: Transitional Migration Features*

Annexes A, G and H form a normative part of this Part of ISO/IEC 29500. Annexes B–F and I–N are for information only.

This Part of ISO/IEC 29500 includes five annexes (Annex A, Annex B, Annex F, Annex G, and Annex H) that refer to data files provided in electronic form.

©ISO/IEC 2012 – All rights reserved

The document representation formats defined by this Part are different from the formats defined in the corresponding Part of ECMA-376:2006. Some of the differences are reflected in schema changes, as shown in Annex M of this Part.

©ISO/IEC 2012 – All rights reserved

Introduction

ISO/IEC 29500 specifies a family of XML schemas, collectively called *Office Open XML*, which define the XML vocabularies for word-processing, spreadsheet, and presentation documents, as well as the packaging of documents that conform to these schemas.

The goal is to enable the implementation of the Office Open XML formats by the widest set of tools and platforms, fostering interoperability across office productivity applications and line-of-business systems, as well as to support and strengthen document archival and preservation, all in a way that is fully compatible with the existing corpus of Microsoft Office documents.

The following organizations have participated in the creation of ISO/IEC 29500 and their contributions are gratefully acknowledged:

Apple, Barclays Capital, BP, The British Library, Essilor, Intel, Microsoft, NextPage, Novell, Statoil, Toshiba, and the United States Library of Congress

©ISO/IEC 2012 – All rights reserved

Information technology — Document description and processing languages — Office Open XML File Formats

Part 1:

Fundamentals and Markup Language Reference

1. Scope

ISO/IEC 29500 defines a set of XML vocabularies for representing word-processing documents, spreadsheets and presentations. On the one hand, the goal of ISO/IEC 29500 is to be capable of faithfully representing the pre-existing corpus of word-processing documents, spreadsheets and presentations that had been produced by the Microsoft Office applications (from Microsoft Office 97 to Microsoft Office 2008, inclusive) at the date of the creation of ISO/IEC 29500. It also specifies requirements for Office Open XML consumers and producers. On the other hand, the goal is to facilitate extensibility and interoperability by enabling implementations by multiple vendors and on multiple platforms.

This Part of ISO/IEC29500 specifies concepts for documents and applications of both strict and transitional conformance.

2. Conformance

2.1 Document Conformance

A document of conformance class Office Open XML Strict shall be a package of conformance class OPC, as specified in ISO/IEC 29500-2, for which all the following shall hold:

- The document obeys all constraints specified in this Part of ISO/IEC 29500
- The document is of category Wordprocessing, Spreadsheet, or Presentation, as defined in §4
- For each OPC Part of the document of the types listed in §11.3, §12.3, §13.3, §14.2 or §15.2, all the following shall hold:
    i. The part is of conformance class MCE, as specified in ISO/IEC 29500-3
    ii. After the removal of any extensions using the mechanisms in ISO/IEC 29500-3, the part is valid against the strict W3C XML Schema (Appendix A)

This Part of ISO/IEC 29500 uses the following further terms to refer to documents of conformance class Office Open XML Strict:

- *WML Strict*, if the document is of category Wordprocessing
- *SML Strict*, if the document is of category Spreadsheet
- *PML Strict*, if the document is of category Presentation

2.2 Application Conformance

Application conformance incorporates both syntax and semantics.

- A conforming consumer shall not reject any conforming documents of at least one document conformance class.
- A conforming producer shall be able to produce conforming documents of at least one document conformance class.
- A conforming application shall treat the information in Office Open XML documents in a manner consistent with the semantic definitions given in ISO/IEC 29500. An application's intended behavior need not require that application to process all of the information in an Office Open XML document. However, the information that it does process shall be processed in a manner that is consistent with the semantic definitions given in ISO/IEC 29500.

[*Note*: This note illustrates the third bullet above. Conforming applications might serve various functions. Examples include a viewer, an editor, and a back-end processor. Here is an illustration of how the third bullet applies to each of those examples:

- If a conforming viewer supports a given feature, then when it displays information using that feature, it respects the semantics of that feature as described in the Standard.

- If a conforming editor supports a given feature, then when it provides its user with an interface for manipulating information using that feature, it respects the semantics of that feature as described in the Standard.
- If a conforming back-end processor supports a given feature, then when that processor transforms or assembles information involving that feature, that processor respects the semantics of that feature as described in the Standard.

end note]

This Part of ISO/IEC 29500 defines the following application conformance classes:

- *WML Strict*, if the application is a conforming application that is a consumer or producer of documents having conformance class WML Strict.
- *SML Strict*, if the application is a conforming application that is a consumer or producer of documents having conformance class SML Strict.
- *PML Strict*, if the application is a conforming application that is a consumer or producer of documents having conformance class PML Strict.

Conformance can also involve the use of application descriptions; see §2.3 for details.

2.3 Application Descriptions

An application can be defined as conforming to zero or more *application descriptions* in a particular conformance class.

The application descriptions defined within ISO/IEC 29500 are:

- Base
- Full

[*Note*: These application descriptions should not be taken as limiting the ability of an application provider to create innovative applications. They are intended as a mechanism for labelling applications rather than for restricting their capabilities. The intention is to promote interoperability between different applications that share the same conformance class. Application descriptions are orthogonal to the conformance of the documents produced by those applications. For example, a tool used for automated translation of documents might have an application description of “Base” but will still produce fully conformant documents. *end note*]

The application descriptions are determined in terms of an application’s semantic understanding of particular features. *Semantic understanding* is to be interpreted in that an application shall treat the information in Office Open XML documents in a manner consistent with the semantic definitions given in ISO/IEC 29500.

Each application description is identified by a URI.

The application descriptions are defined in the following subclauses.

2.3.1 Base Application Description

Description URI: http://purl.oclc.org/ooxml/descriptions/base

An application conforming to this description has a semantic understanding of at least one feature within its conformance class.

[*Note*: In addition, applications that include a user interface are strongly recommended to support all accessibility features appropriate to that user interface. *end note*]

2.3.2 Full Application Description

Description URI: http://purl.oclc.org/ooxml/descriptions/full

An application conforming to this description has a semantic understanding of every feature within its conformance class.

2.3.3 Additional Application Descriptions

It is expected that additional application descriptions will be defined within the maintenance process for ISO/IEC 29500. It is also expected that third parties might define their own application descriptions; for example to inform their procurement decisions, or to deal with domains such as accessibility.

[*Note*: A possible application description would be a “standard” application description for a wordprocessing application. This could be created by taking the intersection of the features available in common wordprocessing applications such as Word 2000, OpenOffice 2, WordPerfect, and iWork Pages. In addition, it could define formats such as specific image and video formats required to be supported to conform to the description. Similar descriptions could be created for spreadsheet applications and presentation applications. Such a description would promote interoperability between applications implementing OOXML. It would also promote interoperability between applications implementing OOXML and applications implementing other document formats such as ISO/IEC 26300. *end note*]

Application descriptions are not required to be strict subsets of each other. An application can simultaneously conform to multiple application descriptions.

Any such newly created description shall enumerate the features that are required for conformance to it. Such a description should provide a machine-processable schema, preferably using a standard such as ISO/IEC 19757.

[*Note*: If the application conforming to a description is a document consumer, it should be able to consume any document that respects such a schema associated with the description. If the application is a document producer, any document produced by that application should respect the schema of the description. *end note*]

Any such description should be identified using a URI, in a similar manner to the names used for application descriptions within ISO/IEC 29500.

[*Note*: For the convenience of users of the description, it is recommended that creators of a description should make a human- or machine-readable form of that description available at a URL corresponding to the description URI. *end note*]

2.3.4 Representation of Application Descriptions within Documents

An application description is related to applications, rather than to document conformance. Therefore, there is no normative mechanism for representing an application description within a document.

[*Note*: It is recommended that implementers wishing to represent an application description within a document use the standard metadata mechanism for Office Open XML. *end note*]

2.4 Interoperability Guidelines

[*Guidance*: The following interoperability guidelines incorporate semantics6ϕ

For the guidelines to be meaningful, a software application should be accompanied by documentation that describes what subset of ISO/IEC 29500 it supports. The documentation should highlight any behaviors that would, without that documentation, appear to violate the semantics of document XML elements. Together, the application and documentation should satisfy the following conditions.

1. The application need not implement operations on all XML elements defined in ISO/IEC 29500. However, if it does implement an operation on a given XML element, then that operation should use semantics for that XML element that are consistent with ISO/IEC 29500.
2. If the application moves, adds, modifies, or removes XML element instances with the effect of altering document semantics, it should declare the behavior in its documentation.

The following scenarios illustrate these guidelines.

- A presentation editor that interprets the preset shape geometry “`rect`” as an ellipse does not observe the first guideline because it implements “`rect`” but with incorrect semantics.
- A batch spreadsheet processor that saves only computed values even if the originally consumed cells contain formulas, might satisfy the first condition, but does not observe the second because the editability of the formulas is part of the cells’ semantics. To observe the second guideline, its documentation should describe the behavior.
- A batch tool that reads a word-processing document and reverses the order of text characters in every paragraph with “Title” style before saving it can be conforming even though ISO/IEC 29500 does not recommend this behavior. This tool’s behavior would be to transform the title “Office Open XML” into “LMX nepO eciffO”. Its documentation should declare its effect on such paragraphs.

The normative requirements in §2.1 imply that a conforming producer shall not write unescaped non-XML characters. As an implementation guideline, a conforming producer additionally should not write escaped non-XML characters. Doing so damages interoperability with existing XML-based standards such as SOAP and RDF.

For example, implementers could either refuse to create documents including such characters, or warn users that including such characters compromises the re-usability of their documents. *end guidance*]

©ISO/IEC 2012 – All rights reserved

3. Normative References

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ANSI X3.4-1986, *American Standard Code for Information Interchange (ASCII)*

Bureau of Standards, Metrology and Inspection of the Ministry of Economic Affairs, *CNS 7648: Data Elements and Interchange Formats — Information Interchange — Representation of Dates and Times*

Calendar Reform Committee, Indian Ephemeris and Nautical Almanac. 1957

Stokes, M., M. Anderson, S. Chandrasekar, and R. Motta. *A Standard Default color Space for the Internet. Vers. 1.10*. November 5, 1996. http://www.w3.org/Graphics/Color/sRGB

Har'El, Zvi, *Gauss Formula for the Julian Date of Passover*. Deptartment of Mathematics, Technion, Israel Institue of Technology, Haifa 32000, Isreal, 2005, 6

Duerst, M, and M Suignard. *Internationalized Resource Identifiers (IRIs).* IETF. January 2005. http://tools.ietf.org/html/rfc3987

IANA, *Character Sets from IANA*, as specified at http://www.iana.org/assignments/character-sets

IANA. *MIME Media Types*. Internet Assigned Numbers Authority. http://www.iana.org/assignments/media-types/

IEC 60559:1989, *Binary Floating-Point Arithmetic for Microprocessor Systems*

ISO/IEC 2382-1:1993, *Information technology — Vocabulary — Part 1: Fundamental terms*

ISO 8601:2004, *Data elements and interchange formats — Information interchange — Representation of dates and times*

ISO/IEC 8859-1:1998, *Information technology — 8-bit single-byte coded graphic character sets — Part 1: Latin alphabet No. 1* (referred to in ISO/IEC 29500 as the ANSI character set)

ISO/IEC 9075-1, *Information technology — Database languages — SQL — Part 1: Framework (SQL/Framework)*

ISO/IEC 10118-3:2004, *Information technology — Security techniques — Hash-functions — Part 3: Dedicated hash-functions*.

ISO/IEC 10646, *Information technology — Universal Coded Character Set (UCS)*.

©ISO/IEC 2012 – All rights reserved

ISO/IEC 14496-22:2009, *Information technology — Coding of audio-visual objects — Part 22: Open Font Format*

ISO/IEC 29500-2:2011, Information technology — Document description and processing languages — Office Open XML File Formats – Part 2: Open Packaging Conventions

ISO/IEC 29500-3:2011, Information technology — Document description and processing languages — Office Open XML File Formats – Part 3: Markup Compatibility and Extensibility

Japanese Industrial Standard, JIS X 0301: *Data elements and interchange formats —Information interchange — Representation of dates and times*. Japan, 2002.

Kingdom of Saudi Arabia, Ministry of Islamic Affairs, Endowments, Daʻwah and Guidance.

Korean Law Enactment No. 4, 1961.

Faure, D. (n.d.). *Creating and Using Components (KParts)*. http://techbase.kde.org/Projects/Documentation.

Maimon, Rabbi Moshe ben, *Complete Restatement of the Oral Law (Mishneh Torah)*.

Ausbrooks, Ron, et al. *Mathematical Markup Language (MathML) Version 2.0 (Second Edition)*. October 21, 2003. http://www.w3.org/TR/MathML/.

Kaliski, B. *The MD2 Message-Digest Algorithm*. April 1992. http://www.ietf.org/rfc/rfc1319.txt

Rivest, R. *The MD4 Message-Digest Algorithm*. April 1992. http://www.ietf.org/rfc/rfc1320.txt

The MD5 Message-Digest Algorithm. April 1992. http://www.ietf.org/rfc/rfc1321.txt.

National Measurement Regulations 1999, Commonwealth of Australia
http://www.comlaw.gov.au/Details/F2011C00445

NIST Guide to SI Units, http://physics.nist.gov/Pubs/SP811/appenB9.html

QuickTime File Format Specification (2007-09-04 version)
http://developer.apple.com/standards/classicquicktime.html

Resource Description Framework (RDF), http://www.w3.org/RDF/

RFC 822, *Standard for ARPA Internet Text Messages* (http://www.ietf.org/rfc/rfc0822.txt)

RFC 2045, Borenstein, N., and N. Freed. *Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies*. The Internet Society. 1996. http://www.ietf.org/rfc/rfc2045.txt

RFC 2119, Bradner, Scott, 1997: *Key words for use in RFCs to Indicate Requirement Levels*. http://www.ietf.org/rfc/rfc2119.txt

RFC 2616, Berners-Lee, T., R. Fielding, H. Frystyk, J. Gettys, P. Leach, L. Masinter, and J. Mogul. *Hypertext Transfer Protocol—HTTP/1.1*. The Internet Society. 1999. http://www.ietf.org/rfc/rfc2616.txt

RFC 3066, Alvestrand, H. *Tags for the Identification of Languages*. The Internet Society. 2001. http://www.ietf.org/rfc/rfc3066.txt

RFC 3339, Klyne, G. and C. Newman. *Date and Time on the Internet: Timestamps*. The Internet Society. 2002. http://www.ietf.org/rfc/rfc3339.txt

RFC 3629, Yergeau, F. *UTF-8, a transformation format of ISO 10646*. The Internet Society. 2003. http://www.ietf.org/rfc/rfc3629.txt

RFC 3986, Berners-Lee, T., R. Fielding, and L. Masinter. *Uniform Resource Identifier (URI): Generic Syntax*. The Internet Society. 2005. http://www.ietf.org/rfc/rfc3986.txt

Simple Object Access Protocol (SOAP), http://www.w3.org/TR/soap12

SMIL, Bulterman, D., Grassel, G., Jansen, J., Koivisto, A., Layaïda, N., Michel, T., et al. (2005, December 13). *Synchronized Multimedia Integration Language (SMIL 2.1)*. Retrieved from W3C: http://www.w3.org/TR/SMIL/

SVG, Andersson, O., Armstrong, P., Axelsson, H., Berjon, R., Bézaire, B., Bowler, J., et al. (2003, January 14). *Scalable Vector Graphics (SVG) 1.1 Specification*. Retrieved from W3C - World Wide Web Consortium: http://www.w3.org/TR/SVG/

The GNOME Project. (2003, December 12). *Component Model - Bonobo Document Model*. Retrieved from The GNOME Development Site: http://developer.gnome.org/bonobo-activation/stable/

The Unicode Consortium. *The Unicode Standard*, http://www.unicode.org/standard/standard.html.

Unicode Technical Note #28, *Nearly Plain-Text Encoding of Mathematics*. August 29, 2006, http://www.unicode.org/notes/tn28

United States Postal Service. *Domestic Mail Manual*. United States Postal Service. November 8, 2007. http://pe.usps.com/cpim/ftp/manuals/dmm300/Full/MailingStandards.pdf

The Units of Measurement Regulations 1995, United Kingdom http://www.opsi.gov.uk/si/si1995/Uksi_19951804_en_2.htm

Universal Postal Union. *POST*CODE: Postal addressing systems*. Berne: UPU Publications, 2006, ISBN 92-95025-37-7, ISSN 1020-6019

Web Accessibility Initiative (WAI), http://www.w3.org/WAI/

XSLT, Clark, James, *XSL Transformations (XSLT) Version 1.0*, World Wide Web Consortium Recommendation. 1999. http://www.w3.org/TR/xslt

XML, Tim Bray, Jean Paoli, Eve Maler, C. M. Sperberg-McQueen, and François Yergeau (editors). *Extensible Markup Language (XML) 1.0, Fourth Edition*. World Wide Web Consortium. 2006. http://www.w3.org/TR/2006/REC-xml-20060816/ [Implementers should be aware that a further correction of the normative reference to XML to refer to the 5th Edition will be necessary when the related Reference

Specifications to which this International Standard also makes normative reference and which also depend upon XML, such as XSLT, XML Namespaces and XML Base, are all aligned with the 5th Edition.]

XML Base, Marsh, Jonathan. *XML Base*. World Wide Web Consortium. 2001. http://www.w3.org/TR/2001/REC-xmlbase-20010627/

XML Namespaces, Tim Bray, Dave Hollander, Andrew Layman, and Richard Tobin (editors). *Namespaces in XML 1.0 (Third Edition)*, 8 December 2009. World Wide Web Consortium. http://www.w3.org/TR/2009/REC-xml-names-20091208/

XPATH, Clark, James; DeRose, Steve *XML Path Language (XPath) Version 1.0*, World Wide Web Consortium Recommendation. 1999. http://www.w3.org/TR/xpath.

XML Schema Part 0: Primer (Second Edition), W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-0/

XML Schema Part 1: Structures (Second Edition), W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-1/

XML Schema Part 2: Datatypes (Second Edition), W3C Recommendation 28 October 2004, http://www.w3.org/TR/xmlschema-2/

.ZIP File Format Specification from PKWARE, Inc., version 6.2.0 (2004), as specified in http://www.pkware.com/documents/APPNOTE/APPNOTE_6.2.0.txt

©ISO/IEC 2012 – All rights reserved

4. Terms and Definitions

For the purposes of this document, the following terms and definitions apply. Other terms are defined where they appear in *italic* typeface, on the left side of a syntax rule, or within subclauses of language-specific grammars (§17.16 and §18.17). Terms explicitly defined in this Part of ISO/IEC 29500 are not to be presumed to refer implicitly to similar terms defined elsewhere. [*Note*: This Part uses OPC-related terms, which are defined in ISO/IEC 29500-2. *end note*]

application — A consumer or producer.

behavior — External appearance or action.

behavior, implementation-defined — Unspecified behavior where each implementation is expected to document that behavior, which would thereby promote predictability and reproducibility within any given implementation. (This term is sometimes called “application-defined behavior”.)

behavior, locale-specific — Behavior that depends on local conventions of nationality, culture, and language.

behavior, unspecified —Behavior where ISO/IEC 29500 makes no recommendations. (This term is sometimes called “application-dependent behavior”.) [*Note*: To add an extension, an implementer must use the extensibility mechanisms described by ISO/IEC 29500 rather than trying to do so by giving meaning to otherwise unspecified behavior. *end note*]

byte — A sequence of 8 bits treated as a unit.

comment — A note that an author or reviewer attaches to content in a document. Although a consumer might chose to display comments, they are not considered part of the body of the document. A comment might include the text of the note, the comment author's name and initials, and date of creation, among other things.

consumer — A piece of software or a device that reads packages through a package implementer. A consumer is often designed to consume packages only for a specific physical package format.

content type — Describes the content stored in a part. Content types define a media type, a subtype, and an optional set of parameters, as defined in RFC 2616.

document category — One of the three categories of Office Open XML documents: Wordprocessing, Spreadsheet, and Presentation, defined as follows:

- A document whose package-relationship item contains a relationship to a Main Document part (§11.3.10) is a document of category Wordprocessing.

- A document whose package-relationship item contains a relationship to a Workbook part (§12.3.23) is a document of category Spreadsheet.
- A document whose package-relationship item contains a relationship to a Presentation part (§13.3.6) is a document of category Presentation.

An Office Open XML document can contain one or more embedded Office Open XML packages (§15.2.11) with each embedded package having any of the three document categories. However, the presence of these embedded packages does not change the category of the document.

DrawingML — A set of conventions for specifying the location and appearance of drawing elements in an Office Open XML document.

extension — Any XML element, XML attribute, relationship, or part not explicitly included in ISO/IEC 29500, but that uses the extensibility mechanisms described by ISO/IEC 29500.

id — In some XML-related technologies, the term *id* implies use of the xsd:ID data type. In this international standard, this term is used to refer to a variety of different identification schemes. See *unique identifier*.

MDX — A multi-dimensional expression language, passed to an OLAP provider. The method of interpreting of this is defined by the server-side OLAP provider implementation.

Office Open XML document — A rendition of a data stream formatted using the wordprocessing, spreadsheet, or presentation ML and its related MLs as described in ISO/IEC 29500-1 and ISO/IEC 29500-4. Such a document is represented as a package as described in ISO/IEC 29500-2.

OLAP — A type of online analytical processing database which uses a multi-dimensional data model.

OLE – OLE in this context does not refer to any specific technology; instead, it refers to the generalized abstraction of embedding and linking objects within a document.

package— A ZIP archive that conforms to the Open Packaging Conventions specification defined in ISO/IEC 29500-2.

package, embedded— A package that has been stored as the target of an Embedded Package relationship (§15.2.11) in an Office Open XML document

PresentationML — A set of conventions for representing an Office Open XML document of category Presentation.

producer — A piece of software or a device that writes packages through a package implementer. A producer is often designed to produce packages according to a particular physical package format specification.

relationship —The kind of connection between a source part and a target part in a package. Relationships make the connections between parts directly discoverable without looking at the content in the parts, and without altering the parts themselves. (See also Package Relationships.)

©ISO/IEC 2012 – All rights reserved

relationships part — A part containing an XML representation of relationships.

relationship, explicit — A relationship in which a resource is referenced from a source part's XML using the Id attribute of a Relationship tag.

relationship, implicit — A relationship that is not explicit.

SpreadsheetML — A set of conventions for representing an Office Open XML document of category Spreadsheet.

unique identifier — In some XML-related technologies, the term *unique identifier* implies use of the xsd:ID data type. In this international standard, this term is used to refer to a variety of different identification schemes. See *id*.

WordprocessingML — A set of conventions for representing an Office Open XML document of category Wordprocessing.

©ISO/IEC 2012 – All rights reserved

5. Notational Conventions

The following typographical conventions are used in ISO/IEC 29500:

- The first occurrence of a new term is written in italics. [*Example*: The text in ISO/IEC 29500 is divided into *normative* and *informative* categories. *end example*]
- In each definition of a term in §4 (Terms and Definitions), the term is written in bold. [*Example*: **behavior** — External appearance or action. *end example*]
- The tag name of an XML element is written using a distinct style and typeface. [*Example*: The bookmarkStart and bookmarkEnd elements specify … *end example*]
- The name of an XML attribute is written using a distinct style and typeface. [*Example*: The dropCap attribute specifies … *end example*]
- The value of an XML attribute is written using a constant-width style. [*Example*: The attribute value of `auto` specifies … *end example*]
- The qualified or unqualified name of a simple type, complex type, or base datatype is written using a distinct style and typeface. [*Example*: The possible values for this attribute are defined by the ST_HexColor simple type. *end example*]

When assigned namespaces are used in examples, they are included at the beginning of the example, but with the specific namespace replaced with ellipsis ("…") for brevity.

©ISO/IEC 2012 – All rights reserved

6. Acronyms and Abbreviations

This clause is informative

The following acronyms and abbreviations are used throughout ISO/IEC 29500:

IEC — the International Electrotechnical Commission

ISO — the International Organization for Standardization

W3C — World Wide Web Consortium

End of informative text

7. General Description

This Part of ISO/IEC 29500 is divided into the following subdivisions:

1. Front matter (clauses 1–7);
2. Overview (clause 8);
3. Package Part Structure (clauses 9–16);
4. Reference Material (clauses 17–23);
5. Annexes

Examples are provided to illustrate possible forms of the constructions described. References are used to refer to related clauses. Notes are provided to give advice or guidance to implementers or programmers. Rationale provides explanatory material as to why something is or is not in ISO/IEC 29500. Annexes provide additional information or summarize the information contained in ISO/IEC 29500.

Clauses 1–5, 7, 9–15, 17–23, Annex A, Annex F, and Annex G form the normative part of this Part of ISO/IEC 29500; the Introduction, clauses 6, 8, and 16, Annex B–Annex E, Annex H–Annex M, as well as notes, examples, rationale, guidance, and the index, are informative.

Except for whole clauses or annexes that are identified as being informative, informative text that is contained within normative text is indicated in the following ways:

1. [*Example:* code fragment, possibly with some narrative … *end example*]
2. [*Note:* narrative … *end note*]
3. [*Rationale:* narrative … *end rationale*]
4. [*Guidance*: narrative … *end guidance*]

In addition to the declarations in the “General Description”, each annex that is informative, also contains the following text at the beginning of the annex: "**This annex is informative.**"

©ISO/IEC 2012 – All rights reserved

8. Overview

This clause is informative.

This clause contains an overview of Office Open XML.

8.1 Content Overview

This standard contains predominantly the following three types of information:

1. Normative W3C XML Schemas, informative RELAX NG schemas and an associated validation procedure for validating document syntax against those schemas (Annex A and Annex B)
2. Descriptions of XML element semantics. The semantics of an XML element refers to its intended interpretation by a human being (chiefly in §11, §12, §13, and §14)
3. Additional syntax constraints in written form

8.2 Packages and Parts

An Office Open XML document is represented as a series of related *parts* that are stored in a container called a *package*. Information about the *relationships* between a package and its parts is stored in the package's *package-relationship ZIP item*. Information about the *relationships* between two parts is stored in the *part-relationship ZIP item* for the source part. A package is an ordinary ZIP archive, which contains that package's content-type item, relationship items, and parts. (Packages are discussed further in ISO/IEC 29500-2.)

A WordprocessingML document contains a part for the body of the text; it might also contain a part for an image referenced by that text, and parts defining document characteristics, styles, and fonts. A SpreadsheetML document contains a separate part for each worksheet; it might also contain parts for images. A PresentationML document contains a separate part for each slide.

8.3 Consumers and Producers

A tool that can read and understand a package is called a *consumer*, while one that can create a package is called a *producer*. An application can be a consumer, a producer, or both. For example, when a word processor creates a new document, it acts as a producer. When it is used to open an existing document for reading or search purposes, it acts as a consumer. When it is used to open an existing document, edit it, and save the result, it acts as both consumer and producer. Similar scenarios exist for spreadsheet and presentation applications.

8.4 WordprocessingML

This subclause introduces the overall form of a WordprocessingML package, and identifies some of its main components. (See Annex L for a more detailed introduction.)

A WordprocessingML package has a relationship of type officeDocument, which specifies the location of the main part in the package. For a WordprocessingML document, that part contains the main text of the document.

A WordprocessingML package's main part starts with a word processing root element. That element contains a *body*, which, in turn, contains one or more *paragraphs* (as well as tables, pictures, and the like). A paragraph contains one or more runs, where a *run* is a container for one or more pieces of *text* having the same set of properties. Like many elements that defined a logical piece of a word processing document, each run and paragraph can have associated with it a set of *properties*. For example, a run might have the property bold, which indicates that run's text is to be displayed in a bold typeface.

A WordprocessingML document is organized into *sections*, and the layout of a page on which the text appears within a section is controlled by that section's properties. For example, each section can have its own *headers* and *footers*.

One relationship from the document part specifies the document's styles. A *style* defines a text display format. A style can have properties, which can be applied to individual paragraphs or runs. Styles make runs more compact by reducing the number of repeated definitions and properties, and the amount of work required to make changes to the document's appearance. With styles, the appearance of all the pieces of text that share a common style can be changed in one place, in that style's definition.

A series of paragraphs can have *numbering* applied to them via a numbering definition instance or a numbering style.

Data in a WordprocessingML document can be organized in a *table*, a two-dimensional grid of *cells* organized into *rows* and *columns*. Cells and whole tables can have associated properties. A cell can contain text and paragraphs, for example.

Text within a WordprocessingMLdocument can be determined dynamically via the use of *fields*. Fields consist of *field instructions* (the text that dictates the field's dynamic behavior) and the *field result* (the text resulting from the dynamic calculation of the field instructions. For example, page numbers are represented as fields. A *hyperlink* consists of two pieces of information: the hyperlink itself—the text the user clicks—and the target for the link. Potential targets include external files, e-mail addresses, web sites, and bookmarks within the document itself.

A WordprocessingML document can also contain *custom markup*, user-defined semantics applied to arbitrary document content.

A WordprocessingML document is not stored as one large body in a single part; instead, the elements that implement certain groupings of functionality are stored in separate parts. For example, all footnotes in a document are stored in one footnote part, while each section can have up to three different header parts and three different footer parts, to support headers and footers on odd-numbered pages, even-numbered pages, and the first page.

8.5 SpreadsheetML

This subclause introduces the overall form of a SpreadsheetML package, and identifies some of its main components. (See Annex L for a more detailed introduction.)

A SpreadsheetML package has a relationship of type officeDocument, which specifies the location of the main part in the package. For a SpreadsheetML document, that part contains the workbook definition.

A SpreadsheetML package's main part starts with a spreadsheet root element. That element is a *workbook*, which refers to one or more *worksheets*, which, in turn, contain the data. A worksheet is a two-dimensional grid of *cells* that are organized into *rows* and *columns*.

The cell is the primary place in which data is stored and operated on. A cell can have a number of characteristics, such as numeric, text, date, or time *formatting*; *alignment*; *font*; *color*; and a *border*. Each cell is identified by a *cell reference*, a combination of its column and row headings.

Each horizontal set of cells in a worksheet is called a *row*, and each row has a heading numbered sequentially, starting at 1. Each vertical set of cells in a worksheet is called a *column*, and each column has an alphabetic heading named sequentially from A–Z, then AA–AZ, BA–BZ, and so on.

Instead of data, a cell can contain a *formula*, which is a recipe for calculating a value. Some formulas—called *functions*—are predefined, while others are user-defined. Examples of predefined formulas are `AVERAGE`, `MAX`, `MIN`, and `SUM`. A function takes one or more arguments on which it operates, producing a result. For example, in the formula `SUM(B1:B4)`, there is one argument, `B1:B4`, which is the range of cells B1–B4, inclusive.

Other features that a SpreadsheetML document can contain include the following: *comments*, *hyperlinks*, *images*, and sorted and filtered *tables*.

A SpreadsheetML document is not stored as one large body in a single part; instead, the elements that implement certain groupings of functionality are stored in separate parts. For example, all the data for a worksheet is stored in that worksheet's part, all string literals from all worksheets are stored in a single shared string part, and each worksheet having comments has its own comments part.

8.6 PresentationML

This subclause introduces the overall form of a PresentationML package, and identifies some of its main components. (See Annex L for a more detailed introduction.)

A PresentationML package has a relationship of type officeDocument, which specifies the location of the main part in the package. For a PresentationML document, that part contains the presentation definition.

A PresentationML package's main part starts with a presentation root element. That element contains a *presentation*, which, in turn, refers to a *slide* list, a *slide master* list, a *notes master* list, and a *handout master* list. The slide list refers to all of the slides in the presentation; the slide master list refers to all of the slide masters used in the presentation; the notes master contains information about the formatting of notes pages; and the handout master describes how a handout looks.

A *handout* is a printed set of slides that can be handed out to an *audience* for future reference.

As well as text and graphics, each slide can contain *comments* and *notes*, can have a *layout*, and can be part of one or more *custom presentations*. (A comment is an annotation intended for the person maintaining the presentation slide deck. A note is a reminder or piece of text intended for the presenter or the audience.)

Other features that a PresentationML document can contain include the following: *animation*, *audio*, *video*, and *transitions* between slides.

A PresentationML document is not stored as one large body in a single part; instead, the elements that implement certain groupings of functionality are stored in separate parts. For example, all comments in a document are stored in one comment part while each slide has its own part.

8.7 Supporting MLs

This subclause introduces the set of markup languages used across package categories. (See Annex L for a more detailed introduction.)

The three markup languages described above define the structure of a package that is either a document (WordprocessingML), a spreadsheet (SpreadsheetML), or a presentation (PresentationML). However, there is also a set of shared markup languages used for common elements such as charts, diagrams, and drawing objects. These MLs are discussed below.

8.7.1 DrawingML

DrawingML specifies the location and appearance of drawing elements in a package. For example, these elements could be, but are not limited to, shapes, pictures, and tables. The root element of a DrawingML XML fragment specifies the presence of a drawing at this location in the document.

A *shape* is a geometric object such as a circle, square, or rectangle; a *picture* is an image presented inside the document; and a *table* is a two-dimensional grid of *cells* organized into *rows* and *columns*. Cells and whole tables can have associated properties. A cell can contain text, for example.

DrawingML also specifies the location and appearance of charts in a package. The root element of a chart part is chart, and specifies the appearance of the chart at this location in the document.

In addition, DrawingML specifies package-wide appearance characteristics, such as the package's theme. The *theme* of a document specifies the *color scheme*, *fonts*, and *effects*, which can be referenced by parts of the document—such as text, drawings, charts, and diagrams—in order to create a consistent visual presentation.

A *chart* is a presentation of data in a graphical fashion, such as a pie chart, bar chart, line chart, in order to make trends and exceptions in the data more visually apparent.

DrawingML also specifies the location and appearance of diagrams in a document. Together, the following four parts define a diagram:

- The *data* part (§14.2.4) specifies individual items of information presented in the diagram. Typically, each piece is a simple line of text, but depending on the diagram, an item of data might also be an image.
- The *layout* part (§14.2.5) specifies how the data and shapes are laid out to create the resulting diagram.
- The *colors* part (§14.2.3) specifies the color which is applied to each individual shape in the diagram.
- The *styles* part (§14.2.6) defines how each individual shape in the diagram maps to the document's theme.

8.7.2 Custom XML Data Properties

Custom XML Data properties allow the ability to store arbitrary XML in a package, along with schema information used by that XML.

8.7.3 File Properties

The *core file properties* of a package enable users to discover, get, and set common sets of properties from within that package, regardless of whether it’s a WordprocessingML, SpreadsheetML, or PresentationML package, or another use of OPC. Such properties include creator name, creation date, title, and description.

Extended file properties are specific to Office Open XML packages. For example, for a WordprocessingML package, these properties include the number of characters, words, lines, paragraphs, and pages in the document. For a SpreadsheetML package, these properties include worksheet titles. For a PresentationML package, these properties include presentation format, the number of slides, the number of notes, and whether or not any slides are hidden.

Custom file properties are defined by the user. Examples include the name of the client for whom the document was prepared, a date/time on which some event happened, a document number, or some Boolean status flag. Each custom file property has a value, and that value has a data type.

8.7.4 Math

Math is used, mainly in documents, to specify the structure and appearance of equations. The outermost root element can be either oMath or oMathPara, the latter being a math paragraph with one or more equations where each equation is specified using a single oMath element

8.7.5 Bibliography

Bibliography specifies the structure for all references stored within a document, for use in citations or a bibliography.

End of informative text.

9. Packages

An Office Open XML document is stored as a package, whose format is defined by ISO/IE 29500-2. This subclause contains information regarding Office Open XML's use of OPC.

Throughout ISO/IEC 29500, the Open Packaging Conventions are referred to by their abbreviated name, OPC.

9.1 Office Open XML's Use of OPC

While the OPC specification is designed for the representation of Office Open XML documents, it could also support a much broader range of applications. Clarifications to the use of OPC are discussed in the following subordinate subclauses. Any requirement not mentioned here is inherited from the OPC specification.

9.1.1 Part Addressing

Parts in an Office Open XML package targeted by relationships are addressed in relationship markup through part names. External document resources targeted by a relationship can be addressed using both relative and absolute references.

9.1.2 Fragments

Fragment identifiers are supported as part of all Office Open XML external relationship targets and some Office Open XML internal relationship targets.

9.1.3 Physical Packages

Each Office Open XML document is implemented as a ZIP archive.

9.1.4 Unknown Parts

With the exception of relationship parts, all other parts in an Office Open XML document that are not the target of an implicit or explicit relationship are considered *unknown parts*. Unknown parts shall be ignored on document consumption and can, but need not, be discarded on production.

9.1.5 Trash Items

Trash items represent parts that have been discarded or are no longer in use. Trash items shall not conform to OPC part naming guidelines as defined in ISO/IEC 29500-2 and shall not be associated with a content type. All trash items shall follow the naming scheme: `[trash]/HHHH.dat` where H represents a hexadecimal digit.

[*Example*: A package has two parts that must be updated in-place but both parts have grown beyond their growth hints. The newer updated parts are added as new ZIP items while the original parts are renamed to:

```
[trash]/0000.dat
[trash]/0001.dat
```

©ISO/IEC 2012 – All rights reserved

end example]

9.1.6 Invalid Parts

ZIP archive items that do not conform to OPC part naming guidelines or are not associated with a content type shall not be allowed in an Office Open XML document, with the exception of items specifically defined by ISO/IEC 29500-2 and trash items.

9.1.7 Unknown Relationships

All relationships not defined within ISO/IEC 29500 are considered *unknown relationships*. Unknown relationships are permitted within an Office Open XML document provided that they conform to relationship markup guidelines as defined by the OPC specification. Specifically:

- Conforming consumers shall not fail to load a document containing unknown relationships.
- Conforming producers that are also consumers can, but are not required to, roundtrip and preserve unknown relationships and their target parts.

It is a requirement of ISO/IEC 29500 that dynamic extension mechanisms, such as scripting languages and macro mechanisms, shall use, for the executable parts, the correct content types, and shall not use any of the content types already defined in ISO/IEC 29500.

[*Guidance:* This subclause defines the general behavior for the consumption of unknown relationships. A conforming producer that wishes to store custom business data in an Office Open XML document should use instead the known relationship type for Custom XML Data Storage parts, as defined in §15.2.4. *end guidance*]

9.1.8 Interleaving

Interleaving as defined in ISO/IEC 29500-2 shall not be used for Office Open XML documents.

9.2 Relationships in Office Open XML

In OPC, relationships describe references from parts to other internal resources in the package or to external resources. They represent the type of connection between a source part and a target resource, and make the connection directly discoverable without looking at the part contents, so they are quick to resolve.

The same ZIP item can be the target of multiple relationships. [*Note*: Having multiple paths to a target can make access to that target more convenient. *end note*]

Office Open XML imposes constraints on relationships, described in subsequent clauses of this Part of ISO/IEC 29500. Relationships in Office Open XML are either explicit or implicit.

For an explicit relationship, a resource is referenced from a source part's XML using the Id attribute of a Relationship tag. [*Example*: A document part can have a relationship to a hyperlink only if that hyperlink's Relationship element's Id attribute value is referenced explicitly by the document part's XML. *end example*] [*Note*: Because this mechanism is used generically across multiple XML elements, explicit relationships can be extracted from an Office Open XML document without prior knowledge of tag semantics. *end note*]. Certain relationships shall be *explicit*. All other relationships are *implicit.* [*Note*: The syntax for specifying an implicit relationship varies based on the content model of the XML element which is the source of the relationship. *end note*]. Relationships that are required or permitted, and restrictions on those relationships are described in §10–15 of this Part of ISO/IEC 29500.

[*Example*: Consider a WordprocessingML document that contains the following footnote sentence fragment, "… produced by Ecma[1] (http://www.ecma-international.org/).", which contains a footnote and a hyperlink to a web site. The relationship from a source to a footnote is implicit while that to a hyperlink is explicit.

The Main Document part’s relationship file contains the following:

```
<Relationships …>
  <Relationship Id="rId5" Type="…/footnotes"
    Target="footnotes.xml"/>
  <Relationship Id="rId7" Type="…/hyperlink"
    Target="http://www.ecma-international.org/" TargetMode="External"/>
</Relationships>
```

All footnotes for a WordprocessingML document are contained in the same Footnotes part. Let’s look at how the Main Document refers to the footnote. At the point at which the footnote reference is inserted, the following XML is present:

```
<w:r>
     <w:footnoteReference w:id="2"/>
</w:r>
```

The w:id=“2” refers to the footnote with id=2 in the Footnotes part, the relevant piece of which is:

```
<w:footnote w:id="2">
  …
  Ecma is an international standards development organization (SDO).
  …
</w:footnote>
```

In the case of the hyperlink, the main document part makes an explicit reference to this relationship when it refers to the hyperlink, by using the following:

```
<w:hyperlink r:id="rId7" w:history="1">
  …
</w:hyperlink>
```

The important distinction here is that there is no explicit reference to a relationship ID designating the Footnotes part. The reference to the footnote with id=2 is “understood” to be in the Footnotes part that must always exist if there are any footnotes in the document. *end example*]

[*Example*: The following figure shows how the source, relationship item, and the target relate to each other for implicit and explicit relationships, respectively. The target does not have to be a file, however.

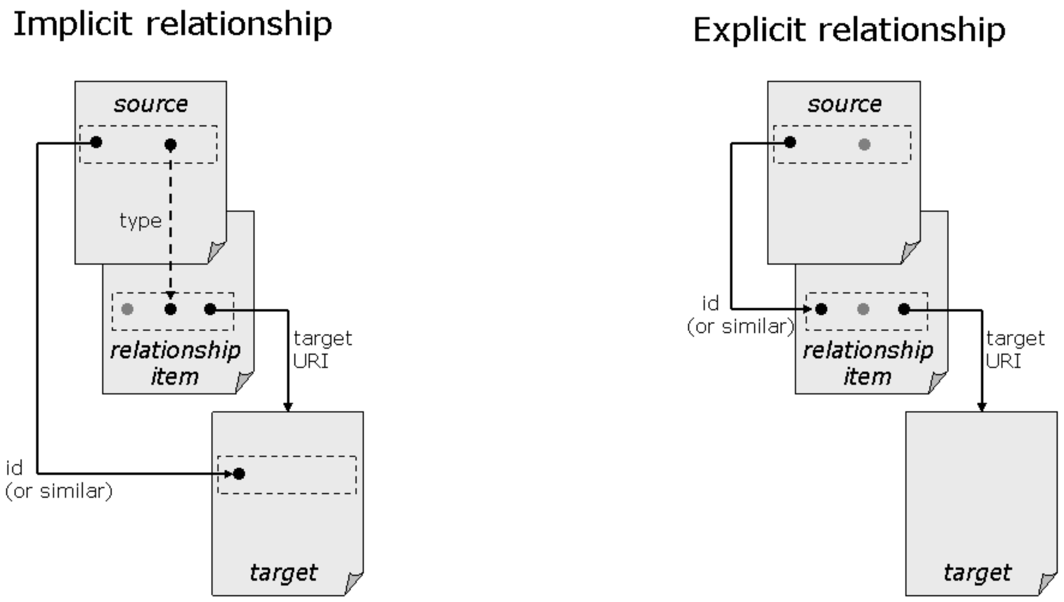
The dots correspond to attributes of relevant elements. Where one attribute refers to a piece in another part, this is indicated by arrows. Solid arrows indicate that the value of the source directly corresponds to the value at the target (for instance, id=rId4 in the source part corresponds to id=rId4 in the relationship item).

Dotted arrows indicate that the value of the source only implicit corresponds to the value of the target (for instance, "footnoteReference" in the source indicates the relationship type "footnotes" in the relationship item).

The main difference between the two types of relationship is that for implicit relationships, the id of the reference refers to an element with the same id in the target part, whereas for explicit relationships, the id refers to a relationship with the same id in the relationship item.

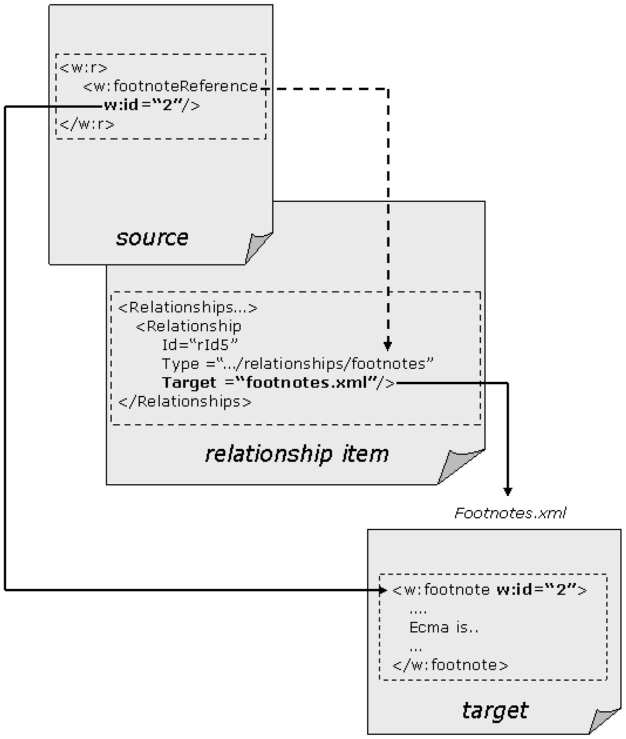
Both relationship types use the target URI of the relationship in the relationship item to locate the target.

For explicit relationships, the id in the source XML maps directly to the id of a relationship item with a direct explicit reference to the target. For implicit relationships, the relationship item is implied by the containing tag (e.g., footnote) and the id in the source XML is used to locate the correct element within the implied target.



end example]

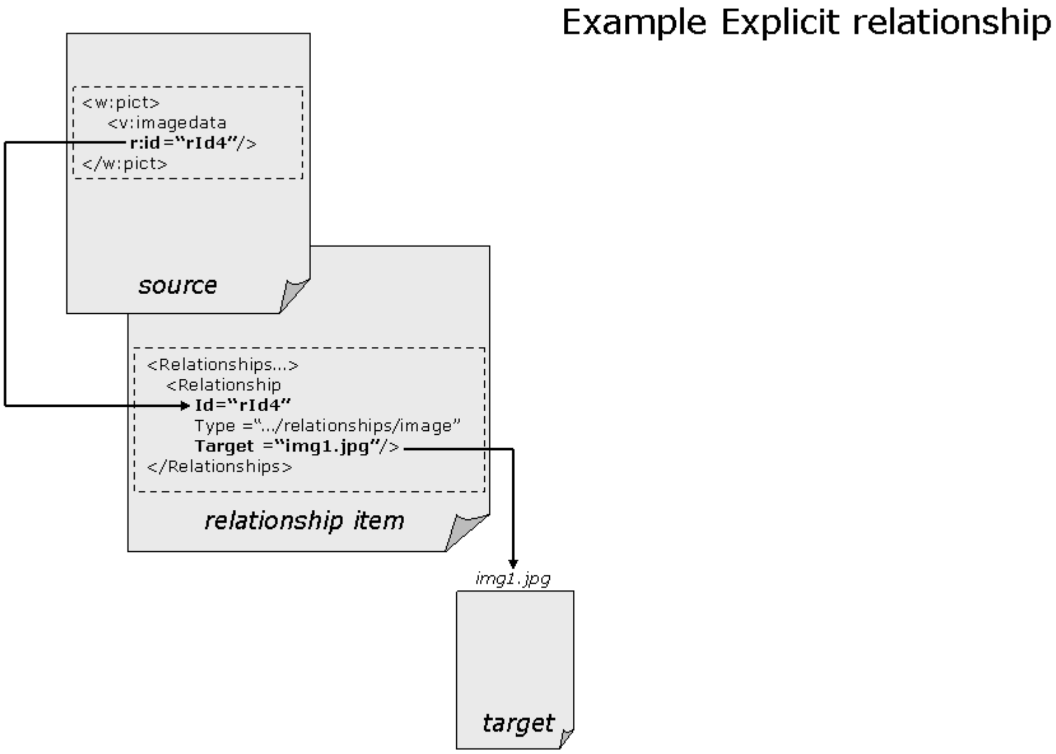
[*Example*: The following figure shows the implicit relationship for the footnote example described earlier.



Example Implicit relationship

end example]

[*Example*: The following figure shows an explicit relationship.



Example Explicit relationship

```
<w:pict>
    <v:imagedata
      r:id="rId4"/>
</w:pict>
```

```
<Relationships...>
   <Relationship
       Id="rId4"
       Type ="…/relationships/image"
       Target ="img1.jpg"/>
</Relationships>
```

end example]

10. Markup Compatibility and Extensibility

Office Open XML documents are designed to allow for innovation by extending their capabilities via a scheme defined by Part 3. This subclause contains information regarding Office Open XML's use of the Markup Compatibility constructs.

10.1 Constraints on Office Open XML's Use of Markup Compatibility and Extensibility

While the Markup Compatibility and Extensibility specification is designed for and used by Office Open XML documents, it could also be used to support a much broader range of applications. As a result, the use of some Markup Compatibility and Extensibility features is restricted within Office Open XML documents. These additional requirements are discussed in the following subordinate subclauses. Unless explicitly specified below, all normative requirements of the Markup Compatibility and Extensibility specification apply to Office Open XML documents.

10.1.1 PreserveElements and PreserveAttributes

The PreserveElements and PreserveAttributes elements, as defined in Part 3, allow a markup language to specify the conditions under which extensions should be round-tripped, even when their contents are edited. Within the context of the markup languages explicitly defined by ISO/IEC 29500, no such conditions are specified, and therefore applications are not obliged to support these hints at any point in an Office Open XML document. Instead, the well-defined extensibility constructs defined below should be used.

All other constructs defined in Part 3 shall be supported.

10.1.2 Office Open XML Native Extensibility Constructs

Clause 12 of Part 3 specifies the ability for a markup language to define additional constructs for extensibility of a specific markup language. Within the context of Office Open XML documents, the extLst element(s) defined in individual markup languages shall allow the round-tripping of all unknown content regardless of the state of the PreserveElements and PreserveAttributes elements. See the reference material in §17–23 for additional information on the XML elements that allow the use of the extLst construct.

11. WordprocessingML

This clause contains specifications for relationship items and parts that are specific to WordprocessingML. Parts that can occur in a WordprocessingML document, but are not WordprocessingML-specific, are specified in §15.2. Unless stated explicitly, all references to relationship items, content-type items, and parts in this clause refer to WordprocessingML ZIP items.

11.1 Glossary of WordprocessingML-Specific Terms

The following terms are used in the context of a WordprocessingML document:

document setting — A document-level property that affects the handling of a given document, and influences the appearance and behavior of the current document, as well as the stored document-level state.

document building block — A reusable element in a template. [*Note*: Such elements include boilerplate text, cover pages, equations, footers, headers, tables, text boxes, and watermarks. *end note*]

glossary document — An additional WordprocessingML document story used to store reusable fragments of rich WordprocessingML content. It is called the glossary document as this story contains one or more fragments that can be indexed and extracted by name, like items in a glossary.

master document — A document that is the parent of one or more subdocuments. [*Note*: A master document can be used to manage a multipart document, such as a book having several chapters. In such as case, the master document might contain the cover page, front matter, table of contents, and cross-reference index, while each chapter and appendix resides in its own subdocument. *end note*]

section — A portion of a document in which certain page formatting options can be set. [*Note*: A new section is created to change such properties as line numbering, number of columns, or headers and footers. *end note*]

subdocument — A piece of a master document. [*Note*: A chapter or appendix might be a subdocument in a book. *end note*]

supplementary document storage location — A part within a WordprocessingML document in which fragments of WordprocessingML content can be stored separate from the printed page. See also **glossary document**

template — A document that is a pattern for creating other documents. A template can contain text, formatting, and graphics, among other things, such that documents based on it automatically have access to these elements.

11.2 Package Structure

A WordprocessingML package shall contain a package-relationship item and a content-type item. The package-relationship item shall have implicit relationships with targets of the following type:

- One Main Document part (§11.3.10)

The package-relationship item is permitted to have implicit relationships with targets of the following type:

- Digital Signature Origin (§15.2.7)
- File Property parts (§15.2.12) (Application-Defined File Properties, Core File Properties, and Custom File Properties), as appropriate.
- Thumbnail (§15.2.16).

The required and optional relationships between parts are defined in §16.1 and its subordinate clauses.

[*Example*: The following package represents the minimal conformant WordprocessingML package as defined by ISO/IEC 29500:

First, the content type for relationship parts and the Main Document part (the only required part) must be defined (physically located at /[Content_Types].xml in the package):

```
<Types xmlns="http://schemas.openxmlformats.org/package/2006/content-types">
  <Default Extension="rels"
    ContentType="application/vnd.openxmlformats-
      package.relationships+xml"/>
  <Override PartName="/document.xml"
    ContentType="application/vnd.openxmlformats-
      officedocument.wordprocessingml.document.main+xml"/>
</Types>
```

Next, the single required relationship (the package-level relationship to the Main Document part) must be defined (physically located at /_rels/.rels in the package):

```
<Relationships xmlns="…">
  <Relationship Id="rId1"

Type="http://purl.oclc.org/ooxml/officeDocument/relationships/officeDocument"
    Target="document.xml"/>
</Relationships>
```

Finally, the minimum content for the Main Document part must be defined (physically located at /document.xml in the package):

```
<w:document xmlns:w="…">
  <w:body>
    <w:p/>
  </w:body>
</w:document>
```

end example]

[*Example*: Consider a WordprocessingML document that is an early draft of ISO/IEC 29500. Here's an example of the hierarchical folder structure that might be used for the ZIP items in the package for that document. As shown, one part, Main Document (stored in the ZIP item /word/document.xml), has its own relationship item:

```
/[Content_Types].xml       Content-type item
/_rels/.rels                     Package-relationship item
/docProps/app.xml                Application-Defined File Properties part
/docProps/core.xml               Core File Properties part
/word/document.xml               Main Document part
/word/_rels/document.xml.rels    Part-relationship item
/word/comments.xml               Comment part
/word/endnotes.xml               Endnotes part
/word/fontTable.xml              Font Table part
/word/footer1.xml                Footer parts
/word/footer2.xml
/word/footer3.xml
/word/footer4.xml
/word/footnotes.xml              Footnotes part
/word/header1.xml                Header parts
/word/header2.xml
/word/header3.xml
/word/header4.xml
/word/header5.xml
/word/header6.xml
/word/numbering.xml              Numbering Definitions part
/word/settings.xml               Document Settings part
/word/styles.xml                 Style Definitions part
/word/theme/theme1.xml           Theme part
```

The package-relationship item contains the following:

```
<Relationships xmlns="…">
  <Relationship Id="rId3"
    Type="http://…/extended-properties" Target="docProps/app.xml"/>
  <Relationship Id="rId2"
    Type="http://…/core-properties" Target="docProps/core.xml"/>
  <Relationship Id="rId1"
    Type="http://…/officeDocument" Target="word/document.xml"/>
</Relationships>
```

end example]

11.3 Part Summary

The subclauses subordinate to this one describe in detail each of the part types specific to WordprocessingML. [*Note*: For convenience, information from those subclauses is summarized in the following table:

Part	Relationship Target of	Root Element	Ref.
Alternative Format Import	Comments, Endnotes, Footer, Footnotes, Header, or Main Document	Not applicable	§11.3.1
Comments	Glossary Document or Main Document	comments	§11.3.2
Document Settings	Glossary Document or Main Document	settings	§11.3.3
Endnotes	Glossary Document or Main Document	endnotes	§11.3.4
Font Table	Glossary Document or Main Document	fonts	§11.3.5
Footer	Glossary Document or Main Document	ftr	§11.3.6
Footnotes	Glossary Document or Main Document	footnotes	§11.3.7
Glossary Document	Main Document	glossaryDocument	§11.3.8
Header	Glossary Document or Main Document	hdr	§11.3.9
Main Document	WordprocessingML package	document	§11.3.10
Numbering Definitions	Glossary Document or Main Document	numbering	§11.3.11
Style Definitions	Glossary Document or Main Document	styles	§11.3.12
Web Settings	Glossary Document or Main Document	webSettings	§11.3.13

end note]

11.3.1 Alternative Format Import Part

Content Type:	Any text-based content, support for which is application-defined. [*Note*: Some examples of formats which might be supported include: • Text = text/plain • HTML = text/html • WordprocessingML = application/vnd.openxmlformats-officedocument.wordprocessingml.document • XHTML = application/xhtml+xml

©ISO/IEC 2012 – All rights reserved

	end note]
Root Namespace:	not applicable
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/aFChunk

An alternative format import part allows content specified in an alternate format specified above to be embedded directly in a WordprocessingML document in order to allow that content to be migrated to the WordprocessingML format.

Any document part that permits a p element can also contain an altChunk element, whose id attribute refers to a relationship. That relationship shall target a part within the package, which contains the content to be imported into this WordprocessingML document.

A package is permitted to contain zero or more Alternative Format Import parts, each of which shall have a corresponding alternate format file that is the target of an explicit relationship from a Comments (§11.3.2), Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), or Main Document (§11.3.10) part. This relationship shall be explicitly referenced using its relationship ID in the source part using the appropriate XML syntax (i.e.; in the id attribute on the altChunk element), and the presence of this relationship without such a reference shall be considered non-conformant.

ISO/IEC 29500 does not specify how one might create a WordprocessingML package that contains Alternative Format Import relationships and altChunk elements.

The following requirements are applied to applications with respect to this part:

- An application that is solely a conforming consumer shall not reject documents containing one or more instances of this part
- An application that is both a conforming consumer and producer shall not reject documents containing instances of this part and shall convert/remove any instances of this part before acting as producer.
- An application that is solely a conforming producer shall not create a WordprocessingML package that contains Alternative Format Import relationships and elements.

[*Note*: The Alternative Format Import machinery provides a one-time conversion facility. A producer could have an extension that allows it to generate a package containing these relationships and elements, yet when run in conforming mode, does not do so. *end note*]

[*Example*: The following Main Document part-relationship item contains a relationship to an Alternative Format Import part:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/aFChunk" Target="Demo.html"
    TargetMode="Internal"/>
</Relationships>
```

The Main Document part contains the following XML fragment:

```
<w:body>
  …
  <w:p/>
  <w:altChunk r:id="rId5"/>
  <w:p/>
  …
</w:body>
```

which results in the entire contents of Demo.html being converted and brought into the document at that point (assuming that the content type of Demo.html is supported by the application consuming this WordprocessingML file). *end example*]

An Alternative Format Import part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

An Alternative Format Import part shall not have any explicit or implicit relationships to parts defined by ISO/IEC 29500.

A producer that wants interoperability should use one of the following standard formats:

- HTML - application/text/html
- TEXT - application/text/plain (UTF-16)

11.3.2 Comments Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.comments+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/comments

An instance of this part type contains the information about each comment in the document.

A package shall contain no more than two Comments parts. If it exists, one instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

©ISO/IEC 2012 – All rights reserved

[*Example*: The following Main Document part-relationship item contains a relationship to the Contents part, which is stored as the ZIP item comment.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId93"
    Type="http://…/comments" Target="comments.xml"/>
</Relationships>
```

end example]

The root element for a Comment part shall be comments.

[*Example*:

```
<w:comments … >
  <w:comment>
    …
  </w:comment>
  …
</w:comments>
```

end example]

The XML markup for a comment in a Main Document part uses the commentReference element.

[*Example*: Consider the case in which the Main Document part contains the text "… in the Standard.", and there is an comment inserted immediately after the period:

```
<w:p …>
  …
  <w:r>
    <w:t>… in the Standard.</w:t>
  </w:r>
  <w:r>
    <w:commentReference w:id="1"/>
  </w:r>
</w:p>
```

end example]

Each comment has a corresponding comment element in the Comments part, which contains the text of the comment.

[*Example*: The text of the comment is "This is my comment.":

```
<w:comments xmlns:w="…"
  <w:comment w:id="1">
    <w:p>
      <w:r>
        <w:t>This is my comment.</w:t>
    </w:p>
  </w:comment>
</w:comments>
```

end example]

A Comments part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Comments part is permitted to contain explicit relationships to the following parts defined by ISO/IEC 29500:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlinks (§15.3)
- Images (§15.2.14)
- Video (§15.2.17)

A Comments part shall not have any implicit or explicit relationships to any other part defined by ISO/IEC 29500.

11.3.3 Document Settings Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.settings+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/settings

An instance of this part type contains all the document's properties.

A package shall contain no more than two Document Settings parts. If it exists, one instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*: The following Main Document part-relationship item contains a relationship to a Document Settings part, which is stored in the ZIP item documentProperties1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/settings" Target="settings.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be settings.

[*Example*:

```
<w:settings … >
  …
  <w:defaultTabStop w:val="360"/>
  <w:footnotePr>
    …
  </w:footnotePr>
  <w:endnotePr>
  …
  </w:endnotePr>
  <w:rsids>
  …
  </w:rsids>
  …
</w:settings>
```

end example]

A Document Settings part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be Internal).

A Document Settings part is permitted to contain explicit relationships to the following parts defined by ISO/IEC 29500:

- Document Template (§11.4)
- Mail Merge Data Source (§11.7)
- Mail Merge Header Data Source (§11.8)
- XSL Transformation (§11.9)

A Document Settings part shall not have any implicit or explicit relationships to any other part defined by ISO/IEC 29500.

11.3.4 Endnotes Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.endnotes+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/endnotes

An instance of this part type contains all the endnotes for the document.

A package shall contain no more than two Endnotes parts. If it exists, one instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*: The following Main Document part-relationship item contains a relationship to the Endnotes part, which is stored as the ZIP item endnotes.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId6"
    Type="http://…/endnotes" Target="endnotes.xml"/>
</Relationships>
```

end example]

The root element for an Endnotes part shall be endnotes.

[*Example*:

```
<w:endnotes xmlns:w="…" …>
  <w:endnote …>
    …
  </w:endnote>
  <w:endnote …>
    …
  </w:endnote>
</w:endnotes>
```

end example]

The XML markup for an endnote in a Main Document part uses the endnoteReference element.

[*Example*: Consider the case in which the Main Document part contains the text "… in the Standard.", and there is an endnote inserted immediately after the period:

©ISO/IEC 2012 – All rights reserved

```
<w:p …>
  …
  <w:r>
    <w:t>… in the Standard.</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="EndnoteReference"/>
    </w:rPr>
    <w:endnoteReference w:id="5"/>
  </w:r>
</w:p>
```

end example]

Each endnote has a corresponding endnote element in the Endnotes part, which contains the text of the endnote, and the endnoteRef element.

[*Example*: The text of the endnote is "This can be downloaded from http://www.aabbcc.com/index.html." where "http://www.aabbcc.com/index.html" is marked as a hyperlink:

```
<w:endnotes xmlns:w="…">
  <w:endnote w:id="5">
    <w:p>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="EndnoteReference"/>
        </w:rPr>
        <w:endnoteRef/>
      </w:r>
      <w:r>
        <w:t xml:space="preserve"> This can be downloaded from </w:t>
      </w:r>
      <w:hyperlink r:id="rId2">
        <w:r>
          <w:rPr>
            <w:rStyle w:val="Hyperlink"/>
          </w:rPr>
          <w:t>http://www.aabbcc.com/index.html</w:t>
        </w:r>
      </w:hyperlink>
```

```
        <w:r>
          <w:t>.</w:t>
      </w:p>
    </w:endnote>
  </w:endnotes>
```

end example]

An Endnotes part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

An Endnotes part is permitted to contain explicit relationships to the following parts defined by ISO/IEC 29500:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlinks (§15.3)
- Images (§15.2.14)
- Video (§15.2.17)

An Endnotes part shall not have any implicit or explicit relationships to any other part defined by ISO/IEC 29500.

11.3.5 Font Table Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.fontTable+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/fontTable

An instance of this part type contains information about each of the fonts used by content in the document. When a consumer reads a WordprocessingML document, it shall use this information to determine which fonts to use to display the document when the specified fonts are not available on the consumer’s system.

A package shall contain no more than two Font Table parts. If it exists, one instance of that part shall be the target of an implicit relationship in the part-relationship item for the Main Document (§11.3.10) part, and the other instance shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*: The following Main Document part-relationship item contains a relationship to the Font Table part, which is stored as the ZIP item fontTable.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/fontTable" Target="fontTable.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be fonts.

[*Example*:

```
<w:fonts … >
  <w:font w:name="Calibri">
    <w:panose1 w:val="020F0502020204030204"/>
    <w:charset w:val="00"/>
    <w:family w:val="swiss"/>
    <w:pitch w:val="variable"/>
    <w:sig w:usb0="A00002EF" w:usb1="4000207B" w:usb2="00000000"
      w:usb3="00000000" w:csb0="0000009F" w:csb1="00000000"/>
  </w:font>
</w:fonts>
```

end example]

A Font Table part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Font Table part is permitted to contain explicit relationships to the following parts defined by ISO/IEC 29500:

- Fonts (§15.2.13)

A Font Table part shall not have any implicit or explicit relationships to any other part defined by ISO/IEC 29500.

11.3.6 Footer Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.footer+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/footer

An instance of this part type contains the information about a footer displayed for one or more sections.

A package is permitted to contain zero or one Footer part for each kind of footer (first page, odd page, or even page) in each section of the document. Each Footer part shall be the target of an explicit relationship in the part-relationship item for the Main Document (§11.3.10) part, or the Glossary Document (§11.3.8) part.

[*Example*: The Main Document part-relationship item contains one relationship, for the odd footer part, which is stored as the ZIP item footer3.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId91"
    Type="http://…/footer" Target="footer3.xml"/>
</Relationships>
```

end example]

The root element for a Footer part shall be ftr.

[*Example*:

```
<w:ftr xmlns:w="…" …>
  …
</w:ftr>
```

end example]

The XML markup for a footer in a section of a Main Document part involves the footerReference element in that section's sectPr element which explicitly references the relationship for the header.

[*Example*: Consider the case in which a section in the Main Document part contains odd and even headers, and an odd footer:

```
<w:document xmlns:w="…">
  …
  <w:sectPr>
    <w:footerReference w:val="rId89" w:type="default"/>
    <w:footerReference w:val="rId90" w:type="even"/>
    <w:footerReference w:val="rId91" w:type="first"/>
    <w:type w:val="oddPage"/>
    <w:pgSz w:w="11909" w:h="16834" w:code="9"/>
    <w:pgMar w:top="1440" w:right="1152" w:bottom="1440"
      w:left="1152" w:header="720" w:footer="720" w:gutter="0"/>
    <w:lnNumType w:countBy="1"/>
    <w:pgNumType w:numFmt="lowerRoman"/>
    <w:cols w:space="720"/>
  </w:sectPr>
</w:document>
```

end example]

Each footer has a corresponding ftr element in a Footer part, which contains the text of the footer.

[*Example*: Here is the odd footer corresponding to the example above. It has the page number centered and displayed using lowercase Roman numerals (as set by the pgNumType element above):

```
<w:ftr xmlns:w="…">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="Centered"/>
    </w:pPr>
    <w:fldSimple w:instr="PAGE">
      <w:r>
        <w:rPr>
          <w:noProof/>
        </w:rPr>
        <w:t>i</w:t>
      </w:r>
    </w:fldSimple>
  </w:p>
</w:ftr>
```

end example]

A Footer part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Footer part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlinks (§15.3)
- Images (§15.2.14)
- Video (§15.2.17)

A Footer part shall not have any implicit or explicit relationships to any other part defined by ISO/IEC 29500.

11.3.7 Footnotes Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.footnotes+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/footnotes

An instance of this part type contains all the footnotes for the document.

A package shall contain no more than two Footnotes parts. If it exists, one instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*: The Main Document part-relationship item contains a relationship to the Footnotes part, which is stored as the ZIP item footnotes.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/footnotes" Target="footnotes.xml"/>
</Relationships>
```

end example]

The root element for a Footnotes part shall be footnotes.

[*Example*:

```
<w:footnotes xmlns:w="…" …>
  <w:footnote …>
    …
  </w:footnote>
  <w:footnote …>
    …
  </w:footnote>
</w:footnotes>
```

end example]

The XML markup for a footnote in a Main Document part involves the footnoteReference element.

[*Example*: Consider the case in which the Main Document part contains the text "… in the Standard.", and there is a footnote inserted immediately after the period:

```
<w:p …>
  …
  <w:r>
    <w:t>… in the Standard.</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="FootnoteReference"/>
    </w:rPr>
    <w:footnoteReference w:id="5"/>
  </w:r>
</w:p>
```

end example]

Each footnote has a corresponding footnote element in the Footnotes part, which contains the text of the footnote and the footnoteRef element.

[*Example*: The text of the footnote is "This can be downloaded from http://www.aabbcc.com/index.html." where "http://www.aabbcc.com/index.html" is marked as a hyperlink:

```
<w:footnotes xmlns:w="…"
  <w:footnote w:id="5">
    <w:p>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="FootnoteReference"/>
        </w:rPr>
        <w:footnoteRef/>
      </w:r>
      <w:r>
        <w:t xml:space="preserve">This can be downloaded from </w:t>
      </w:r>
      <w:hyperlink r:id="rId2" w:history="1">
        <w:r>
          <w:rPr>
            <w:rStyle w:val="Hyperlink"/>
          </w:rPr>
          <w:t>http://www.aabbcc.com/index.html</w:t>
        </w:r>
      </w:hyperlink>
```

```
        <w:r>
          <w:t>.</w:t>
        </w:r>
      </w:p>
    </w:footnote>
  </w:footnotes>
```

end example]

A Footnotes part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Footnotes part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlinks (§15.3)
- Images (§15.2.14)
- Video (§15.2.17)

A Footer part shall not have any implicit or explicit relationships to any other part defined by ISO/IEC 29500.

11.3.8 Glossary Document Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.document.glossary+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/glossaryDocument

An instance of this part type is a supplementary document storage location which stores the definition and content for content that shall be carried with the document for future insertion and/or use, but which shall not be visible within the contents of the main document story. [*Example*: A legal contract template might include one or more optional clauses that shall not appear in the document until those clauses are inserted explicitly via a user action. To store these optional clauses until they are inserted, their contents are placed in the glossary document part. *end example*]

[*Note*: This part is intended for storage of optional "document fragments" which are often used to perform document assembly. The use of the word *glossary* is a reference to the fact that each of these entries was historically referenced by its first word in legacy word processing applications, like definitions of terms in a traditional glossary. *end note*]

The root element for a part of this content type shall be glossaryDocument.

[*Example*: The following part contains two building blocks. The first block is named "rainbow colors", belongs to a category called "Misc", belongs to a gallery called "docParts", and contains the text "The colors … and violet." The details of the second block have been omitted:

```
<w:glossaryDocument xmlns:w="…" >
  <w:docParts>
    <w:docPart>
      <w:docPartPr>
        <w:name w:val="rainbow colors"/>
        <w:style w:val="Normal"/>
        <w:category>
          <w:name w:val="Misc"/>
          <w:gallery w:val="docParts"/>
        </w:category>
      </w:docPartPr>
      <w:docPartBody>
        <w:p>
          <w:r>
            <w:t>The colors of the rainbow are red, orange, yellow,
              green, blue, indigo, and violet.</w:t>
          </w:r>
        </w:p>
      </w:docPartBody>
    </w:docPart>
    <w:docPart>
      …
    </w:docPart>
  </w:docParts>
</w:glossaryDocument>
```

end example]

A package shall contain at most one Glossary Document part, and that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part.

[*Example*: The following Main Document part-relationship item contains a relationship to a Glossary Document part, which is stored in the ZIP item `glossary/document.xml`:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/glossaryDocument" Target="glossary/document.xml"/>
</Relationships>
```

end example]

A Glossary Document part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Glossary Document part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Comments (§11.3.2)
- Document Settings (§11.3.3)
- Endnotes (§11.3.4)
- Font Table (§11.3.5)
- Footnotes (§11.3.7)
- Numbering Definitions (§11.3.11)
- Style Definitions (§11.3.11)

A Glossary Document part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors (§14.2.3), Diagram Data (§14.2.4), Diagram Layout Definition (§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Footer (§11.3.6)
- Header (§11.3.9)
- Hyperlinks (§15.3)
- Images (§15.2.14)
- Printer Settings (§15.2.15)
- Video (§15.2.17)

A Glossary Document part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

©ISO/IEC 2012 – All rights reserved

11.3.9 Header Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.header+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/header

An instance of this part type contains the information about a header displayed for one or more sections.

A package shall contain zero or one Header part for each kind of header (first page, odd page, or even page) in each section of the document. Each Header part shall be the target of an explicit relationship from the Main Document (§11.3.10) part or the Glossary Document (§11.3.8) part.

[*Example*: The Main Document part-relationship item contains two relationships: one for the even header part (which is stored as the ZIP item header2.xml) and one for the odd header part (which is stored as the ZIP item header3.xml):

```
<Relationships xmlns="…">
  <Relationship Id="rId89" Type="http://…/header" Target="header2.xml"/>
  <Relationship Id="rId90" Type="http://…/header" Target="header3.xml"/>
</Relationships>
```

end example]

The root element for a Header part shall be hdr.

[*Example*:

```
<w:hdr xmlns:w="…" …>
  …
</w:hdr>
```

end example]

The XML markup for a header in a section of a Main Document part involves the headerReference element in that section's sectPr element.

[*Example*: Consider the case in which a section in the Main Document part contains odd and even headers, and an odd footer:

```
<w:body>
  …
  <w:sectPr w:rsidR="00363F31" w:rsidSect="008D4B40">
    <w:headerReference w:val="rId89" w:type="default"/>
    <w:headerReference w:val="rId90" w:type="even"/>
    <w:headerReference w:val="rId91" w:type="first"/>
    <w:type w:val="oddPage"/>
    <w:pgSz w:w="11909" w:h="16834" w:code="9"/>
    <w:pgMar w:top="1440" w:right="1152" w:bottom="1440"
      w:left="1152" w:header="720" w:footer="720" w:gutter="0"/>
    <w:lnNumType w:countBy="1"/>
    <w:pgNumType w:fmt="lowerRoman"/>
    <w:cols w:space="720"/>
  </w:sectPr>
</w:body>
```

end example]

Each header has a corresponding hdr element in a Header part, which contains the text of the header.

[*Example*: Here is the even header corresponding to the examples above:

```
<w:hdr xmlns:w="…">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="Header"/>
    </w:pPr>
    <w:r>
      <w:t>My Test Document</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Here is the odd header corresponding to the examples above:

```
<w:hdr xmlns:w="…">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="Header"/>
    </w:pPr>
    <w:r>
      <w:tab/>
      <w:t>Table of Contents</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

end example]

A Header part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Header part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlinks (§15.3).
- Images (§15.2.14)
- Video (§15.2.17)

A Header part shall not have any implicit or explicit relationships to other parts defined by ISO/IEC 29500.

11.3.10 Main Document Part

Content Type(s):	application/vnd.openxmlformats-officedocument.wordprocessingml.document.main+xml application/vnd.openxmlformats-officedocument.wordprocessingml.template.main+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/officeDocument

An instance of this part type contains the body of the document.

A package shall contain a Main Document part (§11.3.10) part. The Main Document part shall be the target of a relationship in the package-relationship item.

The root element for a part of this content type shall be document.

[*Example*: Given the following package-relationship item excerpt:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/officeDocument" Target="word/document.xml"/>
</Relationships>
```

/word/document.xml" contains the following:

```
<w:document …>
  <w:body>
    …
  </w:body>
</w:document>
```

end example]

A Main Document part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Main Document part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Comments (§11.3.2)
- Custom XML Data Storage (§15.2.4)
- Document Settings (§11.3.3)
- Endnotes (§11.3.4)
- Font Table (§11.3.5)
- Footnotes (§11.3.7)
- Glossary Document (§11.3.8)
- Numbering Definitions (§11.3.11)
- Style Definitions (§11.3.12)
- Theme (§14.2.7)
- Thumbnail (§15.2.16)

A Main Document part is permitted to contain explicit relationships to the following parts defined by ISO/IEC 29500:

- Alternative Format Import (§11.3.1)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Footer (§11.3.6)
- Header (§11.3.9)
- Hyperlinks (§15.3)
- Images (§15.2.14)
- Printer Settings (§15.2.15)
- Subdocument (§11.6)
- Video (§15.2.17)

A Main Document shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

11.3.11 Numbering Definitions Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.numbering+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/numbering

An instance of this part type contains a definition for the structure of each unique numbering definition in this document.

[*Example:* If a set of paragraphs are added to a document which have a circle bullet at the first level, a square bullet at the second level, and a checkmark bullet at the third level, such as the following:

- First level
  - ▪ Second level
    - ✓ Third level

The numbering definition part contains the definition for each of these levels (their bullet style, indent, etc.) even if the second and third levels are not actually used in the document *end example*]

A package shall contain no more than two Numbering Definitions parts. If they exist, one instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/numbering" Target="numbering.xml"/>
</Relationships>
```

end example]

The XML markup for a list usage involves a reference to a numbering definition via the child elements of the numPr element.

[*Example*: Here we have a paragraph set using the style Text, followed by a list of things which have the paragraph style ListBullet, followed by another paragraph set using the style Text:

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="Text"/>
  </w:pPr>
  <w:r>
    <w:t>The kinds of fruit needed are:</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:pStyle w:val="ListBullet"/>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Apples</w:t>
  </w:r>
</w:p>
```

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="ListBullet"/>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Oranges</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:pStyle w:val="Text"/>
  </w:pPr>
  <w:r>
    <w:t>Other items may be needed too.</w:t>
  </w:r>
</w:p>
```

end example]

The root element for a Numbering Definition part shall be numbering, with each numbering definition being defined by an abstractNum element.

[*Example*:

```
<w:numbering xmlns:w="…">
  <w:abstractNum w:numId="11">
    <w:nsid w:val="394E2425"/>
    <w:multiLevelType w:val="hybridMultilevel"/>
    <w:tmpl w:val="F628E89A"/>
    <w:lvl w:ilvl="0" w:tplc="151C4798">
      <w:start w:val="1"/>
      <w:numFmt w:val="bullet"/>
      <w:pStyle w:val="ListBullet"/>
      <w:lvlText w:val="…"/>
      <w:lvlJc w:val="start"/>
      <w:pPr>
        <w:tabs>
          <w:tab w:val="num" w:pos="720"/>
        </w:tabs>
        <w:ind w:start="720" w:hanging="360"/>
      </w:pPr>
```

```
        <w:rPr>
          <w:rFonts w:ascii="Symbol" w:hAnsi="Symbol" w:hint="default"/>
        </w:rPr>
      </w:lvl>
      …
    </w:abstractNum>
  </w:numbering>
```

end example]

A Numbering Definitions part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Numbering Definitions part is permitted to contain explicit relationships to the following parts defined by ISO/IEC 29500:

- Images (§15.2.14)

A Numbering Definitions part shall not have any implicit or explicit relationships to any other part defined by ISO/IEC 29500.

11.3.12 Style Definitions Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.styles+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/styles

An instance of this part type contains the definition for a set of styles used by this document.

A package shall contain at most two Style Definitions parts. One instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship in from the Glossary Document (§11.3.8) part.

[*Example*:

```
  <Relationships xmlns="…">
    <Relationship Id="rId3"
      Type="http://…/styles" Target="styles.xml"/>
  </Relationships>
```

end example]

The root element for a Styles Definition part shall be styles, which is a container for one or more style elements.

©ISO/IEC 2012 – All rights reserved

[*Example*: Here is the style ListBullet (which is used in a Main Document Part in §11.3.10):

```
<w:styles xmlns:w="…" … xml:space="preserve">
  <w:style w:type="paragraph" w:styleId="ListBullet">
    <w:name w:val="List Bullet"/>
    <w:basedOn w:val="Text"/>
    <w:autoRedefine/>
    <w:rsid w:val="00081289"/>
    <w:pPr>
      <w:pStyle w:val="ListBullet"/>
      <w:numPr>
        <w:numId w:val="1"/>
      </w:numPr>
      <w:tabs>
        <w:tab w:val="clear" w:pos="360"/>
      </w:tabs>
      <w:ind w:start="648"/>
    </w:pPr>
  </w:style>
</w:styles>
```

end example]

A Style Definitions part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be Internal).

A Style Definitions part shall not have implicit or explicit relationships to any part defined by ISO/IEC 29500.

11.3.13 Web Settings Part

Content Type:	application/vnd.openxmlformats-officedocument.wordprocessingml.webSettings+xml
Root Namespace:	http://purl.oclc.org/ooxml/wordprocessingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/webSettings

An instance of this part type contains the definition for web-specific settings used by this document.

A package shall contain at most two Web Settings parts. One instance of that part shall be the target of an implicit relationship from the Main Document (§11.3.10) part, and the other shall be the target of an implicit relationship from the Glossary Document (§11.3.8) part.

[*Example*:

```
<Relationships xmlns="…">
  <Relationship Id="rId3"
    Type="http://…/webSettings" Target="webSettings.xml"/>
</Relationships>
```

end example]

The root element for a Web Settings part shall be webSettings.

[*Example*:

```
<w:webSettings …>
  <w:frameset>
    …
    <w:frame>
      <w:sz w:val="216" />
      <w:name w:val="Frame2" />
      <w:sourceFileName r:id="rId1" />
    </w:frame>
    <w:frame>
      <w:name w:val="Frame1" />
      <w:sourceFileName r:id="rId2" />
    </w:frame>
  </w:frameset>
</w:webSettings>
```

end example]

A Web Settings part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Web Settings part is permitted to contain explicit relationships to the following parts defined by ISO/IEC 29500:

- Frameset (§11.5)

A Web Settings part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

11.4 Document Template

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/attachedTemplate

A *document template* can be represented by an instance of a WordprocessingML package, and contains styles, numbering definitions, and so on that are made available when documents based on that template are edited. A WordprocessingML document can refer to another document as its document template, by having a Document

Settings part (§11.3.3) that contains an explicit relationship to the file location of the necessary document template using the id attribute on the attachedTemplate element.

[*Example*: Consider a document specifying a document template located at `c:\template.docx`:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/attachedTemplate" Target="file:///c:\template.docx"
    TargetMode="External"/>
</Relationships>
```

The document's Document Settings part contains an attachedTemplate element that explicitly references this relationship:

```
<w:settings … >
  <w:attachedTemplate r:id="rId1"/>
</w:settings>
```

end example]

11.5 Framesets

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/frame

A *frameset* is a WordprocessingML document which specifies the location and placement of other WordprocessingML documents (which, when used in this context, are referred to as *frames*). A frameset shall be represented by an instance of a WordprocessingML document with a Web Settings part (§11.3.13) whose relationship item targets each of that frameset's frames.

[*Example*: Consider a frameset document having two frames. The frameset's Web Settings part-relationships item contains the following, in which frame1.docx and frame2.docx are packages containing the corresponding frames:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/frame" Target="frame1.docx" TargetMode="External"/>
  <Relationship Id="rId2"
    Type="http://…/frame" Target="frame2.docx" TargetMode="External"/>
</Relationships>
```

The frameset document's Web Settings part contains a frameset element that references its frames:

```
<w:webSettings …>
  <w:frameset>
    …
    <w:frame>
      <w:sz w:val="216" />
      <w:name w:val="Frame2" />
      <w:sourceFileName r:id="rId1" />
    </w:frame>
    <w:frame>
      <w:name w:val="Frame1" />
      <w:sourceFileName r:id="rId2" />
    </w:frame>
  </w:frameset>
</w:webSettings>
```

end example]

A frame shall be represented by an instance of a WordprocessingML package.

A frame shall be located external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

11.6 Master Documents and Subdocuments

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/subDocument

A master document shall be represented by an instance of a WordprocessingML document whose Main Document (§11.3.10) part targets each of that master document's subdocuments.

[*Rationale*: Sometimes, it is convenient to deal with a document as a collection of pieces, especially when those pieces might be edited by different authors in a collaborative group. Perhaps it simply makes sense to think about a book as a collection of chapters rather than as one big document. The breaking-up of a document into such pieces can be achieved by having a master document with one or more subdocuments. *end rationale*]

[*Example*: Consider a master document, whose three subdocuments are called Start, Middle, and End, respectively. Master's Main Document part has a corresponding relationships part that contains the following, in which Start.docx, Middle.docx, and End.docx are packages containing the corresponding subdocuments:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/subDocument"
    Target="Start.docx" TargetMode="External"/>
```

©ISO/IEC 2012 – All rights reserved

```
    <Relationship Id="rId6"
      Type="http://…/SubDocument"
      Target="Middle.docx" TargetMode="External"/>
    <Relationship Id="rId7"
      Type="http://…/SubDocument"
      Target="End.docx" TargetMode="External"/>
  </Relationships>
```

The master document's Main Document part contains subDoc elements that reference its subdocuments:

```
  <w:document xmlns:r="…" xmlns:w="…" …>
    <w:body>
      <w:p …>
        <w:pPr>
          …
        </w:pPr>
      </w:p>
      <w:subDoc r:id="rId5"/>
      …
      <w:subDoc r:id="rId6"/>
      …
      <w:subDoc r:id="rId7"/>
      …
    </w:body>
  </w:document>
```

end example]

A subdocument shall be represented by an instance of a WordprocessingML package.

A subdocument shall be located external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

11.7 Mail Merge Data Source

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/mailMergeSource

A document that stores information about a mail merge operation is permitted to contain a Document Settings part (§11.3.3) whose relationship item targets the file location of the necessary data source using this relationship.

[*Example*: Consider a document specifying a mail merge whose data source is located at `http://www.openxmlformats.org/data.txt`:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/mailMergeSource"
    Target="http://www.openxmlformats.org/data.txt"
    TargetMode="External"/>
</Relationships>
```

The document's Document Settings part contains a dataSource element that explicitly references this relationship:

```
<w:settings …>
  <w:mailMerge>
    …
    <w:dataSource r:id="rId1" />
    …
  </w:mailMerge>
</w:settings>
```

end example]

A mail merge data source shall be located external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

11.8 Mail Merge Header Data Source

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/mailMergeHeaderSource

A document that stores information about a mail merge operation is permitted to contain a Document Settings part (§11.3.3) whose relationship item targets the file location of the necessary header data source using this relationship.

[*Example*: Consider a document specifying a mail merge whose header data source is located at `http://www.openxmlformats.org/header.txt`:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/mailMergeHeaderSource"
    Target="http://www.openxmlformats.org/header.txt"
    TargetMode="External"/>
</Relationships>
```

The document's Document Settings part contains a headerSource element that explicitly references this relationship:

©ISO/IEC 2012 – All rights reserved

```
<w:settings …>
  <w:mailMerge>
    …
    <w:headerSource r:id="rId2" />
    …
  </w:mailMerge>
</w:settings>
```

end example]

A mail merge header data source shall be located external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

11.9 XSL Transformation

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/transform

A document can store information about an XSL Transformation which might be applied when the document is output as a single file (e.g. as XML or HTML).That information is stored in a Document Settings part (§11.3.3) whose part relationship item contains an explicit relationship to the file location of the XSL Transformation using this relationship. [*Note*: A full description of how this relationship is used (in conjunction with the saveThroughXslt element) is provided in §**17.15.1.76**. *end note*]

[*Example*: Consider a document specifying an XSL Transformation located at `http://www.openxmlformats.org/test.xsl`:

```
<Relationships xmlns="…">
  <Relationship Id="rId8" Type="http://…/transform"
    Target="http://www.openxmlformats.org/test.xsl"
    TargetMode="External"/>
</Relationships>
```

The document’s Document Settings part contains a saveThroughXslt element that explicitly references this relationship:

```
<w:settings …>
  …
  <w:saveThroughXslt r:id="rId8" />
  …
</w:settings>
```

end example]

An XSL transformation shall be located external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

12. SpreadsheetML

This clause contains specifications for relationship items and parts that are specific to SpreadsheetML. Parts than can occur in a SpreadsheetML document, but are not SpreadsheetML-specific, are specified in §15.2. Unless stated explicitly, all references to relationship items, content-type items, and parts in this clause refer to SpreadsheetML ZIP items.

12.1 Glossary of SpreadsheetML-Specific Terms

The following terms are used in the context of a SpreadsheetML document:

cell — The location at the intersection of a row and column, in which numeric or textual data or a formula is stored. A cell can have a number of characteristics, such as numeric or text formatting, alignment, font, color, and border.

cell reference — An individual cell's designation using a combination of its column and row headings, as in A13, H19, and BX1200. A **relative cell reference** in a formula automatically changes when the formula is copied down a column or across a row. An **absolute cell reference** is fixed. Absolute references don't change when a formula is copied from one cell to another. A **mixed cell reference** has either an absolute column and a relative row, or an absolute row and a relative column.

chart — A graphical representation of data, as in a bar, column, line, pie chart, for example.

column — Any vertical set of cells in a worksheet. Each column has an alphabetic heading. Columns are named sequentially, going from A–Z, then AA–AZ, BA–BZ, and so on.

connection — The means by which external data—that is, data stored outside of a workbook (in a database or on a Web server, for example)—can be imported into a worksheet.

formula — A recipe for calculating a value. Some formulas are predefined; others are user-defined.

function — A predefined formula, such as AVERAGE, MAX, MIN, and SUM. A function takes one or more arguments on which it operates, producing a result. [*Note*: In the formula =SUM(B1:B4), there is one argument, B1:B4, which is the range of cells B1–B4, inclusive. *end note*]

pivot table — A kind of table that is used to manage and analyze related data that is stored elsewhere.

row — Any horizontal set of cells in a worksheet. Each row has a numeric heading. Rows are numbered sequentially, starting at 1.

table — A rectangular-shaped set of related rows and columns that can be sorted, filtered, and totaled as a group. Rows in a table can be hidden by applying **autofilters** to one or more columns.

workbook — A collection of worksheets.

worksheet — A two-dimensional grid of cells that are organized into rows and columns.

12.2 Package Structure

A SpreadsheetML package shall contain a package-relationship item and a content-type item. The package-relationship item shall have implicit relationships with targets of the following type:

- One Workbook part (12.3.23).

The package-relationship item is permitted to have implicit relationships with targets of the following type:

- Digital Signature Origin (§15.2.7)
- File Property parts (§15.2.12) (Application-Defined File Properties, Core File Properties, and Custom File Properties), as appropriate.
- Thumbnail (§15.2.16).

The required and optional relationships between parts are defined in §12.3 and its subordinate clauses.

[*Example*: The following package represents the minimal conformant SpreadsheetML package as defined by ISO/IEC 29500:

First, the content types for relationship parts, the Workbook part, and at least one Sheet part must be defined (physically located at /[Content_Types].xml in the package):

```
<Types xmlns="…">
  <Default Extension="rels"
   ContentType="application/vnd.openxmlformats-package.relationships+xml"
   />
  <Override PartName="/workbook.xml"
   ContentType="application/vnd.openxmlformats-officedocument.
   spreadsheetml.sheet.main+xml" />
  <Override PartName="/sheet1.xml"
   ContentType="application/vnd.openxmlformats-
   officedocument.spreadsheetml.worksheet+xml" />
</Types>
```

Next, the required package-level relationship to the Workbook part must be defined (physically located at /_rels/.rels in the package):

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
   Type=http://purl.oclc.org/ooxml/officeDocument/relationships/officeDocument"
   Target="workbook.xml" />
</Relationships>
```

Next, the minimum content for the Workbook part must be defined (physically located at /workbook.xml in the package):

```
<workbook xmlns="…" xmlns:r="…">
  <sheets>
    <sheet name="1" sheetId="1" r:id="rId1" />
  </sheets>
</workbook>
```

Next, the required workbook-level relationship to the single Sheet part must be defined, (physically located at /_rels/workbook.xml.rels in the package):

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
   Type="http://purl.oclc.org/ooxml/officeDocument/relationships/worksheet"
Target="sheet1.xml" />
</Relationships>
```

Finally, the minimum content for a single Sheet part must be defined (physically located at /sheet1.xml in the package):

```
<worksheet xmlns="…" xmlns:r="…">
  <sheetData />
</worksheet>
```

end example]

[*Example*: Consider a SpreadsheetML document that contains a workbook having three worksheets. Here's an example of the hierarchical folder structure that might be used for the ZIP items in the package for that document. As shown, one part, Workbook (stored in the ZIP item /xl/workbook.xml), has its own relationship item:

```
/_rels/.rels                        Package-relationship item
/[Content_Types].xml                Content-type item
/docProps/app.xml                   Application-Defined File Properties part
/docProps/core.xml                  Core File Properties part
/xl/workbook.xml                    Workbook part
/xl/_rels/workbook.xml.rels         Part-relationship item
/xl/calcChain.xml                   Calculation Chain part
/xl/sharedStrings.xml               Shared String Table part
/xl/styles.xml                      Styles part
/xl/volatileDependencies.xml        Volatile Dependencies part
/xl/theme/theme1.xml                Theme part
/xl/worksheets/sheet1.xml           Worksheet parts
/xl/worksheets/sheet2.xml
/xl/worksheets/sheet3.xml
```

The package-relationship item contains the following:

```
<Relationships xmlns="…">
  <Relationship Id="rId3"
    Type="http://…/extended-properties" Target="docProps/app.xml"/>
  <Relationship Id="rId2"
    Type="http://…/core-properties" Target="docProps/core.xml"/>
  <Relationship Id="rId1"
    Type="http://…/officeDocument" Target="xl/workbook.xml"/>
</Relationships>
```

end example]

12.3 Part Summary

The subclauses subordinate to this one describe in detail each of the part types specific to SpreadsheetML. [*Note*: For convenience, information from those subclauses is summarized in the following table:

Part	Relationship Target of	Root Element	Ref.
Calculation Chain	Workbook	calcChain	§12.3.1
Chartsheet	Workbook	chartsheet	§12.3.2
Comments	Dialogsheet, Worksheet	comments	§12.3.3
Connections	Workbook	connections	§12.3.4
Custom Property	Workbook	Not applicable	§12.3.5
Custom XML Mappings	Workbook	MapInfo	§12.3.6
Dialogsheet	Workbook	dialogsheet	§12.3.7
Drawings	Chartsheet, Worksheet	wsDr	§12.3.8
External Workbook References	Workbook	externalLink	§12.3.9
Metadata	Workbook	metadata	§12.3.10
Pivot Table	Worksheet	pivotTableDefinition	§12.3.11
Pivot Table Cache Definition	Pivot Table, Workbook	pivotCacheDefinition	§12.3.12
Pivot Table Cache Records	Pivot Table Cache Definition	pivotCacheRecords	§12.3.13
Query Table	Worksheet	queryTable	§12.3.14
Shared String Table	Workbook	sst	§12.3.15
Shared Workbook Revision Headers	Workbook	headers	§12.3.16
Shared Workbook	Shared Workbook	revisions	§12.3.17

©ISO/IEC 2012 – All rights reserved

Part	Relationship Target of	Root Element	Ref.
Revision Log	Revision Headers		
Shared Workbook User Data	Workbook	users	§12.3.18
Single Cell Table Definitions	Dialogsheet, Worksheet	singleXmlCells	§12.3.19
Styles	Workbook	styleSheet	§12.3.20
Table Definition	Dialogsheet, Worksheet	table	§12.3.21
Volatile Dependencies	Workbook	volTypes	§12.3.22
Workbook	SpreadsheetML package	workbook	§12.3.23
Worksheet	Workbook	worksheet	§12.3.24

end note]

12.3.1 Calculation Chain Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.calcChain+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/calcChain

An instance of this part type contains an ordered set of references to all cells in all worksheets in the workbook whose value is calculated from any formula. The ordering allows inter-related cell formulas to be calculated in the correct order when a worksheet is loaded for use.

A package shall contain no more than one Calculation Chain part. If it exists, that part shall be the target of an implicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains a relationship to the Calculation Chain part, which is stored in the ZIP item calcChain.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId7"
    Type="http://…/calcChain" Target="calcChain.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be calcChain.

[*Example*: Cells D8, E8, and F8 each contain a value that is the result of calculations that shall be performed in the order E8, D8, F8:

```
<calcChain xmlns="…">
  <c r="E8" i="1"/>
  <c r="D8"/>
  <c r="F8" s="1"/>
</calcChain>
```

end example]

A Calculation Chain part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Calculation Chain part shall not have implicit or explicit relationships to any part defined by ISO/IEC 29500.

12.3.2 Chartsheet Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.chartsheet+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/chartsheet

An instance of this part type represents a chart that is stored in its own sheet.

A package is permitted to contain zero or more Chartsheet parts. Each such part shall be the target of an explicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains three relationships to Chartsheet parts, which are stored in the ZIP items chartsheets/sheet*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/chartsheet" Target="chartsheets/sheet1.xml"/>
  <Relationship Id="rId5"
    Type="http://…/chartsheet" Target="chartsheets/sheet2.xml"/>
  <Relationship Id="rId6"
    Type="http://…/chartsheet" Target="chartsheets/sheet3.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be chartsheet.

[*Example*: sheet1.xml refers to a drawing that is the target of a relationship in the Chartsheet part's relationship item:

```
<chartsheet xmlns:r="…" …>
  <sheetViews>
    <sheetView scale="64"/>
  </sheetViews>
  <drawing r:id="rId1"/>
</chartsheet>
```

end example]

A Chartsheet part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Chartsheet part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Printer Settings (§15.2.15)

A Chartsheet part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Drawings (§12.3.8)

A Chartsheet part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.3 Comments Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.comments+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/comments

An instance of this part type contains all the comments for a given worksheet, as well as the names of the authors of those comments.

A package shall contain exactly one Comments part for each worksheet that contains one or more comments. If a Comments part exists, it shall be the target of an implicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Worksheet part-relationship item contains a relationship to the Comments part, which is stored in the ZIP item comments2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/comments" Target="../comments2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be comments.

[*Example*: This Comments part results from a workbook that has one or more comments from each of two people: James Jones and Mary Smith:

```
<comments xmlns:st="…" >
  <authors>
    <author>James Jones</author>
    <author>Mary Smith</author>
  </authors>
  <commentList>
    <comment r="C7" authorId="0">
      <text>
        <st:r>
          <st:rPr>
            …
          </st:rPr>
          <st:t>James Jones:</st:t>
        </st:r>
        <st:r>
          <st:rPr>
            …
          </st:rPr>
          <st:t>Check that this date is correct.</st:t>
        </st:r>
      </text>
    </comment>
    <comment r="E7" authorId="1">
      …
    </comment>
  </commentList>
</comments>
```

end example]

A Comments part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Comments part shall not implicit or explicit relationships to any part defined by ISO/IEC 29500.

12.3.4 Connections Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.connections+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/connections

An instance of this part type describes all of the connections currently established for a workbook.

A package shall contain no more than one Connections part, and that part shall be the target of an implicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains a relationship to the Connections part, which is stored in the ZIP item connections.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/connections" Target="connections.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be connections.

[*Example*: A workbook has three connections, two from one worksheet, and one from another. connections.xml defines these three connections:

```
<connections …>
  <connection id="1" odcFile="…" keepAlive="1" name="…" type="5"
    refreshedVersion="2" background="1" saveData="1">
    <dbPr connection="Provider=MSDASQL.1;Persist Security Info=True;Data
Source=dBASE Files;Extended Properties=&quot;DSN=dBASE Files;DBQ=E:\MY
DOCUMENTS;DefaultDir=E:\MY
DOCUMENTS;DriverId=533;MaxBufferSize=2048;PageTimeout=5;&quot;;Initial
Catalog=E:\MY DOCUMENTS" command="`E:\MY DOCUMENTS`\`ADDRESS`" commandType="3"/>
  </connection>
  <connection id="2" …>
    <dbPr … />
  </connection>
  <connection id="3" …>
    <dbPr … />
  </connection>
</connections>
```

end example]

A Connections part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Connections part shall not have implicit or explicit relationships to any part defined by ISO/IEC 29500..

12.3.5 Custom Property Part

Content Type:	Any content, support for which is application-defined. [*Note*: Some examples of formats which might be supported include: • application/vnd.openxmlformats-officedocument.spreadsheetml.customProperty • application/xml *end note*]
Root Namespace:	Not applicable
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/customProperty

This part supports the storage of user-defined data.

[*Note*: It is recommended that a Custom Property Part contain XML content for improved interoperability; however, there is no requirement on the format of the content contained in a Custom Property Part. *end note*]

A package is permitted to contain zero or more Custom Property parts, and each such part shall be the target of an implicit relationship from the Worksheet (§12.3.24) part.

[*Example*: The following Worksheet part-relationship item contains a relationship to the Custom Property part, which is stored in the ZIP item CustomProperty.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId7"
    Type="http://…/customProperty" Target="CustomProperty.xml"/>
</Relationships>
```

where the contents of CustomProperty.xml contain the following XML

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<CustomApplicationData xmlns="…">
    <CustomProperty name="PropertyName" value="PropertyValue" />
</CustomApplicationData>
```

end example]

A Custom Property part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Custom Property part shall not have implicit or explicit relationships to any part defined by ISO/IEC 29500.

12.3.6 Custom XML Mappings Part

Content Type:	application/xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/xmlMaps

An instance of this part type contains a schema for an XML file, and information on the behavior that is used when allowing this custom XML schema to be mapped into the spreadsheet.

A package shall contain no more than one Custom XML Mappings part, and that part shall be the target of an implicit relationship from the Workbook part (§12.3.23). The Worksheet part into which this data is imported shall also have a relationship file that targets one or more Table Definition (§12.3.21) parts and/or one or more Single Cell Table Defintions (§12.3.19) parts.

[*Example*: The following Workbook part-relationship item contains a relationship to the Custom XML Mappings part, which is stored in the ZIP item xmlMaps.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId9"
    Type="http://…/xmlMaps" Target="xmlMaps.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be MapInfo.

[*Example*: xmlMaps.xml contains the following:

```
<mapInfo SelectionNamespaces="">
  <Schema ID="Schema1">
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element nillable="true" name="names">
            <xsd:complexType>
           <xsd:sequence minOccurs="0">
             <xsd:element minOccurs="0" maxOccurs="unbounded"
               nillable="true" name="name" form="unqualified">
```

```
                  <xsd:complexType>
                    <xsd:sequence minOccurs="0">
                      <xsd:element minOccurs="0" nillable="true"
                        type="xsd:string" name="firstname"
                        form="unqualified"/>
                      <xsd:element minOccurs="0" nillable="true"
                        type="xsd:string" name="initial"
                        form="unqualified"/>
                      <xsd:element minOccurs="0" nillable="true"
                        type="xsd:string" name="lastName"
                        form="unqualified"/>
                   </xsd:sequence>
                  </xsd:complexType>
                </xsd:element>
              </xsd:sequence>
            </xsd:complexType>
          </xsd:element>
        </xsd:schema>
      </Schema>
      <Map ID="1" Name="names_Map" RootElement="names" SchemaID="Schema1"
        ShowImportExportValidationErrors="false" AutoFit="true"
        Append="false"
        PreserveSortAFLayout="true" PreserveFormat="true">
        <DataBinding FileBinding="Test.xml" DataBindingLoadMode="1"/>
      </Map>
    </mapInfo>
```

end example]

A Custom XML Mappings part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Custom XML Mappings part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.7 Dialogsheet Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.dialogsheet+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/dialogsheet

An instance of this part type contains information about a legacy custom dialog box for a user form.

©ISO/IEC 2012 – All rights reserved

A package is permitted to contain one or more Dialogsheet parts, and each such part shall be the target of an explicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains relationships to a Dialogsheet part, which is stored in the ZIP item dialogsheets/sheet1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/dialogsheet" Target="dialogsheets/sheet1.xml"/>
</Relationships>
```

The Workbook part contains the following:

```
<workbook xmlns:r="…" …>
  …
  <sheets>
    …
    <sheet name="Dialog1" sheetId="4" r:id="rId2"/>
  </sheets>
  …
</workbook>
```

end example]

The root element for a part of this content type shall be dialogsheet.

[*Example*: sheet1.xml contains the following:

```
<dialogsheet xmlns:r="…" …>
  <sheetPr>
    <pageSetUpPr/>
  </sheetPr>
  <sheetViews>
    …
  </sheetViews>
    …
  <legacyDrawing r:id="rId1"/>
</dialogsheet>
```

end example]

A Dialogsheet part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Dialogsheet part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Printer Settings (§15.2.15)

A Dialogsheet part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Embedded Control Persistence (§15.2.9)
- Drawings (§12.3.8)
- Embedded Object (§15.2.10)

A Dialogsheet part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.8 Drawings Part

Content Type:	application/vnd.openxmlformats-officedocument.drawing+xml
Root Namespace:	http://purl.oclc.org/ooxml/drawingml/spreadsheetDrawing
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/drawing

An instance of this part type contains the presentation and layout information for one or more drawing elements that are present on this worksheet.

A package is permitted to contain one or more Drawings parts, and each such part shall be the target of an explicit relationship from a Worksheet part (§12.3.24), or a Chartsheet part (§12.3.2). There shall be only one Drawings part per worksheet or chartsheet.

[*Example*: The following Chartsheet part-relationship item contains a relationship to a Drawings part, which is stored in the ZIP item ../drawings/drawing1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http:// …/drawing" Target="../drawings/drawing1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be wsDr.

[*Example*: drawing1.xml refers to a chart that is the target of a relationship in the Drawing part's relationship item:

```
<xdr:wsDr xmlns:xdr="…" xmlns:a="…">
  <xdr:absoluteAnchor>
    <xdr:pos x="1518046" y="-1443632"/>
    <xdr:extents cx="8587382" cy="5848945"/>
    <xdr:graphicFrame macro="">
      <xdr:nvGraphicFramePr>
        <xdr:cNvPr id="24" name="Chart 24" descr=""/>
        <xdr:cNvGraphicFramePr/>
      </xdr:nvGraphicFramePr>
      <xdr:xfrm>
        <a:off x="0" y="0"/>
        <a:ext cx="0" cy="0"/>
      </xdr:xfrm>
      <a:graphic>
        <a:graphicData uri="http://…/chart">
          <a:chart relId="rId1"/>
        </a:graphicData>
      </a:graphic>
    </xdr:graphicFrame>
    <xdr:clientData/>
  </xdr:absoluteAnchor>
</xdr:wsDr>
```

end example]

A Drawings part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Drawings part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Hyperlinks (§15.3)
- Images (§15.2.14)

A Drawings part shall not have any implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.9 External Workbook References Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.externalLink+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/externalLink

An instance of this part specifies information about data referenced in other SpreadsheetML packages.

[*Example*: Consider two workbooks, wb1 and wb2, stored in packages called wb1.xlsx and wb2.xlsx, respectively. The value of a cell on a worksheet in wb1 can be computed using the value of one or more cells in wb2. This is done by having wb1 contain an external reference to wb2. *end example*]

A package is permitted to contain one or more External Workbook References parts, and those parts shall be the target of an explicit relationship in the Workbook part (§12.3.23).

[*Example*: A Workbook part for wb1 contains the following, which indicates that somewhere in its three worksheets, an external reference is made to a target specified in relationship id rId4 of the part's relationship item:

```
<workbook xmlns:r="…"/>
  …
  <sheets>
    <sheet name="Sheet1" sheetId="1" r:id="rId1"/>
    <sheet name="Sheet2" sheetId="2" r:id="rId2"/>
    <sheet name="Sheet3" sheetId="3" r:id="rId3"/>
  </sheets>
  …
  <externalReferences>
    <externalReference r:id="rId4"/>
  </externalReferences>
  …
</workbook>
```

That part's relationship item contains the following:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/externalLink"
    Target="externalReferences/externalReference1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be externalLink.

[*Example*: externalReference1.xml contains:

```
<externalLink xmlns:r="…" … r:id="rId1">
  <externalBook>
    <sheetNames>
      <sheetName val="Sheet1"/>
      <sheetName val="Sheet2"/>
      <sheetName val="Sheet3"/>
    </sheetNames>
    <sheetDataSet>
      <sheetData sheetId="0">
        <row r="7">
          <cell r="C8">
            <v>0</v>
          </cell>
        </row>
      </sheetData>
      <sheetData sheetId="1"/>
      <sheetData sheetId="2"/>
    </sheetDataSet>
  </externalBook>
</externalLink>
```

This part's relationship item contains the following:

```
<Relationships …>
  <Relationship Id="rId1"
    Type="…/externalReference"
    Target="wb2.xlsx" TargetMode="External"/>
</Relationships>
```

where wb2.xlsx is the workbook in which one or more cells' values are used in calculating the values of a cell in workbook wb1. *end example*]

An External Workbook References part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

An External Workbook References part shall specify an explicit relationship to one or more External Workbooks (§12.4).

An External Workbook References part shall not have any implicit or explicit relationships to other parts defined by ISO/IEC 29500.

12.3.10 Metadata Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.sheetMetadata+xml
Root	http://purl.oclc.org/ooxml/spreadsheetml/main

Namespace:	
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/sheetMetadata

An instance of this part type contains information relating to a cell whose value is related to one or more other cells via OnLine Analytical Processing (OLAP) technology.

A package shall contain no more than one Cell Metadata part, and that part shall be the target of an implicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains a relationship to the Metadata part, which is stored in the ZIP item metadata.xml. Cell B3 contains the formula CUBEMEMBER `("externalData", "[Account].[All Account]")`:

```
<Relationships xmlns="…">
  <Relationship Id="rId10"
    Type="http://…/sheetMetadata" Target="metadata.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be metadata.

[*Example*: metadata.xml contains the following:

```
<metadata …>
  <metadataTypes count="1">
    <metadataType name="XLMDX" minSupportedVersion="120000" copy="1"
      pasteAll="1" pasteValues="1" merge="1" splitFirst="1" rowColShift="1"
      clearFormats="1" clearComments="1" assign="1" coerce="1"/>
  </metadataTypes>
  <metadataStrings count="2">
    <s v="externalData"/>
    <s v="[Account].[All Account]"/>
  </metadataStrings>
  <mdxMetadata count="1">
    <m n="0" f="m">
      <t c="1">
        <n v="1"/>
      </t>
    </m>
  </mdxMetadata>
```

```
    <valueMetadata count="1">
      <b>
        <r t="1" v="0"/>
      </b>
    </valueMetadata>
  </metadata>
```

The corresponding Connections part contains the following:

```
  <connections …>
    <connection id="1" odcFile="…" keepAlive="1" name="externalData"
      description="…" type="5" refreshedVersion="3" background="1">
      <dbPr connection="Provider=MSOLAP.2;…" command="Budget" commandType="1"/>
      <olapPr sendLocale="1" rowDrillCount="1000" serverFill="1"
        serverNumberFormat="1" serverFont="1" serverFontColor="1"/>
    </connection>
  </connections>
```

The corresponding Volatile Dependencies part contains the following:

```
  <volTypes …">
    <volType type="cubeFunctions">
      <main first="externalData">
        <tp t="e">
          <v>#N/A</v>
          <stp>1</stp>
          <r r="B3" s="1"/>
        </tp>
      </main>
    </volType>
  </volTypes>
```

The corresponding Pivot Table Cache Definition part contains the following:

```
  <pivotCacheDefinition … saveData="0" refreshedBy="…"
    refreshedDateIso="2005-11-28T16:55:44" backgroundQuery="1" createdVersion="3"
    refreshedVersion="3" recordCount="0">
    <cacheSource type="external" connectionID="1"/>
    <cacheFields count="0"/>
    <cacheHierarchies count="6">
      …
    </cacheHierarchies>
  <kpis count="0"/>
  <tupleCache>
    <queryCache count="3">
      <query mdx="[product].[category]"/>
      <query mdx=""/>
      <query mdx="[Account].[All Account]">
        <tpls c="1">
          <tpl hier="0" item="4294967295"/>
        </tpls>
      </query>
    </queryCache>
  </tupleCache>
</pivotCacheDefinition>
```

end example]

A Metadata part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Metadata part shall not have implicit or explicit relationships to any part defined by ISO/IEC 29500.

12.3.11 Pivot Table Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.pivotTable+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/pivotTable

An instance of this part type contains a pivot table definition.

A package shall contain exactly one Pivot Table part per pivot table, and each such part shall be the target of an implicit relationship in the relationship part for the Worksheet part (§12.3.24) that corresponds to the worksheet containing the pivot table.

[*Example*: The following Worksheet part-relationship item contains a relationship to two Pivot Table parts, which are stored in the ZIP items ../pivotTables/pivotTable*N*.xml:

```
  <Relationships xmlns="…">
    <Relationship Id="rId1"
      Type="http://…/pivotTable" Target="../pivotTables/pivotTable1.xml"/>
    <Relationship Id="rId2"
      Type="http://…/pivotTable" Target="../pivotTables/pivotTable2.xml"/>
  </Relationships>
```

end example]

The root element for a part of this content type shall be pivotTableDefinition.

[*Example*: pivotTable1.xml contains the following:

```
<pivotTableDefinition … cache="4" applyNumberFormats="0" applyBorderFormats="0"
  applyFontFormats="0" applyPatternFormats="0" applyAlignmentFormats="0"
  applyWidthHeightFormats="1" dataCaption="Data" updatedVersion="3"
  minRefreshableVersion="3" useAutoFormatting="1" itemPrintTitles="1"
  createdVersion="3" indent="0" outline="1" outlineData="1">
  <location ref="H4:H5" firstHeaderRow="1" firstDataRow="1" firstDataCol="0"/>
  <pivotFields count="1">
    <pivotField dataField="1" numFmtId="0" outline="1"
      subtotalTop="1" showAll="0" measureFilter="0" sortType="manual"/>
  </pivotFields>
  <rowItems count="1">
    <i t="data"/>
  </rowItems>
  <colItems count="1">
    <i t="data"/>
  </colItems>
  <dataFields count="1">
    <dataField name="Sum of 1000" fld="0" subtotal="average"
      baseField="0" baseItem="0" numFmtId="0"/>
  </dataFields>
  <tableStyle name="TableStyle2" showRowHeaders="1" showColHeaders="1"
    showRowStripes="1" showColStripes="1"/>
</pivotTableDefinition>
```

end example]

A Pivot Table part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Pivot Table part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Pivot Table Cache Definition (§12.3.12).

A Pivot Table part shall not have any implicit or explicit relationships to other parts defined by ISO/IEC 29500.

12.3.12 Pivot Table Cache Definition Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.pivotCacheDefinition+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/pivotCacheDefinition

An instance of this part type contains a cache definition for a pivot table.

A package shall contain exactly one Pivot Table Cache Definition part per pivot table, and each such part shall be the target of an implicit relationship from a Pivot Table (§12.3.11) part as well as an explicit relationship from a Workbook (§12.3.23) part.

[*Example*: The following Pivot Table part-relationship item contains a relationship to the Pivot Table Cache Definition part, which is stored in the ZIP item ../pivotCache/pivotCacheDefinition2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type=http://…/pivotCacheDefinition
    Target="../pivotCache/pivotCacheDefinition2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be pivotCacheDefinition.

[*Example*: pivotCacheDefinition2.xml contains the following:

```
<pivotCacheDefinition … r:id="rId1" refreshedBy="John Jones"
  refreshedDateIso="2005-11-18T16:47:49" createdVersion="3"
  refreshedVersion="3" recordCount="11">
  <cacheSource type="worksheet">
    <worksheet range="C4:C15" sheet="Sheet1"/>
  </cacheSource>
  <cacheFields count="1">
    <cacheField name="1000">
      <sharedItems containsSemiMixedTypes="0" containsString="0"
        containsNumber="1" containsInteger="1" minValue="234
        maxValue="2543"/>
    </cacheField>
  </cacheFields>
</pivotCacheDefinition>
```

end example]

A Pivot Table Cache Definition part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Pivot Table Cache Definition part is permitted to have an explicit relationship to the following part defined by ISO/IEC 29500:

- Pivot Table Cache Records (§12.3.13).

A Pivot Table Cache Definition part shall not have any implicit or explicit relationships to other parts defined by ISO/IEC 29500.

12.3.13 Pivot Table Cache Records Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.pivotCacheRecords+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/pivotCacheRecords

An instance of this part type contains the cache records for a pivot table.

A package shall contain zero or one Pivot Table Cache Records part per pivot table, and each such part shall be the target of an explicit relationship in the Pivot Table Cache Definition (§12.3.12) part for the corresponding pivot table.

[*Example*: The following Pivot Table Cache Definition part-relationship item contains a relationship to the Pivot Table Cache Records part, which is stored in the ZIP item pivotCacheRecords2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/pivotCacheRecords" Target="pivotCacheRecords2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be pivotCacheRecords.

[*Example*: pivotCacheRecords2.xml contains the following:

```
<pivotCacheRecords … count="11">
  <r>
    <n v="1234"/>
  </r>
    …
  <r>
    <n v="876"/>
  </r>
</pivotCacheRecords>
```

end example]

A Pivot Table Cache Records part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Pivot Table Cache Records part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.14 Query Table Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.queryTable+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/queryTable

An instance of this part type contains information that describes how the source table is connected to an external data source, and defines the properties that is used when this table is refreshed from that source.

A package is permitted to contain one Query Table part per table, and each of those parts shall be the target of an implicit relationship from the corresponding Table Definitions (§12.3.21) part.

[*Example*: The following Table part-relationship item contains a relationship to the Query Table part corresponding to the connections details for that table. These parts are stored in the ZIP items ../queryTables/queryTable*n*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/queryTable"
    Target="../queryTables/queryTable1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be queryTable.

[*Example*: queryTable2.xml deals with a connection to a database file having the seven fields shown:

```
<queryTable … name="+Connect to New Data Source_1"
  growShrinkType="insertDelete" connectionId="2" autoFormatId="16"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="1"
  applyPatternFormats="1" applyAlignmentFormats="0"
  applyWidthHeightFormats="0">
```

©ISO/IEC 2012 – All rights reserved

```
  <queryTableRefresh nextId="8">
    <queryTableFields count="7">
      <queryTableField id="1" name="ACCOUNT"/>
      <queryTableField id="2" name="CHECKNUM"/>
      <queryTableField id="3" name="DATE"/>
      <queryTableField id="4" name="AMOUNT"/>
      <queryTableField id="5" name="PAYEE"/>
      <queryTableField id="6" name="CHARGECODE"/>
      <queryTableField id="7" name="DESCRIPT"/>
    </queryTableFields>
  </queryTableRefresh>
</queryTable>
```

end example]

A Query Table part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Query Table part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.15 Shared String Table Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.sharedStrings+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/sharedStrings

An instance of this part type contains one occurrence of each unique string that occurs on all worksheets in a workbook.

A package shall contain exactly one Shared String Table part, and that part shall be the target of an implicit relationship from the Workbook part (§12.3.23).

[*Example*: The following Workbook part-relationship item contains a relationship to the Shared String Table part, which is stored in the ZIP item sharedStrings.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId6"
    Type="http://…/sharedStrings" Target="sharedStrings.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be sst.

[*Example*: Here are three of the six strings used in the worksheets:

```
<sst xmlns:st="…" … totalCount="6" uniqueCount="6">
  <sstItem>
    <t>Expenses Log</t>
  </sstItem>
  <sstItem>
    <t>Period Start</t>
  </sstItem>
  <sstItem>
    <t>Period End</t>
  </sstItem>
  …
</sst>
```

end example]

A Shared String Table part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Shared String Table part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.16 Shared Workbook Revision Headers Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.revisionHeaders+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/revisionHeaders

An instance of this part type contains information about each of the editing sessions performed on the parent workbook at the worksheet level (worksheets added and rearranged in each session).

A package shall contain at most one Shared Workbook Revision Headers part. If it exists, that part shall be the target of an implicit relationship from the Workbook (§12.3.23) part.

[*Example*: The following Workbook part-relationship item contains a relationship to the Shared Workbook Revision Headers part, which is stored in the ZIP item handout revisions/revisionHeaders.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId9"
    Type="http://…/revisionHeaders"
    Target="revisions/revisionHeaders.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be headers.

[*Example*: revisionHeaders.xml contains the following:

```
<headers xmlns:r="…" guid="{233BEE23-EB5C-4542-905D-0230EFFED88B}"
  diskRevisions="1" revisionId="4" version="3">
  <header guid="…" dateTime="…" maxSheetId="4" userName="…" r:id="rId1">
    <sheetIdMap count="3">
      …
    </sheetIdMap>
  </header>
  …
  <header guid="…" dateTime="…" maxSheetId="4" userName="…" r:id="rId3">
    <sheetIdMap count="3">
      …
    </sheetIdMap>
  </header>
</headers>
```

end example]

A Shared Workbook Revision Headers part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Shared Workbook Revision Headers part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Shared Workbook Revision Log (§12.3.17)

A Shared Workbook Revision Headers part shall not have any implicit or explicit relationships to other parts defined by ISO/IEC 29500.

12.3.17 Shared Workbook Revision Log Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.revisionLog+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/revisionLog

An instance of this part type contains information about edits performed on individual cells in the parent workbook's worksheets in each editing session.

A package shall contain one Shared Workbook Revision Log part for each session's set of changes, and those parts shall be the target of an explicit relationship from the Shared Workbook Revision Headers (§12.3.16) part.

[*Example*: The following Shared Workbook Revision Headers part-relationship item contains a number of relationships to Shared Workbook Revision Log parts, which are stored in the ZIP item revisionLog*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/revisionLog" Target="revisionLog1.xml"/>
  …
  <Relationship Id="rId6"
    Type="http://…/revisionLog" Target="revisionLog6.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be revisions.

[*Example*: revisionLog2.xml contains the following:

```
<revisions xmlns:xs="…" …>
  <rfmt sheetId="1" sqref="B4:B15">
    <dxf>
      <xs:fill>
        <xs:pattern patternType="solid">
          <xs:fgColor type="icv" val="64"/>
          <xs:bgColor type="rgb" val="4278252287"/>
        </xs:pattern>
      </xs:fill>
    </dxf>
  </rfmt>
  <rcv guid="{CBCE5672-5A4D-48C9-A120-F72804F8CF64}" action="delete"/>
  <rcv guid="{CBCE5672-5A4D-48C9-A120-F72804F8CF64}" action="add"/>
</revisions>
```

end example]

A Shared Workbook Revision Log part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Shared Workbook Revision Log part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.18 Shared Workbook User Data Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.userNames+xml

Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/usernames

An instance of this part type contains a list of all the users that are sharing the parent workbook.

A package shall contain at most one Shared Workbook User Data part, and that part shall be the target of an implicit relationship in the Workbook (§12.3.23) part.

[*Example*: The following Workbook part-relationship item contains a relationship to the Shared Workbook User Data part, which is stored in the ZIP item revisions/userNames.xml:

```
<Relationships xmlns="…">
  Relationship Id="rId8"
    Type="http://…/usernames" Target="revisions/userNames.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be users.

[*Example*: userNames.xml shows that there are two users sharing this workbook:

```
<users … count="2">
  <usrinfo guid="{B5A024F7-40BE-4A48-9B6D-B1655241C84D}"
    name="Mary Jones" id="-264292310" dateTime="2005-11-18T18:53:16"/>
  <usrinfo …/>
</users>
```

end example]

A Shared Workbook User Data part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Shared Workbook User Data part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.19 Single Cell Table Definitions Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.tableSingleCells+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/tableSingleCells

An instance of this part type contains information on how to map non-repeating elements from a custom XML file into cells in a worksheet. [*Note*: Repeating custom XML elements are mapped using a Table (§12.3.21). *end note*]

A package shall contain at most one Single Cell Table Definitions part per worksheet, and that part shall be the target of an implicit relationship from a Worksheet (§12.3.24) part. A Single Cell Table Definitions part can describe one or more single cell table definitions for any given worksheet.

[*Example*: The following Worksheet part-relationship item contains a relationship to the Single Cell Table Definitions part, which is stored in the ZIP item ../tables/tableSingleCells1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/tableSingleCells"
    Target="../tables/tableSingleCells1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be singleXmlCells.

[*Example*: A worksheet contains two single cell table definitions; e.g., ../tables/tableSingleCells1.xml contains the following, where the elements id and count are nested inside element names:

```
<singleXmlCells …>
  <singleCell id="1" name="Table1" displayName="Table1" ref="B4">
    <cellPr id="1" uniqueName="id">
      <xmlPr mapId="1" xpath="/names/id" xmlDataType="string"/>
    </cellPr>
  </singleCell>
  <singleCell id="2" name="Table2" displayName="Table2" ref="B7">
    <cellPr id="1" uniqueName="count">
      <xmlPr mapId="1" xpath="/names/count" xmlDataType="integer"/>
    </cellPr>
  </singleCell>
</singleXmlCells>
```

end example]

A Single Cell Table Definitions part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Single Cell Table Definitions part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

©ISO/IEC 2012 – All rights reserved

12.3.20 Styles Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.styles+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/styles

An instance of this part type contains all the characteristics for all the cells in the workbook. Such information includes numeric and text formatting, alignment, font, color, and border.

A package shall contain no more than one Styles part, and that part shall be the target of an implicit relationship from the Workbook (§12.3.23) part.

[*Example*: The following Workbook part-relationship item contains a relationship to the Styles part, which is stored in the ZIP item styles.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/styles" Target="styles.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be styleSheet.

[*Example*:

```
<styleSheet xmlns="…">
  <numFmts count="5">
    <numFmt numFmtId="164" formatCode="&quot;$&quot;#,##0.00"/>
    <numFmt numFmtId="165"
      formatCode="&quot;Yes&quot;;&quot;Yes&quot;;&quot;No&quot;"/>
    <numFmt numFmtId="166"
      formatCode="&quot;True&quot;;&quot;True&quot;;&quot;False&quot;"/>
    <numFmt numFmtId="167"
      formatCode="&quot;On&quot;;&quot;On&quot;;&quot;Off&quot;"/>
    <numFmt numFmtId="168"
      formatCode="[$€-2]\ #,##0.00_);[Red]\([$€-2]\ #,##0.00\)"/>
  </numFmts>
  <fonts count="5">
    …
  </fonts>
```

```
    <fills count="4">
      …
    </fills>
    <borders count="1">
      …
    </borders>
    …
    <colors>
      …
    </colors>
  </styleSheet>
```

end example]

A Styles part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Styles part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.21 Table Definition Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.table+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/table

An instance of this part type contains a description of a single table and its autofilter information. (The data for the table is stored in the corresponding Worksheet part.)

A package shall contain one Table Definition part per table, and each such part shall be the target of an implicit relationship from the Worksheet (§12.3.24) part that corresponds to the worksheet containing that table.

[*Example*: The following Worksheet part-relationship item contains relationships to two Table Definition parts, which are stored in the ZIP items ../tables/table*N*.xml:

```
  <Relationships xmlns="…">
    <Relationship Id="rId2"
      Type="http://…/table" Target="../tables/table1.xml"/>
    <Relationship Id="rId3"
      Type="http://…/table" Target="../tables/table2.xml"/>
  </Relationships>
```

end example]

©ISO/IEC 2012 – All rights reserved

The root element for a part of this content type shall be table.

[*Example*: table2.xml describes a table that spans a 2-column range of cells, F2:G19:

```
<table xmlns:af="…" … id="2" name="Table2" displayName="Table2" ref="F2:G19"
  totalsRowShown="0" headerRowDxfId="7">
  <autoFilter ref="F2:G19"/>
  <tableColumns count="2">
    <tableColumn id="1" name="Salesman" dataDxfId="9" totalsRowDxfId="6"/>
    <tableColumn id="2" name="Units" dataDxfId="8" totalsRowDxfId="5"/>
  </tableColumns>
  <tableStyle name="TableStyle2" showFirstColumn="0" showLastColumn="0"
    showRowStripes="1" showColumnStripes="1"/>
</table>
```

When the filter "Salesman equal to Smith" is applied, the autoFilter element in table2.xml is extended, as follows:

```
<autoFilter ref="F2:G19">
  <af:filterColumn colId="0">
    <af:filters>
      <af:filter val="Smith"/>
    </af:filters>
  </af:filterColumn>
</autoFilter>
```

end example]

A Table Definition part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be Internal).

A Table Definition part is permitted to explicit relationships to the following parts defined by ISO/IEC 29500:

- Query Table (§12.3.14)

A Table Definition part shall not have any implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.22 Volatile Dependencies Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.volatileDependencies+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/volatileDependencies

©ISO/IEC 2012 – All rights reserved

An instance of this part type contains information involving real-time data formulas in a workbook. Real-time data formulas return values that change over time — often every few seconds — and require connectivity to programs outside of the workbook to retrieve their data. In cases where those programs are not available, real-time data formulas can use the information stored in the Volatile Dependencies part to calculate results, rather than generate errors. More information on real-time data functions can be found in §18.17.7.284 and §18.17.7.65 through §18.17.7.71.

A package shall contain exactly one Volatile Dependencies part, and that part shall be the target of an implicit relationship from the Workbook (§12.3.23) part.

[*Example*: The following Workbook part-relationship item contains a relationship to the Volatile Dependencies part, which is stored in the ZIP item volatileDependencies.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId8"
    Type="http://…/volatileDependencies"
    Target="volatileDependencies.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be volTypes.

[*Example*:

```
<volTypes xmlns="…"/>
```

end example]

A Volatile Dependencies part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Volatile Dependencies part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.23 Workbook Part

Content Type(s):	application/vnd.openxmlformats-officedocument.spreadsheetml.sheet.main+xml application/vnd.openxmlformats-officedocument.spreadsheetml.template.main+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/officeDocument

An instance of this part type contains workbook data and references to all of its worksheets.

©ISO/IEC 2012 – All rights reserved

A package shall contain exactly one Workbook part, and that part shall be the target of a relationship in the package-relationship item.

[*Example*: The following SpreadsheetML package-relationship item contains a relationship to the Workbook part, which is stored in the ZIP item xl/workbook.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/officeDocument" Target="xl/workbook.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be workbook.

[*Example*: This workbook has three worksheets, named January, February, and March:

```
<workbook xmlns="…" xmlns:r="…">
  <fileVersion lastEdited="4" lowestEdited="4" rupBuild="3417"/>
  <bookViews>
    <workbookView xWindow="240" yWindow="15" windowWidth="8505"
      windowHeight="6240"/>
  </bookViews>
  <sheets>
    <sheet name="January" sheetId="1" r:id="rId1"/>
    <sheet name="February" sheetId="2" r:id="rId2"/>
    <sheet name="March" sheetId="3" r:id="rId3"/>
  </sheets>
  <workbookPr showObjects="all"/>
  <webPublishing codePage="1252"/>
  <calcPr calcId="122211" fullCalcOnLoad="1"/>
</workbook>
```

end example]

A Workbook part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Workbook part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Calculation Chain (§12.3.1)
- Cell Metadata (§12.3.10)
- Connections (§12.3.4)
- Custom XML Mappings (§12.3.6)

- Custom XML Data Storage (§15.2.4)
- Shared String Table (§12.3.15)
- Shared Workbook Revision Headers (§12.3.16)
- Shared Workbook User Data (§12.3.18)
- Styles (§12.3.20)
- Theme (§14.2.7)
- Thumbnail (§15.2.16)
- Volatile Dependencies (§12.3.22)

A Workbook part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Chartsheet (§12.3.2)
- Dialogsheet (§12.3.7)
- External Workbook References (§12.3.8)
- Pivot Table Cache Definition (§12.3.12)
- Worksheet (§12.3.24)

A Workbook part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.3.24 Worksheet Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.worksheet+xml
Root Namespace:	http://purl.oclc.org/ooxml/spreadsheetml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/worksheet

An instance of this part type contains all the data, formulas, and characteristics associated with a given worksheet.

A package shall contain exactly one Worksheet part per worksheet, and those parts shall be the target of an explicit relationship from the Workbook (§12.3.23) part. Specifically, the id attribute on the sheet element shall reference the desired worksheet part.

[*Example*: The following Workbook part-relationship item contains three relationships to Worksheet parts, which are stored in the ZIP items worksheets/sheet*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/worksheet" Target="worksheets/sheet1.xml"/>
  <Relationship Id="rId2"
    Type="http://…/worksheet" Target="worksheets/sheet2.xml"/>
  <Relationship Id="rId3"
    Type="http://…/worksheet" Target="worksheets/sheet3.xml"/>
 </Relationships>
```

end example]

The root element for a part of this content type shall be worksheet.

[*Example*: This worksheet, has cells in the range B1 to F8. Row 8 contains three cells whose values are calculated using the following formulas: D8=SUM(D5:D7), E8=SUM(E5:E7), and F8= D8-E8:

```
<worksheet xmlns="…" …>
  <sheetPr/>
  <dimension range="B1:F8"/>
  …
  <sheetData>
    <row r="1" spans="2:6" ht="360">
      <c r="B1" s="1" t="s">
        <v>0</v>
      </c>
    </row>
    …
    <row r="8" spans="2:6" ht="360">
      <c r="D8" s="5">
        <f>SUM(D5:D7)</f>
        <v>2280.5299999999997</v>
      </c>
      <c r="E8" s="5">
        <f>SUM(E5:E7)</f>
        <v>1251.31</v>
      </c>
      <c r="F8" s="6">
        <f>D8-E8</f>
        <v>1029.2199999999998</v>
      </c>
    </row>
  </sheetData>
  …
</worksheet>
```

end example]

A Worksheet part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Worksheet part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Comments (§12.3.3)
- Pivot Table Definitions (§12.3.11)
- Printer Settings (§15.2.15)
- Query Table Part (§12.3.14)
- Single Cell Table Definitions (§12.3.19)
- Table Definition (§12.3.21)

A Worksheet part is permitted to contain explicit relationships to the following parts defined by ISO/IEC 29500:

- Drawings (§12.3.8)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlinks (§15.3)
- Images (§15.2.14)

A Worksheet part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

12.4 External Workbooks

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/externalLinkPath

An *external workbook* is a SpreadsheetML package whose contents are being referenced by the current SpreadsheetML package. When a package refers to external workbooks, it shall store the location of those workbooks using this relationship.

A package is permitted to contain one or more External Workbook relationships, and those relationships shall be an explicit relationship from the External Workbook References (§12.3.9) part.

[*Example*: An External Workbook References part, which references the package c:\sourceData.xlsx would have an External Workbook relationship, which points at that file:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/externalLinkPath"
    Target="c:\sourceData.xlsx" TargetMode="External"/>
</Relationships>
```

©ISO/IEC 2012 – All rights reserved

end example]

A external workbook shall be located external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

13. PresentationML

This clause contains specifications for relationship items and parts that are specific to PresentationML. Parts than can occur in a PresentationML document, but are not PresentationML-specific, are specified in §15.2. Unless stated explicitly, all references to relationship items, content-type items, and parts in this clause refer to PresentationML ZIP items.

13.1 Glossary of PresentationML-Specific Terms

The following terms are used in the context of a PresentationML document:

handout — A printed set of slides that can be handed out to an audience for future reference.

note — A slide annotation, reminder, or piece of text intended for the presenter or the audience.

presentation — A collection of slides intended to be viewed by an audience.

slide — A frame containing one or more pieces of text and/or images.

slide layout — The organization of elements on a slide.

13.2 Package Structure

A PresentationML package shall contain a package-relationship item and a content-type item. The package-relationship item shall have implicit relationships with targets of the following type:

- One Presentation part (§13.3.6).

The package-relationship item is permitted to have implicit relationships with targets of the following type:

- Digital Signature Origin (§15.2.7)
- File Property parts (§15.2.12) (Application-Defined File Properties, Core File Properties, and Custom File Properties), as appropriate.
- Thumbnail (§15.2.16).

The required and optional relationships between parts are defined in §13.3 and its subordinate clauses.

[*Example*: The following package represents the minimal conformant PresentationML package as defined by ISO/IEC 29500:

First, the content type for relationship parts and the Main Presentation part (the only required part) must be defined (physically located at /[Content_Types].xml in the package):

©ISO/IEC 2012 – All rights reserved

```
<Types xmlns="…">
  <Default Extension="rels"
    ContentType="application/vnd.openxmlformats-
      package.relationships+xml"/>
  <Override PartName="/presentation.xml"
    ContentType="application/vnd.openxmlformats-
      officedocument.presentationml.presentation.main+xml"/>
</Types>
```

Next, the single required relationship (the package-level relationship to the Main Presentation part) must be defined (physically located at /_rels/.rels in the package):

```
<Relationships xmlns="…">
  <Relationship Id="rId1"

Type="http://purl.oclc.org/ooxml/officeDocument/relationships/officeDocument"
    Target="presentation.xml"/>
</Relationships>
```

Finally, the minimum content for the Main Presentation part must be defined (physically located at /presentation.xml in the package):

```
<p:presentation xmlns:p="…">
  <p:notesSz cx="913607" cy="913607"/>
</p:presentation>
```

end example]

[*Example*: Consider a simple PresentationML document containing two slides, which both use an image as a template. Here's an example of the hierarchical folder structure that might be used for the ZIP items in the package for that document. As shown, a number of parts have their own relationship items:

```
/_rels/.rels                            Package-relationship item
/[Content_Types].xml                    Content-type item
/docProps/app.xml                       Application-Defined File Properties
part
/docProps/core.xml                      Core File Properties part
/docProps/custom.xml                    Custom File Properties part
/docProps/thumbnail.wmf                 Package thumbnail image
/ppt/presentation.xml                   Presentation part
/ppt/_rels/presentation.xml.rels        Part-relationship item
/ppt/presProps.xml                      Presentation Properties part
/ppt/tableStyles.xml                    Table Styles part
/ppt/viewProps.xml                      View Properties part
```

```
/ppt/handoutMasters/handoutMaster1.xml       Handout Master part
/ppt/handoutMasters/_rels/handoutMaster1.xml.rels
                                             Part-relationship item
/ppt/media/image1.jpeg                       Slide template image
/ppt/notesMasters/notesMaster1.xml           Notes Master part
/ppt/notesMasters/_rels/notesMaster1.xml.rels
                                             Part-relationship item
/ppt/notesSlides/notesSlide1.xml             Notes Slide parts
/ppt/notesSlides/notesSlide2.xml
/ppt/notesSlides/_rels/notesSlide1.xml.rels
                                             Part-relationship items
/ppt/notesSlides/_rels/notesSlide2.xml.rels
/ppt/slideLayouts/slideLayout1.xml           Slide Layout parts 1-6
/ppt/slideLayouts/slideLayout2.xml
/ppt/slideLayouts/slideLayout3.xml
/ppt/slideLayouts/slideLayout4.xml
/ppt/slideLayouts/slideLayout5.xml
/ppt/slideLayouts/slideLayout6.xml
/ppt/slideLayouts/_rels/slideLayout1.xml.rels
                                             Part-relationship items
/ppt/slideLayouts/_rels/slideLayout2.xml.rels
/ppt/slideLayouts/_rels/slideLayout3.xml.rels
/ppt/slideLayouts/_rels/slideLayout4.xml.rels
/ppt/slideLayouts/_rels/slideLayout5.xml.rels
/ppt/slideLayouts/_rels/slideLayout6.xml.rels
/ppt/slideMasters/slideMaster1.xml           Slide Master part
/ppt/slideMasters/_rels/slideMaster1.xml.rels
                                             Part-relationship item
/ppt/slides/slide1.xml                       Slide parts
/ppt/slides/slide2.xml
/ppt/slides/_rels/slide1.xml.rels            Part-relationship items
/ppt/slides/_rels/slide2.xml.rels
/ppt/theme/theme1.xml                        Theme parts
/ppt/theme/theme2.xml
/ppt/theme/theme3.xml
```

©ISO/IEC 2012 – All rights reserved

```
/ppt/theme/themeOverride1.xml                Theme Override parts
/ppt/theme/themeOverride2.xml
/ppt/theme/themeOverride3.xml
/ppt/theme/themeOverride4.xml
/ppt/theme/themeOverride5.xml
/ppt/theme/themeOverride6.xml
/ppt/theme/themeOverride7.xml
/ppt/theme/themeOverride8.xml
/ppt/theme/themeOverride9.xml
/ppt/theme/themeOverride10.xml
```

The package-relationship item contains the following:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/officeDocument" Target="ppt/presentation.xml"/>
  <Relationship Id="rId3"
    Type="http://…/core-properties" Target="docProps/core.xml"/>
  <Relationship Id="rId2"
    Type="http://…/thumbnail" Target="docProps/thumbnail.wmf"/>
  <Relationship Id="rId4"
    Type="http://…/extended-properties" Target="docProps/app.xml"/>
</Relationships>
```

end example]

13.3 Part Summary

The subclauses subordinate to this one describe in detail each of the part types specific to PresentationML.

[*Note*: For convenience, information from those subclauses is summarized in the following table:

Part	Relationship Target of	Root Element	Ref.
Comment Authors	Presentation	cmAuthorLst	§13.3.1
Comments	Slide	cmLst	§13.3.2
Handout Master	Presentation	handoutMaster	§13.3.3
Notes Master	Notes Slide, Presentation	notesMaster	§13.3.4
Notes Slide	Slide	notes	§13.3.5
Presentation	PresentationML package	presentation	§13.3.6
Presentation Properties	Presentation	presentationPr	§13.3.7
Slide	Presentation	sld	§13.3.8
Slide Layout	Slide Master, Notes Slide,	sldLayout	§13.3.9

Part	Relationship Target of	Root Element	Ref.
	Presentation, Slide, Slide Master		
Slide Master	Presentation, Slide Layout	sldMaster	§13.3.10
Slide Synchronization Data	Slide	sldSyncPr	§13.3.11
User-Defined Tags	Presentation, Slide	tagLst	§13.3.12
View Properties	Presentation	viewPr	§13.3.13

end note]

13.3.1 Comment Authors Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.commentAuthors+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/commentAuthors

An instance of this part type contains information about each author who has added a comment to the document. That information includes the author's name, initials, a unique author-ID, a last-comment-index-used count, and a display color. (The color can be used when displaying comments to distinguish comments from different authors.)

A package shall contain at most one Comment Authors part. If it exists, that part shall be the target of an implicit relationship from the Presentation (§13.3.6) part.

[*Example*: The following Presentation part relationship item contains a relationship to the Comment Authors part, which is stored in the ZIP item commentAuthors.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId8"
    Type="http://…/commentAuthors" Target="commentAuthors.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be cmAuthorLst.

[*Example*: Two people have authored comments in this document: Mary Smith and Peter Jones. Her initials are "mas", her author-ID is 0, and her comments' display color index is 0. Since Mary's last-comment-index-used value is 3, the next comment-index to be used for her is 4. His initials are "pjj", his author-ID is 1, and his comments' display color index is 1. Since Peter's last-comment-index-used value is 1, the next comment-index to be used for him is 2:

```
<p:cmAuthorLst xmlns:p="…" …>
  <p:cmAuthor id="0" name="Mary Smith" initials="mas" lastIdx="3"
    clrIdx="0"/>
  <p:cmAuthor id="1" name="Peter Jones" initials="pjj" lastIdx="1"
    clrIdx="1"/>
</p:cmAuthorLst>
```

end example]

A Comment Authors part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Comment Authors part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.3.2 Comments Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.comments+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/comments

An instance of this part type contains the comments for a single slide. Each comment is tied to its author via an author-ID. Each comment's index number and author-ID combination are unique.

A package shall contain one Comments part for each slide containing one or more comments, and each of those parts shall be the target of an implicit relationship from its corresponding Slide (§13.3.8) part.

[*Example*: The following Slide part-relationship item contains a relationship to a Comments part, which is stored in the ZIP item ../comments/comment2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/comments"
    Target="../comments/comment2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be cmLst .

[*Example*: The Comments part contains three comments, two created by one author, and one created by another, all at the dates and times shown. The index numbers are assigned on a per-author basis, starting at 1 for an author's first comment:

```
<p:cmLst xmlns:p="…" …>
  <p:cm authorId="0" dt="2005-11-13T17:00:22.071" idx="1">
    <p:pos x="4486" y="1342"/>
    <p:text>Comment text goes here.</p:text>
  </p:cm>
  <p:cm authorId="0" dt="2005-11-13T17:00:34.849" idx="2">
    <p:pos x="3607" y="1867"/>
    <p:text>Another comment's text goes here.</p:text>
  </p:cm>
  <p:cm authorId="1" dt="2005-11-15T00:06:46.919" idx="1">
    <p:pos x="1493" y="2927"/>
    <p:text>comment …</p:text>
  </p:cm>
</p:cmLst>
```

end example]

A Comments part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Comments part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.3.3 Handout Master Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.handoutMaster+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/handoutMaster

An instance of this part type contains the look, position, and size of the slides, notes, header and footer text, date, or page number on the presentation's handout.

A package shall contain at most one Handout Master part, and it shall be the target of an explicit relationship from the Presentation (§13.3.6) part.

[*Example*: The following Presentation part-relationship item contains a relationship to the Handout Master part, which is stored in the ZIP item handoutMasters/handoutMaster1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/handoutMaster"
    Target="handoutMasters/handoutMaster1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be handoutMaster.

[*Example*:

```
<p:handoutMaster xmlns:p="…">
  <p:cSld name="">
    …
  </p:cSld>
  <p:clrMap … />
</p:handoutMaster>
```

end example]

A Handout Master part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Handout Master part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)
- Theme (§14.2.7)
- Thumbnail (§15.2.16)

A Handout Master part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlink (§15.3)
- Image (§15.2.14)
- Video (§15.2.15)

A Handout Master part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.3.4 Notes Master Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.notesMaster+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/notesMaster

An instance of this part type contains information about the content and formatting of all notes pages.

A package shall contain at most one Notes Master part, and that part shall be the target of an implicit relationship from the Notes Slide (§13.3.5) part, as well as an explicit relationship from the Presentation (§13.3.6) part.

[*Example*: The following Presentation part-relationship item contains a relationship to the Notes Master part, which is stored in the ZIP item notesMasters/notesMaster1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/notesMaster" Target="notesMasters/notesMaster1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be notesMaster.

[*Example*:

```
<p:notesMaster xmlns:p="…">
  <p:cSld name="">
    …
  </p:cSld>
  <p:clrMap … />
</p:notesMaster>
```

end example]

A Notes Master part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be Internal).

A Notes Master part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)

©ISO/IEC 2012 – All rights reserved

- Theme (§14.2.7)
- Thumbnail (§15.2.16)

A Notes Master part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlink (§15.3).
- Image (§15.2.14)
- Video (§15.2.15)

The Notes Master part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.3.5 Notes Slide Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.notesSlide+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main ain
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/notesSlide

An instance of this part type contains the notes for a single slide.

A package shall contain one Notes Slide part for each slide that contains notes. If they exist, those parts shall each be the target of an implicit relationship from the Slide (§13.3.8) part.

[*Example*: The following Slide part-relationship item contains a relationship to a Notes Slide part, which is stored in the ZIP item ../notesSlides/notesSlide1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId3"
    Type="http://…/notesSlide" Target="../notesSlides/notesSlide1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be notes.

[*Example*:

```
<p:notes xmlns:p="…">
  <p:cSld name="">
    …
  </p:cSld>
  <p:clrMapOvr>
    <a:masterClrMapping/>
  </p:clrMapOvr>
</p:notes>
```

end example]

A Notes Slide part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Notes Slide part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)
- Notes Master (§13.3.4)
- Theme Override(§14.2.8)
- Thumbnail (§15.2.16)

A Notes Slide part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlink (§15.3).
- Image (§15.2.14)
- Video (§15.2.15)

The Notes Slide part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.3.6 Presentation Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.presentation.main+xml application/vnd.openxmlformats-officedocument.presentationml.slideshow.main+xml

©ISO/IEC 2012 – All rights reserved

	application/vnd.openxmlformats-officedocument.presentationml.template.main+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/officeDocument

An instance of this part type contains the definition for a slide presentation.

A package shall contain exactly one Presentation part, and that part shall be the target of a relationship in the package-relationship item.

[*Example*: The following PresentationML's package-relationship item contains a relationship to the Presentation part, which is stored in the ZIP item ppt/presentation.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/officeDocument" Target="ppt/presentation.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be presentation.

[*Example*: This presentation contains two slides:

```
<p:presentation xmlns:p="…" … >
  <p:sldMasterIdLst>
    <p:sldMasterId
      xmlns:rel="http://…/relationships" rel:id="rId1"/>
  </p:sldMasterIdLst>
  <p:notesMasterIdLst>
    <p:notesMasterId
      xmlns:rel="http://…/relationships" rel:id="rId4"/>
  </p:notesMasterIdLst>
  <p:handoutMasterIdLst>
    <p:handoutMasterId
       xmlns:rel="http://…/relationships" rel:id="rId5"/>
  </p:handoutMasterIdLst>
  <p:sldIdLst>
    <p:sldId id="267"
      xmlns:rel="http://…/relationships" rel:id="rId2"/>
    <p:sldId id="256"
      xmlns:rel="http://…/relationships" rel:id="rId3"/>
  </p:sldIdLst>
```

```
    <p:sldSz cx="9144000" cy="6858000"/>
    <p:notesSz cx="6858000" cy="9144000"/>
  </p:presentation>
```

end example]

A Presentation part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Presentation part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Additional Characteristics (§15.2.1)
- Comment Authors (§13.3.1)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)
- Font (§15.2.13)
- Presentation Properties (§13.3.7)
- Table Styles (§14.2.9)
- Theme (§14.2.7)
- Thumbnail (§15.2.16)
- View Properties (§13.3.13).

A Presentation part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Notes Master (§13.3.4)
- Handout Master (§13.3.3)
- Slide (§13.3.8)
- Slide Master (§13.3.10)
- User Defined Tags (§13.3.12)

13.3.7 Presentation Properties Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.presProps+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/presProps

An instance of this part type contains all the presentation's properties.

A package shall contain exactly one Presentation Properties part, and that part shall be the target of an implicit relationship from the Presentation (§13.3.6) part.

[*Example*: The following Presentation part-relationship item contains a relationship to the Presentation Properties part, which is stored in the ZIP item presProps.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId6"
    Type="http://…/presProps" Target="presProps.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be presentationPr.

[*Example*:

```
<p:presentationPr xmlns:p="…" …>
  <p:clrMru>
    …
  </p:clrMru>
  …
</p:presentationPr>
```

end example]

A Presentation Properties part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Presentation Properties part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.3.8 Slide Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.slide+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/slide

A Slide part contains the contents of a single slide.

A package shall contain one Slide part per slide, and each of those parts shall be the target of an explicit relationship from the Presentation (§13.3.6) part.

[*Example*: Consider a PresentationML document having two slides. The corresponding Presentation part-relationship item contains two relationships to Slide parts, which are stored in the ZIP items slides/slide1.xml and slides/slide2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/slide" Target="slides/slide1.xml"/>
  <Relationship Id="rId3"
    Type="http://…/slide" Target="slides/slide2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be sld.

[*Example*: slides/slide1.xml contains:

```
<p:sld xmlns:p="…">
  <p:cSld name="">
    …
  </p:cSld>
  <p:clrMapOvr>
    …
  </p:clrMapOvr>
  <p:timing>
    <p:tnLst>
      <p:par>
        <p:cTn id="1" dur="indefinite" restart="never" nodeType="tmRoot"/>
      </p:par>
    </p:tnLst>
  </p:timing>
</p:sld>
```

end example]

A Slide part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Slide part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Comments (§13.3.2)
- Custom XML Data Storage (§15.2.4)
- Notes Slide (§13.3.5)
- Theme Override (§14.2.8)
- Thumbnail (§15.2.16)
- Slide Layout (§13.3.9)
- Slide Synchronization Data (§13.3.11)

©ISO/IEC 2012 – All rights reserved

A Slide part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlink (§15.3).
- Image (§15.2.14)
- User Defined Tags (§13.3.12)
- Video (§15.2.15)

A Slide part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.3.9 Slide Layout Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.slideLayout+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/slideLayout

An instance of this part type contains the definition for a slide layout template for this presentation. This template defines the default appearance and positioning of drawing objects on this slide type when it is created.

A package shall contain one or more Slide Layout parts, and each of those parts shall be the target of an explicit relationship in the Slide Master (§13.3.10) part, as well as an implicit relationship from each of the Slide (§13.3.8) parts associated with this slide layout.

[*Example*: The following Slide Master part-relationship item contains relationships to several Slide Layout parts, which are stored in the ZIP items ../slideLayouts/slideLayout*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/slideLayout"
    Target="../slideLayouts/slideLayout1.xml"/>
  <Relationship Id="rId2"
    Type="http://…/slideLayout"
    Target="../slideLayouts/slideLayout2.xml"/>
  <Relationship Id="rId3"
    Type="http://…/slideLayout"
    Target="../slideLayouts/slideLayout3.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be sldLayout.

[*Example*:

```
<p:sldLayout xmlns:p="…" matchingName="" type="title" preserve="1">
  <p:cSld name="Title Slide">
    …
  </p:cSld>
  <p:clrMapOvr>
    <a:masterClrMapping/>
  </p:clrMapOvr>
  <p:timing/>
  </p:sldLayout>
</p:sldMaster>
```

end example]

A Slide Layout part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)
- Slide Master (§13.3.10)
- Theme Override (§14.2.8)
- Thumbnail (§15.2.16)

A Slide Layout part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Content Part (§15.2.4)

- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlink (§15.3).
- Image (§15.2.14)
- Video (§15.2.15)

A Slide Layout part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.3.10 Slide Master Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.slideMaster+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/slideMaster

An instance of this part type contains the master definition of formatting, text, and objects that appear on each slide in the presentation that is derived from this slide master.

A package shall contain one or more Slide Master parts, each of which shall be the target of an explicit relationship from the Presentation (§13.3.6) part, as well as an implicit relationship from any Slide Layout (§13.3.9) part where that slide layout is defined based on this slide master. Each can optionally be the target of a relationship in a Slide Layout (§13.3.9) part as well.

[*Example*: The following Presentation part-relationship item contains a relationship to the Slide Master part, which is stored in the ZIP item slideMasters/slideMaster1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/slideMaster" Target="slideMasters/slideMaster1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be sldMaster.

[*Example*:

```
<p:sldMaster xmlns:p="…">
  <p:cSld name="">
    …
  </p:cSld>
  <p:clrMap … />
</p:sldMaster>
```

end example]

A Slide Master part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Slide Master part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Additional Characteristics (§15.2.1)
- Bibliography (§15.2.3)
- Custom XML Data Storage (§15.2.4)
- Theme (§14.2.7)
- Thumbnail (§15.2.16)

A Slide Master part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Audio (§15.2.2)
- Chart (§14.2.1)
- Content Part (§15.2.4)
- Diagrams: Diagram Colors(§14.2.3), Diagram Data(§14.2.4), Diagram Layout Definition(§14.2.5) and Diagram Styles (§14.2.6)
- Embedded Control Persistence (§15.2.9)
- Embedded Object (§15.2.10)
- Embedded Package (§15.2.11)
- Hyperlink (§15.3).
- Image (§15.2.14)
- Slide Layout (§13.3.9)
- Video (§15.2.15)

A Slide Master part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.3.11 Slide Synchronization Data Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.slideUpdateInfo+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/slideUpdateInfo

©ISO/IEC 2012 – All rights reserved

An instance of this part type contains properties specifying the current state of a slide that is being synchronized with a version of that slide stored on a central server.

A package shall contain zero or one Slide Synchronization Data part for each slide stored in the presentation, and that part shall be the target of an implicit relationship from the corresponding Slide (§13.3.8) part.

[*Example*: The following Slide part-relationship item contains a relationship to the Slide Synchronization Data part, which is stored in the ZIP item slideUpdateInfo/slideUpdateInfo1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1" Type="http://…/slideUpdateInfo"
    Target="slideUpdateInfo/slideUpdateInfo1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be sldSyncPr.

[*Example*:

```
<p:sldSyncPr xmlns:p="…" serverSldId="1"
  serverSldModifiedTime="2006-08-12T01:31:08"
  clientInsertedTime="2006-08-12T01:34:11.227" />
```

end example]

A Slide Synchronization Data part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Slide Synchronization Data part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Slide Synchronization Server Location (§13.4)

A Slide Synchronization Data part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.3.12 User Defined Tags Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.tags+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/tags

An instance of this part type contains a set of user-defined properties for an object in a presentation (each property consisting of a name/value pair).

A package shall contain zero or more User Defined Tags parts, each as the target of an explicit relationship from the corresponding Presentation (§13.3.6) or Slide (§13.3.8) part.

[*Example*: The following Slide part-relationship item contains a relationship to the User Defined Tags part, which is stored in the ZIP item tags/tag1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1" Type="http://…/tag"
    Target="tags/tag1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be tagLst.

[*Example*:

```
<p:tagLst xmlns:p="…" >
  <p:tag name="testTagName" val="testTagValue" />
</p:tagLst>
```

end example]

A User Defined Tags part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A User Defined Tags part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.3.13 View Properties Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.viewProps+xml
Root Namespace:	http://purl.oclc.org/ooxml/presentationml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/viewProps

An instance of this part type contains display properties for this presentation.

A package shall contain zero or one View Properties part, and if it exists, that part shall be the target of an implicit relationship from the Presentation (§13.3.6) part.

©ISO/IEC 2012 – All rights reserved

[*Example*: The following Presentation part-relationship item contains a relationship to the View Properties part, which is stored in the ZIP item viewProps.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId7"
    Type="http://…/viewProps" Target="viewProps.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be viewPr.

[*Example*:

```
<p:viewPr xmlns:p="…" …>
  <p:normalViewPr showOutlineIcons="0">
    …
  </p:normalViewPr>
  <p:slideViewPr>
    …
  </p:slideViewPr>
  <p:outlineViewPr>
    …
  </p:outlineViewPr>
  <p:notesTextViewPr>
    …
  </p:notesTextViewPr>
  <p:gridSpacing cx="78028800" cy="78028800"/>
</p:viewPr>
```

end example]

A View Properties part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A View Properties part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

13.4 HTML Publish Location

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/htmlPubSaveAs

When a presentation specifies an external location to which an optional copy might be pushed in the HTML format, this relationship shall be used to target the location where the HTML copy of the presentation is published.

A package shall contain one HTML Publish Location relationship for each slide linked with an HTML publish location, and that relationships shall be an explicit relationship from the corresponding Presentation Properties (§13.3.7) part.

[*Example*: A Presentation Properties part, which stores an HTML Publish Location of http://www.openxmlformats.org/test.htm contains the following relationship in that part's relationship part:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/htmlPubSaveAs"
    Target="http://www.openxmlformats.org/test.htm" type=”External”/>
</Relationships>
```

end example]

An HTML publish location shall be located external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

13.5 Slide Synchronization Server Location

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/slideUpdateUrl

When a slide is being synchronized with a copy stored on a remote server, this relationship shall be used to target the location where the server copy of the slide is stored.

A package shall contain one Slide Synchronization Server Location relationship for each slide linked with server data, and that relationships shall be an implicit relationship from the corresponding Slide Synchronization Data (§13.3.11) part.

[*Example*: A Slide Synchronization Data part that stores information about a slide that is synchronized with a server located at http://www.openxmlformats.org/slides/ contains the following relationship in that part's relationship part item:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/slideupdateUrl"
    Target="http://www.openxmlformats.org/slides/" type=”External”/>
</Relationships>
```

end example]

©ISO/IEC 2012 – All rights reserved

A slide synchronization server location shall be located external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `External`).

14. DrawingML

The relationship items and parts defined in this clause are used by one or more of WordprocessingML (§11), SpreadsheetML (§12), and PresentationML (§13) environments.

14.1 Glossary of DrawingML-Specific Terms

diagram — A picture or graphical representation that is displayed using a related set of color, data, layout, and style parts. Examples of diagram types are cycle, organization chart, pyramid, target, and Venn.

14.2 Part Summary

The subclauses subordinate to this one describe in detail each of the part types specific to DrawingML.

[*Note*: For convenience, information from those subclauses is summarized in the following table:

Part	Relationship Target of	Root Element	Ref.
Chart	WordprocessingML: Main Document SpreadsheetML: Drawings PresentationML: Handout Master, Notes Master, Notes Slide, Slide Layout, Slide Master, Slide All: Chart Drawing	chartSpace	§14.2.1
Chart Drawing	All: Chart	userShapes	§14.2.2
Diagram Colors	WordprocessingML: Main Document SpreadsheetML: Drawings PresentationML: Handout Master, Notes Master, Notes Slide, Slide Layout, Slide Master, Slide	colorsDef	§14.2.3
Diagram Data	WordprocessingML: Main Document SpreadsheetML: Drawings PresentationML: Handout Master, Notes Master, Notes Slide, Slide Layout, Slide Master, Slide	dataModel	§14.2.4
Diagram Layout Definition	WordprocessingML: Main Document SpreadsheetML: Drawings PresentationML: Handout Master, Notes Master, Notes Slide, Slide Layout, Slide Master, Slide	layoutDef	§14.2.5

©ISO/IEC 2012 – All rights reserved

Part	Relationship Target of	Root Element	Ref.
Diagram Style	WordprocessingML: Main Document SpreadsheetML: Drawings PresentationML: Handout Master, Notes Master, Notes Slide, Slide Layout, Slide Master, Slide	styleDef	§14.2.6
Theme	WordprocessingML: Main Document SpreadsheetML: Workbook PresentationML: Handout Master, Notes Master, Presentation, Slide Master	theme	§14.2.7
Theme Override	PresentationML: Notes Slide, Slide, Slide Layout	themeOverride	§14.2.8
Table Styles	PresentationML: Presentation	tblStyleLst	§14.2.9

end note]

14.2.1 Chart Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.chart+xml
Root Namespace:	http://purl.oclc.org/ooxml/drawingml/chart
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/chart

An instance of this part type describes a chart.

A package shall contain a Chart part for each chart in the document. In a WordprocessingML document, each such part shall be the target of an explicit relationship in a Main Document (§11.3.10) part. In a SpreadsheetML document, each such part shall be the target of an explicit relationship in a Drawings (§12.3.8) part. In a PresentationML document, each such part shall be the target of an explicit relationship in a Handout Master (§13.3.3), Notes Master (§13.3.4), Notes Slide (§13.3.5), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part. This part is permitted to also be the target of an explicit relationship in a Chart Drawing (§14.2.2) part, if the chart that points at this Chart Drawing part is the target of a relationship from a Chartsheet part. In other words, the only time a chart can embed another chart is if the parent chart is part of a chartsheet.

[*Example*: The following Main Document part-relationship item contains relationships to two Chart parts, which are stored in the ZIP items ../charts/chart*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/chart" Target="charts/chart1.xml"/>
  <Relationship Id="rId5"
    Type="http://…/chart" Target="charts/chart2.xml"/>
</Relationships>
```

The following Drawings part-relationship item contains a relationship to a Chart part, which is stored in the ZIP item ../charts/chart1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/relationships/chart" Target="../charts/chart1.xml"/>
</Relationships>
```

The following Slide part-relationship item contains relationships to two Chart parts, which are stored in the ZIP items ../charts/chart*N*.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/chart" Target="../charts/chart1.xml"/>
  <Relationship Id="rId5"
    Type="http://…/chart" Target="../charts/chart2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be chartSpace.

[*Example*: chart1.xml contains the following clustered bar chart:

```
<c:chartSpace …>
  <c:chart>
    <c:title>
      …
    </c:title>
    <c:plotArea>
      <c:layout>
        …
      </c:layout>
      <c:barChart>
        …
      </c:barChart>
    </c:plotArea>
```

```
          <c:legend>
        …
      </c:legend>
    </c:chart>
    …
  </c:chartSpace>
```

end example]

For WordprocessingML and PresentationML documents, the data for a chart is not stored in the Chart part directly. Instead, it shall be stored in an embedded SpreadsheetML package (§12.2) targeted by an Embedded Package (§15.2.11) part specified by that Chart part. For SpreadsheetML documents, the data for a chart is stored directly in the Drawing's parent worksheet; no embedded SpreadsheetML package shall be used.

A Chart part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Chart part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Chart Drawing (§14.2.2)
- Embedded Package (§15.2.11)

A Chart part shall not have any implicit or explicit relationships to any other part defined by ISO/IEC 29500.

14.2.2 Chart Drawing Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.chartshapes+xml
Root Namespace:	http://purl.oclc.org/ooxml/drawingml/chart
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/chartUserShapes

An instance of this part type contains all basic drawing elements (shapes) which are explicitly associated with this chart. These drawing elements are automatically moved with the chart when it is moved and resized when the chart is resized.

A package is permitted to contain one Chart Drawing part per chart part, and each such part shall be the target of an explicit relationship from a Chart (§14.2.1) part.

[*Example*: The following Chart part-relationship item contains a relationship to a Chart Drawing part, which is stored in the ZIP item ../drawings/drawing1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/chartUserShapes" Target="../drawings/drawing1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be userShapes.

[*Example*:

```
<c:userShapes xmlns:cdr="…" xmlns:c="…">
  <cdr:relSizeAnchor>
    …
  </cdr:relSizeAnchor>
</c:userShapes>
```

end example]

A Chart Drawing part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Chart Drawing part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Chart (§14.2.1)

A Chart Drawing part shall not have any implicit or explicit relationships to any other part defined by ISO/IEC 29500.

14.2.3 Diagram Colors Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.diagramColors+xml
Root Namespace:	http://purl.oclc.org/ooxml/drawingml/diagram
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/diagramColors

An instance of this part type contains color information for a diagram.

A package shall contain exactly one Diagram Colors part per diagram. Each Diagram Colors part shall be the target of an explicit relationship in a WordprocessingML Main Document (§11.3.10), SpreadsheetML Drawings (§12.3.8), or PresentationML Slide (§13.3.8) part.

[*Example*: The following SpreadsheetML Drawings part-relationship item contains a relationship to two Diagram Colors parts, which are stored in the ZIP items ../graphics/colors*N*.xml.

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
     Type="http://…/diagramColors" Target="../graphics/colors1.xml"/>
  <Relationship Id="rId8"
    Type="http://…/diagramColors" Target="../graphics/colors2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be colorsDef.

[*Example*: colors1.xml contains the following:

```
<cs:colorsDef xmlns:cs="…" uniqueId="…" minVer="12.0">
  <cs:title lang="" val="Primary Accent 2"/>
  <cs:desc lang="" val="Primary Accent 2"/>
  <cs:catLst>
    <cs:cat type="accent1" pri="11200"/>
  </cs:catLst>
  <cs:styleLbl …>
    …
  </cs:styleLbl>
  …
  <cs:styleLbl …>
    …
  </cs:styleLbl>
</cs:colorsDef>
```

end example]

A Diagram Colors part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Diagram Colors part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

14.2.4 Diagram Data Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.diagramData+xml
Root Namespace:	http://purl.oclc.org/ooxml/drawingml/diagram
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/diagramData

An instance of this part type contains the semantic data for a diagram.

A package shall contain exactly one Diagram Data part per diagram. Each Diagram Data part shall be the target of an explicit relationship in a WordprocessingML Main Document (§11.3.10); a SpreadsheetML Drawings part (§12.3.8); or a PresentationML Handout Master (§13.3.3), Notes Master (§13.3.4), Notes Slide (§13.3.5), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part.

[*Example*: The following SpreadsheetML Drawings part-relationship item contains a relationship to two Diagram Data parts, which are stored in the ZIP items ../graphics/data*N*.xml.

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/diagramData" Target="../graphics/data1.xml"/>
  <Relationship Id="rId5"
    Type="http://…/diagramData" Target="../graphics/data2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be dataModel.

[*Example*: data1.xml contains the following:

```
<dm:dataModel xmlns:dm="…">
  <dm:ptLst>
    …
  </dm:ptLst>
  <dm:cxnLst>
    …
  </dm:cxnLst>
  <dm:bg/>
  <dm:whole/>
</dm:dataModel>
```

end example]

A Diagram Data part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Diagram Data part is permitted to have explicit relationships to the following parts defined by ISO/IEC 29500:

- Image (§15.2.14)

A Diagram Data part shall not have any implicit or explicit relationships to other parts defined by ISO/IEC 29500.

14.2.5 Diagram Layout Definition Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.diagramLayout+xml
Root	http://purl.oclc.org/ooxml/drawingml/diagram

©ISO/IEC 2012 – All rights reserved

Namespace:	
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/diagramLayout

An instance of this part type is a template that describes how diagram-related data is mapped to a shape.

A package shall contain exactly one Diagram Layout Definition part per diagram. Each Layout Definition part shall be the target of an explicit relationship from a WordprocessingML Main Document (§11.3.10); a SpreadsheetML Drawings part (§12.3.8); or a PresentationML Handout Master (§13.3.3), Notes Master (§13.3.4), Notes Slide (§13.3.5), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part. If a document contains multiple diagrams having the same graphic layout definition, each of those diagrams shall have its own copy of that Diagram Layout Definition part.

[*Example*: The following SpreadsheetML Drawings part-relationship item contains a relationship to two Diagram Layout Definition parts, which are stored in the ZIP items ../graphics/layout*N*.xml.

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/diagramLayout" Target="../graphics/layout1.xml"/>
  <Relationship Id="rId6"
    Type="http://…/diagramLayout" Target="../graphics/layout2.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be layoutDef.

[*Example*: layout1.xml contains the following:

```
<lo:layoutDef xmlns:lo="…" uniqueId="…2" minVer="12.0" defStyle="">
  <lo:title lang="" val="Hierarchy 2"/>
  <lo:desc lang="" val=""/>
  <lo:catLst>
    <lo:cat type="hierarchy" pri="2000"/>
  </lo:catLst>
  <lo:sampData>
    …
  </lo:sampData>
  <lo:styleData>
    …
  </lo:styleData>
  <lo:clrData>
    …
  </lo:clrData>
```

```
    <lo:layoutNode name="Name0" styleLbl="" moveWith="">
      …
     </lo:layoutNode>
  </lo:layoutDef>
```

end example]

A Diagram Layout Definition part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Diagram Layout Definition part is permitted to have explicit relationships to the following parts and items defined by ISO/IEC 29500:

- Image (§15.2.14)

A Diagram Layout Definition part shall not have any implicit or explicit relationships to other parts defined by ISO/IEC 29500.

14.2.6 Diagram Style Part

Content Type:	application/vnd.openxmlformats-officedocument.drawingml.diagramStyle+xml
Root Namespace:	http://purl.oclc.org/ooxml/drawingml/diagram
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/diagramQuickStyle

An instance of this part type maps diagram semantic information to a document's theme.

A package shall contain exactly one Diagram Style part per diagram. Each Style part shall be the target of an explicit relationship from a WordprocessingML Main Document (§11.3.10); a SpreadsheetML Drawings part (§12.3.8); or a PresentationML Handout Master (§13.3.3), Notes Master (§13.3.4), Notes Slide (§13.3.5), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part.

[*Example*: The following SpreadsheetML Drawings part-relationship item contains a relationship to two Diagram Style parts, which are stored in the ZIP items ../graphics/quickStyle*N*.xml.

```
  <Relationships xmlns="…">
    <Relationship Id="rId3"
      Type="http://…/diagramQuickStyle"
      Target="../graphics/quickStyle1.xml"/>
    <Relationship Id="rId7"
      Type="http://…/diagramQuickStyle"
      Target="../graphics/quickStyle2.xml"/>
  </Relationships>
```

end example]

The root element for a part of this content type shall be styleDef.

[*Example*: quickStyle1.xml contains the following:

```
<qs:styleDef xmlns:qs="…" uniqueId="…" minVer="12.0">
  <qs:title lang="" val="Style 2"/>
  <qs:desc lang="" val="Style 2"/>
  <qs:catLst>
    <qs:cat type="simple" pri="10200"/>
  </qs:catLst>
  <qs:scene3d>
    …
  </qs:scene3d>
  <qs:style>
    …
  </qs:style>
  <qs:styleLbl name="…">
    …
  </qs:styleLbl>
  …
  <qs:styleLbl name="…">
    …
  </qs:styleLbl>
</qs:styleDef>
```

end example]

A Diagram Style part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Diagram Style part shall not have implicit or explicit relationships to other parts defined by ISO/IEC 29500.

14.2.7 Theme Part

Content Type:	application/vnd.openxmlformats-officedocument.theme+xml
Root Namespace:	http://purl.oclc.org/ooxml/drawingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/theme

An instance of this part type contains information about a document's *theme*, which is a combination of *color scheme*, *font scheme*, and *format scheme* (the latter also being referred to as *effects*). For a WordprocessingML document, the choice of theme affects the color and style of headings, among other things. For a

SpreadsheetML document, the choice of theme affects the color and style of cell contents and charts, among other things. For a PresentationML document, the choice of theme affects the formatting of slides, handouts, and notes via the associated master, among other things.

A WordprocessingML or SpreadsheetML package shall contain zero or one Theme part, which shall be the target of an implicit relationship in a Main Document (§11.3.10) or Workbook (§12.3.23) part. A PresentationML package shall contain zero or one Theme part per Handout Master (§13.3.3), Notes Master (§13.3.4), Slide Master (§13.3.10) or Presentation (§13.3.6) part via an implicit relationship.

[*Example*: The following WordprocessingML Main Document part-relationship item contains a relationship to the Theme part, which is stored in the ZIP item theme/theme1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/theme" Target="theme/theme1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be theme.

[*Example*: theme1.xml contains the following, where the name attributes of the clrScheme, fontScheme, and fmtScheme elements correspond to the document's color scheme, font scheme, and format scheme, respectively:

```
<a:theme xmlns:a="…">
  <a:themeElements>
    <a:clrScheme name="…">
      …
    </a:clrScheme>
    <a:fontScheme name="…">
      …
    </a:fontScheme>
    <a:fmtScheme name="…">
      …
    </a:fmtScheme>
  </a:themeElements>
  <a:objectDefaults/>
</a:theme>
```

end example]

A Theme part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be Internal).

A Theme part is permitted to contain explicit relationships to the following parts defined by ISO/IEC 29500:

- Image (§15.2.14)

A Theme part shall not have any implicit or explicit relationships to other parts defined by ISO/IEC 29500.

14.2.8 Theme Override Part

Content Type:	application/vnd.openxmlformats-officedocument.themeOverride+xml
Root Namespace:	http://purl.oclc.org/ooxml/drawingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/themeOverride

An instance of this part type contains information about an object's *theme override*, which are overrides to the *color scheme*, *font scheme*, and *format scheme* (the latter also being referred to as *effects*) for a particular slide, notes slide, or handout.

A PresentationML package shall contain zero or one Theme Override part per Notes Slide (§13.3.5), Slide (§13.3.8), or Slide Layout (§13.3.9) part via an implicit relationship.

[*Example*: The following WordprocessingML Main Document part-relationship item contains a relationship to the Theme part, which is stored in the ZIP item theme/theme1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/themeOverride" Target="theme/themeoverride1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be ossOverride.

[*Example*:

```
<a:ossOverride xmlns:a="…" >
  …
</a:ossOverride>
```

end example]

A Theme Override part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Theme Override part shall not contain implicit or explicit relationships to other parts defined by ISO/IEC 29500.

14.2.9 Table Styles Part

Content Type:	application/vnd.openxmlformats-officedocument.presentationml.tableStyles+xml
Root Namespace:	http://purl.oclc.org/ooxml/drawingml/main
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/tableStyles

An instance of this part type contains information about the table styles used by tables in this presentation. A table style defines characteristics such as row and column colors, heading row colors, and text.

A PresentationML package shall contain no more than one Table Styles part per Presentation (§13.3.6) part via an implicit relationship.

[*Example*: The following Presentation part-relationship item contains a relationship to a Table Styles part, which is stored in the ZIP item tableStyles.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/tableStyles" Target="tableStyles.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be tblStyleLst.

[*Example*: tablestyles.xml contains the following:

```
<a:tblStyleLst xmlns:a="…">
  <a:tblStyle>
    …
  </a:tblStyle>
</a:tblStyleLst>
```

end example]

A Table Styles part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Table Styles part shall not contain implicit or explicit relationships to other parts defined by ISO/IEC 29500.

15. Shared

The relationship items and parts defined in this clause are used by one or more of WordprocessingML (§11), SpreadsheetML (§12), and PresentationML (§13) environments.

15.1 Glossary of Shared Terms

array — An array of mathematical elements (“e”) stacked vertically in a single math zone.

build down — The process of converting mathematical text from an implementation’s professional form to an implementation’s built-down form.

build up — The process of converting mathematical text from an implementation’s built-down form to an implementation’s professional form.

built-down form — An implementation-specific linear format that may or may not include additional rich formatting in addition to another plain-text linear format (such as TeX) or the linear format defined in Unicode Technical Note 28.

built-up form — See professional form.

control — A region of active content within an Office Open XML document.

display equation — An equation that is in display mode, and thus is part of a display math zone. (Alternative names for display equation are: “display expression”, “display formula”, and “display math”.)

display mode — When mathematical text (i.e., text in one or more oMath blocks) is contained in a display math zone (i.e., an oMathPara block), the mathematical text represented in the oMath block(s) is in *display mode*.

equation array — An array of equations. See array.

inline equation — An equation that is in an inline math zone. (Alternative names for inline equation are: “inline expression”, “inline formula”, and simply “inline math”.)

instance of mathematical text — A single continuous combination of mathematical text represented by a single oMath block and the OMML elements within that oMath block.

linear format — An implementation-specific plain-text 1-dimensional representation of mathematical text.

math accent — A character that is specified as acceptable for use as an accent character by ISO/IEC 29500.

math alphanumerics — Characters with specific math styles, as defined in the Unicode Standard 5.0.

math paragraph — One or more oMath elements (instances of mathematical text) that are in display mode.

©ISO/IEC 2012 – All rights reserved

math zone — An isolated region of text within which mathematical text is used and outside of which mathematical text is not used.

mathematical text — Any text meant to convey mathematical meaning through OMML.

n-ary operator — An operator that involves n terms when expanded. For instance, the following example uses the Unicode (ISO 10646) summation sign (U+2211) which has the official name “N-ARY SUMMATION”.

$$\sum_{j=1}^{n} a_j \equiv a_1 + a_2 + \cdots + a_n$$

oMath container — An oMath block that is part of a display math zone but is not itself a math zone.

OMML — Office Math Markup Language, a Shared ML of ISO/IEC 29500.

professional form — Implementation-specific 2-D representation of mathematical text. (Also referred to as "built-up form".)

15.2 Part Summary

The subclauses subordinate to this one describe in detail each of the shared part types.

[*Note*: For convenience, information from those subclauses is summarized in the following table:

Part	Relationship Target of	Root Element	Ref.
Additional Characteristics	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Characteristics	§15.2.1
Audio	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Not applicable	§15.2.2
Bibliography	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Sources	§15.2.3
Custom XML Data Storage	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Not applicable	§15.2.4
Custom XML Data Storage Properties	Custom XML Data Storage	datastoreItem	§15.2.6
Digital Signature Origin	WordprocessingML, SpreadsheetML, or PresentationML package	Not applicable	§15.2.7
Digital Signature XML Signature	Digital Signature Origin	Signature	§15.2.8
Embedded Control	Numerous PresentationML,	Not applicable	§15.2.9

Part	Relationship Target of	Root Element	Ref.
Persistence	SpreadsheetML, and WordprocessingML parts		
Embedded Object	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Not applicable	§15.2.10
Embedded Package	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Not applicable	§15.2.11
File Properties, Extended	WordprocessingML, SpreadsheetML, or PresentationML package	Properties	§ 15.2.12.3
File Properties, Core	WordprocessingML, SpreadsheetML, or PresentationML package	coreProperties	§15.2.12.1
File Properties, Custom	WordprocessingML, SpreadsheetML, or PresentationML package	properties	§15.2.12.2
Font	WordprocessingML Font Table part, PresentationML Presentation part	Not applicable	§15.2.13
Image	Numerous PresentationML, SpreadsheetML, and WordprocessingML parts	Not applicable	§15.2.14
Printer Settings	SpreadsheetML Chartsheet, Dialogsheet, Worksheet parts, WordprocessingML Main Document or Glossary Document parts	Not applicable	§15.2.15
Thumbnail	WordprocessingML, SpreadsheetML, or PresentationML package	Not applicable	§15.2.16
Video part	Numerous PresentationML and WordprocessingML parts	Not applicable	§15.2.17

end note]

15.2.1 Additional Characteristics Part

Content Type:	application/xml
Root Namespace:	http://schemas.openxmlformats.org/officeDocument/2006/additionalCharacteristics
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/customXml

An instance of this part type contains information about additional characteristics of the producer that generated the document, when those characteristics cannot be specified using elements defined by ISO/IEC 29500. [*Note*: The contents of this part are purely informational, and do not place any requirements on subsequent consumption of the document. They are, however, intended to provide detailed information about the capabilities of the document's producer, allowing those capabilities to be factored in during subsequent processing. For example, an application which supports 100,000 spreadsheet columns might choose to limit its output to 10,000 columns when presented with a document whose characteristics indicate that it was produced by an application with that limitation, in order prevent the introduction of content which is unsupported by the original producer (as that application might be used in the future to process this document). This markup is provided by ISO/IEC 29500 in order to provide an interoperable way of storing this information. *end note*]

A package is permitted to contain zero or one Additional Characteristics parts, and each such part shall be the target of an implicit relationship from a Main Document (§11.3.10) part in a WordprocessingML package; a Workbook (§12.3.23) part in a SpreadsheetML package; or a Handout Master (§13.3.3) , Notes Master (§13.3.4), Notes Slide (§13.3.5), Presentation (§13.3.6), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part in a PresentationML package.

[*Example*: The following Main Document part-relationship item contains a relationship to an Additional Characteristics part, which is stored in the ZIP item ../customXML/item2.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/customXmlData" Target="../customXML/item2.xml"/>
</Relationships>
```

end example]

An Additional Characteristics part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be Internal).

An Additional Characteristics part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Custom XML Data Storage Properties (§15.2.6)

An Additional Characteristics part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

15.2.2 Audio Part

Content Type:	Any supported audio type. [*Note*: Some example content types are: audio/aiff — http://developer.apple.com/documentation/QuickTime/INMAC/SOUN

<table>
<tr><td></td><td><table>
<tr><td></td><td>D/imsoundmgr.30.htm</td></tr>
<tr><td>audio/midi</td><td>http://www.midi.org/about-midi/specinfo.shtml</td></tr>
<tr><td>audio/ogg</td><td>http://xiph.org/vorbis/doc/Vorbis_I_spec.html</td></tr>
<tr><td>audio/mpeg</td><td>ISO/IEC 11172-3</td></tr>
</table>
<i>end note</i>]</td></tr>
<tr><td>Root Namespace:</td><td>not applicable</td></tr>
<tr><td>Source Relationship:</td><td>http://purl.oclc.org/ooxml/officeDocument/relationships/audio</td></tr>
</table>

An instance of this part type contains an audio file.

A PresentationML package is permitted to contain zero or more Sound parts, each of which shall be the target of a relationship in a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part-relationship item. [*Example*: The following Slide part-relationship item contains a relationship to a Sound part, which is stored as the file E:\Beethoven's Symphony No. 9.wma:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/audio/x-ms-wma"
    Target="file:///E:/Beethoven's%20Symphony%20No.%209.wma"
    TargetMode="External"/>
</Relationships>
```

end example]

An Audio part can be located within or external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element can be `Internal` or `External`).

An Audio part is not stored as XML; instead, it involves a relationship target that is an audio clip.

An Audio part shall not have implicit or explicit relationships to other parts defined by ISO/IEC 29500.

A producer that wants interoperability should use the following standard format:

- audio/mpeg ISO/IEC 11172-3

15.2.3 Bibliography Part

Content Type:	application/xml
Root Namespace:	http://purl.oclc.org/ooxml/officeDocument/bibliography

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/customXml

An instance of this part type contains bibliographic data for the current package.

A package is permitted to contain zero or one Bibliography part, and each such part shall be the target of an implicit relationship in a Main Document (§11.3.10) part in a WordprocessingML package; a Workbook (§12.3.23) part in a SpreadsheetML package; or a Handout Master (§13.3.3) , Notes Master (§13.3.4), Notes Slide (§13.3.5), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part in a PresentationML package.

[*Example*: The following Main Document part-relationship item contains a relationship to a Bibliography part, which is stored in the ZIP item ../customXML/bib1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/customXml" Target="../customXML/bib1.xml"/>
</Relationships>
```

end example]

A Bibliography part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Bibliography part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Custom XML Data Storage Properties (§15.2.6)

A Bibliography part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

15.2.4 Content Part

<table>
<tr><td>Content Type:</td><td>Any supported XML content.

[<i>Note</i>: Some example content types are:
<table>
<tr><td>image/svg+xml</td><td>http://www.w3.org/TR/SVG11/</td></tr>
<tr><td>application/smil</td><td>http://www.w3.org/TR/REC-smil/</td></tr>
<tr><td>text/xml</td><td>http://www.w3.org/TR/MathML2/</td></tr>
</table>
<i>end note</i>]

If no explicit MIME type exists for a specific XML format, text/xml shall be used. Consumers who read a value of text/xml should determine the contents by the root namespace of the contents of the part.</td></tr>
<tr><td>Root Namespace:</td><td>Various, as defined by the content type used.</td></tr>
</table>

	[*Example*: MathML has a root namespace of http://www.w3.org/1998/Math/MathML. *end example*]
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/customXml

An instance of this part type can contain XML markup of a format not defined by ISO/IEC 29500.

A package is permitted to contain zero or more Content parts, and each such part shall be the target of an explicit relationship from a Comments (§11.3.2), Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Glossary Document (§11.3.8), Header (§11.3.9), or Main Document (§11.3.10) part in a WordprocessingML package; a Drawings (§12.3.8) part in a SpreadsheetML package; or a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), or a Slide Master (§13.3.10) part in a PresentationML package.

[*Example*: The following Main Document part-relationship item contains a relationship to a Content part containing SVG markup, which is stored in the ZIP item svg1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/customXml" Target="../customXML/svg1.xml"/>
</Relationships>
```

end example]

A Content part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Content part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

If a producer that wants interoperability supports equations, it should use one of the following standard formats:

- Office Open XML Math (§22.1
- W3C MathML 2.0

If a producer that wants interoperability supports ink annotations, it should use an ink annotation in this element in the following reference standard format:

- InkML http://www.w3.org/TR/inkregs

15.2.5 Custom XML Data Storage Part

Content Type:	application/xml
Root Namespace:	any XML allowed

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/customXml

An instance of this part type can contain arbitrary XML. As such, an instance of this part can be used to roundtrip arbitrary custom XML data with this package.

A package is permitted to contain one or more Custom XML Data Storage parts, and each such part shall be the target of an implicit relationship in a Main Document (§11.3.10) part in a WordprocessingML package; a Workbook (§12.3.23) part in a SpreadsheetML package; or a Handout Master (§13.3.3) , Notes Master (§13.3.4), Notes Slide (§13.3.5), Presentation (§13.3.6), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part in a PresentationML package.

[*Example*: The following Main Document part-relationship item contains a relationship to a Custom XML Data Storage part, which is stored in the ZIP item ../customXML/item1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/customXmlData" Target="../customXML/item1.xml"/>
</Relationships>
```

end example]

A Custom XML Data Storage part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be Internal).

A Custom XML Data Storage part is permitted to have implicit relationships to the following parts defined by ISO/IEC 29500:

- Custom XML Data Storage Properties (§15.2.6)

A Custom XML Data Storage part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

15.2.6 Custom XML Data Storage Properties Part

Content Type:	application/vnd.openxmlformats-officedocument.customXmlProperties+xml
Root Namespace:	http://purl.oclc.org/ooxml/officeDocument/customXmlDataProps
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/customXmlProps

An instance of this part type contains the set of properties which are specified for this custom XML data. These properties consist of a unique ID for the storage, as well as information on the set of XML schemas used by this custom XML data storage.

©ISO/IEC 2012 – All rights reserved

A package is permitted to contain zero or more Custom XML Data Storage Properties parts, and each such part shall be the target of an implicit relationship from a Custom XML Data Storage (§15.2.4) part.

[*Example*: The following Custom XML Data Storage part-relationship item contains a relationship to a Custom XML Data Storage Properties part, which is stored in the ZIP item itemProps1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/customXmlProps" Target="itemProps1.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be datastoreItem.

[*Example*:

```
<ds:datastoreItem ds:itemID="{D85…53A}" xmlns:ds="…"/> \
```

end example]

A Custom XML Data Storage Properties part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Custom XML Data Storage Properties part shall not have implicit or explicit relationships to other parts defined by ISO/IEC 29500.

15.2.7 Digital Signature Origin Part

This part is defined in §13.2.1, "Digital Signature Origin Part", of ISO/IEC 29500-2.

15.2.8 Digital Signature XML Signature Part

The part is defined in §13.2.2, "Digital Signature XML Signature Part", of ISO/IEC 29500-2.

15.2.9 Embedded Control Persistence Part

Content Type:	Any supported control type. [*Note*: There are a number of possible control types. One example of a potential control type would be an Active X control, which would use the following content type: `application/vnd.ms-office.activeX+xml`. *end note*]
Root Namespace:	not applicable
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/control

An instance of this part contains information about an embedded control in the package. This information is provided by the specified control when asked to persist. [*Example*: An application might utilize the embedded object server technology KParts or Bonobo to store an embedded object using this part. *end example*]

A package is permitted to contain one or more Embedded Control Persistence parts, and each such part shall be the target of an explicit relationship in an Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), or Main Document (§11.3.10) part-relationship item in a WordprocessingML package; a Worksheet part (§12.3.24) in a SpreadsheetML package; or a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), Slide Master (§13.3.10) part-relationship item in a PresentationML package.

The content type of this part shall determine the format and contents of the embedded control.

[*Example*: The following example shows the persistence that could be used for an embedded control which is a Java applet within a WordprocessingML document (the drawing object which provides a static image representation of the control, used when the Java applet itself is unavailable, has been omitted for brevity):

```
<w:p>
  <w:r w:rsidR="005810E1">
    <w:object w:dxaOrig="1440" w:dyaOrig="1440">
      <w:drawing>
        …
      </w:drawing>
      <w:control r:id="rId5" w:name="CommandButton1" w:shapeid="1027" />
    </w:object>
  </w:r>
</w:p>
```

The relationship type for rId5 is:
`http://purl.oclc.org/ooxml/officeDocument/relationships/control`

The XML content of the part referenced by rId5 could be:

```
<applet xlink:href="../../../../Program%20Files/Application" xlink:type="simple"
xlink:show="embed" xlink:actuate="onLoad" code="CalculateApplet.class" may-
script="false"/>
```

end example]

[*Example*: The following example shows the persistence that could be used for an embedded control which is an ActiveX control within a WordprocessingML document(the drawing object which provides a static image representation of the control, used when the ActiveX control itself is unavailable, has been omitted for brevity):

```
<w:p>
  <w:r w:rsidR="005810E1">
    <w:object w:dxaOrig="1440" w:dyaOrig="1440">
```

```
        <w:drawing>
          …
        </w:drawing>
        <w:control r:id="rId5" w:name="CommandButton1" w:shapeid="1027" />
      </w:object>
    </w:r>
  </w:p>
```

The relationship type for rId5 is:
`http://purl.oclc.org/ooxml/officeDocument/relationships/control`

The content type of the part referenced by rId5 could be: `application/vnd.ms-office.activeX+xml`

The XML content of the part referenced by rId5 could be:

```
  <ax:ocx ax:classid="{D7053240-CE69-11CD-A777-00DD01143C57}"
      ax:persistence="persistPropertyBag"
      xmlns:ax="http://schemas.microsoft.com/office/2006/activeX">
    <ax:ocxPr ax:name="Caption" ax:value="CommandButton1" />
    <ax:ocxPr ax:name="Size" ax:value="2540;847" />
    <ax:ocxPr ax:name="FontName" ax:value="Calibri" />
    <ax:ocxPr ax:name="FontHeight" ax:value="225" />
    <ax:ocxPr ax:name="FontCharSet" ax:value="0" />
    <ax:ocxPr ax:name="FontPitchAndFamily" ax:value="2" />
    <ax:ocxPr ax:name="ParagraphAlign" ax:value="3" />
  </ax:ocx>
```

end example]

An Embedded Control Persistence part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

An Embedded Control Persistence part shall not have any implicit or explicit relationships to other parts defined by ISO/IEC 29500.

15.2.10 Embedded Object Part

Content Type:	Any content type is allowed
Root Namespace:	not applicable
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/oleObject

An instance of this part type can contain an embedded object produced by any embedded object server.

A package is permitted to contain zero or more Embedded Object parts, and each such part shall be the target of an explicit relationship from a Comments (§11.3.2), Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), or Main Document (§11.3.10) part in a WordprocessingML package; a Worksheet part (§12.3.24) in a SpreadsheetML package; or a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), Slide Master (§13.3.10) part in a PresentationML package.

A WordprocessingML document package is permitted to contain zero or more Embedded Object parts, each of which shall be the target of a relationship in a Main Document part-relationship item. Each Embedded Object part shall have an associated image, which appears in the document as a placeholder for the corresponding embedded object.

[*Example*: Consider the case in which a WordprocessingML document has embedded in it one video object and one audio object. The following Main Document part-relationship item contains relationships to two Embedded parts (one each for the video and audio), which are stored in the ZIP items embeddings/embeddedObject*N*.bin:

```
<Relationships xmlns="…">
  <Relationship Id="rId5"
    Type="http://…/oleObject" Target="embeddings/embeddedObject1.bin"/>
  <Relationship Id="rId7"
    Type="http://…/oleObject" Target="embeddings/embeddedObject2.bin"/>
  <Relationship Id="rId4"
    Type="http://…/image" Target="media/image1.png"/>
  <Relationship Id="rId6"
    Type="http://…/image" Target="media/image2.png"/>
</Relationships>
```

example]

A SpreadsheetML document package is permitted to contain zero or more Embedded Object parts, each of which shall be the target of a relationship in a Worksheet part-relationship item.

[*Example*: Consider the case in which a SpreadsheetML document has embedded in it one video object and one audio object on one worksheet, and another audio object embedded in another worksheet. The following Worksheet Document part-relationship item contains relationships to two Embedded Object parts (one each for the video and audio), which are stored in the ZIP items ../embeddings/embeddedObject*N*.bin:

```
<Relationships xmlns="…">
   <Relationship Id="rId2"
     Type="http://…/oleObject" Target="../embeddings/embeddedObject1.bin"/>
  <Relationship Id="rId3"
    Type="http://…/oleObject" Target="../embeddings/embeddedObject2.bin"/>
</Relationships>
```

end example]

A PresentationML document package is permitted to contain zero or more Embedded Object parts, each of which shall be the target of a relationship in a Slide part-relationship item.

[*Example*: Consider the case in which a PresentationML document has embedded in it one video object and one audio object on one slide, and another audio object embedded on another slide. The following Slide part-relationship item contains relationships to two Embedded Object parts (one each for the video and audio), which are stored in the ZIP items ../embeddings/embeddedObject*N*.bin:

```
<Relationships xmlns="…">
  <Relationship Id="rId6"
    Type="http://…/oleObject"
    Target="../embeddings/embeddedObject1.bin"/>
  <Relationship Id="rId7"
    Type="http://…/oleObject"
    Target="../embeddings/embeddedObject2.bin"/>
</Relationships>
```

end example]

An Embedded Object part can be located within or external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element can be `Internal` or `External`).

An Embedded Object part is permitted to have an explicit relationship to the following parts defined by ISO/IEC 29500:

- Hyperlink (§15.3)

An Embedded Object part shall not have any implicit or explicit relationships to other parts defined by ISO/IEC 29500.

15.2.11 Embedded Package Part

Content Type:	Any content type is allowed
Root Namespace:	not applicable
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/package

An instance of this part type contains a complete package. For example, a WordprocessingML document might contain a SpreadsheetML or PresentationML document, in which case, the WordprocessingML document package would contain an embedded package part that defined that SpreadsheetML or PresentationML document.

A package is permitted to contain zero or more Embedded Package parts, and each such part shall be the target of an explicit relationship from a Chart (§14.2.1), Comments (§11.3.2), Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), or Main Document (§11.3.10) part in a WordprocessingML package; a Chart (§14.2.1), or Worksheet part (§12.3.24) in a SpreadsheetML package; or a Chart (§14.2.1), Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), Slide Master (§13.3.10) part in a PresentationML package.

[*Example*: The following Presentation part-relationship item contains relationships to two Embedded Package parts: one is a SpreadsheetML package, which is stored in the ZIP item embeddings/Worksheet1.xlsx, the other is a PresentationML package, which is stored in the ZIP item embeddings/Presentation2.pptx. The image files are used as document display placeholders if the consumer cannot handle the embedded package type:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/image" Target="media/image1.emf"/>
  <Relationship Id="rId5"
    Type="http:package" Target="embeddings/Worksheet1.xlsx"/>
  <Relationship Id="rId6"
    Type="http://…/image" Target="media/image2.emf"/>
  <Relationship Id="rId7"
    Type="http://…/package" Target="embeddings/Presentation2.pptx"/>
</Relationships>
```

end example]

An Embedded Package part can be located within or external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element can be `Internal` or `External`).

An Embedded Package part is permitted to have an explicit relationship to the following parts defined by ISO/IEC 29500:

- Hyperlink (§15.3)

An Embedded Package part shall not have any implicit or explicit relationships to other parts defined by ISO/IEC 29500.

15.2.12 File Properties

There are three kinds of file properties: *core*, *custom*, and *extended*. The *core file properties* of a package enable users to discover, get, and set common sets of properties from within that package, regardless of whether it's a WordprocessingML, SpreadsheetML, or PresentationML package. *Extended file properties* are specific to Office Open XML packages, while *custom file properties* are defined by the user, with each custom file property having a name, a value, and a type.

©ISO/IEC 2012 – All rights reserved

15.2.12.1 Core File Properties Part

This part and the related OPC part is defined in §11, “Core Properties”, of ISO/IEC 29500-2.

15.2.12.2 Custom File Properties Part

Content Type:	application/vnd.openxmlformats-officedocument.custom-properties+xml
Root Namespace:	http://purl.oclc.org/ooxml/officeDocument/customProperties
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/customProperties

An instance of this part contains the names of custom file properties that apply to the package, their values, and the types of those values. A custom file property might be the name of the client for whom the document was prepared, a date/time on which some event happened, a document number, or some Boolean status flag.

A package shall contain at most one Custom File Properties part, and that part shall be the target of a relationship in the package-relationship item for the document.

[*Example*: The following PresentationML's package-relationship item contains a relationship to a Custom File Properties part, stored in the ZIP item docProps/custom.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
  Type="http://…/custom-properties" Target="docProps/custom.xml"/>
</Relationships>
```

end example]

The root element for a part of this content type shall be Properties.

[*Example*: Here's some content markup from a WordprocessingML document, which contains four custom properties: `Client`, having a text value of "ACME Corp."; `Document number`, having a numeric value of 1543; `Recorded date`, having a date/time value of 2005-12-01; and `Special processing needed`, having a Boolean value of false:

```
<Properties … xmlns:vt="…">
  <property fmtid="{D5C…9AE}" pid="2" name="Client">
    <vt:lpwstr>ACME Corp.</vt:lpwstr>
  </property>
  <property fmtid="{D5C…9AE}" pid="3" name="Document number">
    <vt:i4>1543</vt:i4>
  </property>
```

```
    <property fmtid="{D5C…9AE}" pid="4" name="Recorded date">
      <vt:filetime>2005-12-01T05:00:00Z</vt:filetime>
    </property>
    <property fmtid="{D5C…9AE}" pid="5" name="Special processing needed">
      <vt:bool>false</vt:bool>
    </property>
  </Properties>
```

end example]

A Custom File Properties part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Custom File Properties part shall not have implicit or explicit relationships to other parts defined by ISO/IEC 29500.

15.2.12.3 Extended File Properties Part

Content Type:	application/vnd.openxmlformats-officedocument.extended-properties+xml
Root Namespace:	http://purl.oclc.org/ooxml/officeDocument/extendedProperties
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/extendedProperties

An instance of this part contains properties specific to an Office Open XML document. [*Example*: A PresentationML document specifies the number of slides in this presentation when last saved by a producer. *end example*]

A package shall contain at most one Extended File Properties part, and that part shall be the target of a relationship in the package-relationship item for the document.

[*Example*:

```
  <Relationships xmlns="…">
    <Relationship Id="rId4"
      Type="http://…/extended-properties" Target="docProps/app.xml"/>
  </Relationships>
```

end example]

The root element for a part of this content type shall be Properties.

[*Example*: Here's some content markup from a WordprocessingML document:

```
<Properties …>
  <Template>Normal.dotm</Template>
  <TotalTime>0</TotalTime>
  <Pages>1</Pages>
  <Words>3</Words>
  <Characters>22</Characters>
  <Application>Sample Producer</Application>
  <DocSecurity>0</DocSecurity>
  <Lines>1</Lines>
  <Paragraphs>1</Paragraphs>
  …
  <AppVersion>12.0000</AppVersion>
</Properties>
```

here's some content markup from a SpreadsheetML document:

```
<Properties …>
  <Application>Sample Producer</Application>
  <HeadingPairs>
  …
  </HeadingPairs>
  <TitlesOfParts>
  …
  </TitlesOfParts>
  <Company>Consultant</Company>
  …
</Properties>
```

and here's some content markup from a PresentationML document:

```
<Properties …>
  <Template>ppt_template_sdwest05</Template>
  <TotalTime>3166</TotalTime>
  <Words>37</Words>
  <Application>Sample Producer</Application>
  <PresentationFormat>On-screen Show</PresentationFormat>
  <Paragraphs>15</Paragraphs>
  <Slides>2</Slides>
  <Notes>2</Notes>
```

```
    …
    <HeadingPairs>
    …
    </HeadingPairs>
    <TitlesOfParts>
    …
    </TitlesOfParts>
    …
  </Properties>
```

end example]

A Extended File Properties part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

An Extended File Properties part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

15.2.13 Font Part

Content Type:	application/x-fontdata application/x-font-ttf application/vnd.openxmlformats-officedocument.obfuscatedFont
Root Namespace:	not applicable
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/font

An instance of this part type contains a given font embedded directly into the document. (This is useful when using custom fonts or fonts that are not widely distributed.)

Fonts stored in a Font part can be stored in one of the following formats, identified by the associated content type:

- `application/x-fontdata` specifies that the font shall be stored in the Embedded OpenType Format of http://www.w3.org/Submission/2008/SUBM-EOT-20080305
- `application/x-font-ttf` specifies that the font shall be stored in a format conforming to Open Font Structure defined in ISO/IEC 14496-22:2008 §3.5. [*Note*: The TrueType Collection format defined in ISO/IEC 14496-22:2008 §3.6 cannot be used. *end note*]
- application/vnd.openxmlformats officedocument.obfuscatedFont specifies that the font is obfuscated using the algorithm specified by Font Embedding (§17.8.1). The source font shall be stored in a format conforming to Open Font Structure defined in ISO/IEC 14496-22:2008 §3.5. [*Note*: The TrueType Collection format defined in ISO/IEC 14496-22:2008 §3.6 cannot be used. *end note*] Only packages of type WordprocessingML are permitted to reference this content type.

©ISO/IEC 2012 – All rights reserved

If a font is stored in the ISO/IEC 14496-22:2007 format, it shall only be used when stored as an individual font. [*Note:* Font collections should be converted into individual fonts before they are embedded using this part. *end note*]

A package shall contain zero or more Font parts, and for each that exists, that part shall be the target of an explicit relationship in the Font Table (§11.3.5), or Presentation (§13.3.6) part.

A Font part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Font part shall not have implicit or explicit relationships to other parts defined by ISO/IEC 29500.

15.2.14 Image Part

<table>
<tr><td>Content Type:</td><td>Any supported image type.

[<i>Note</i>: Some example content types are:
<table>
<tr><td>image/gif</td><td>http://www.w3.org/Graphics/GIF/spec-gif89a.txt</td></tr>
<tr><td>image/png</td><td>ISO/IEC 15948:2003 http://www.libpng.org/pub/png/spec/</td></tr>
<tr><td>image/tiff</td><td>http://partners.adobe.com/public/developer/tiff/index.html#spec</td></tr>
<tr><td>image/pict</td><td>http://developer.apple.com/documentation/mac/QuickDraw/QuickDraw-2.html</td></tr>
<tr><td>image/jpeg</td><td>http://www.w3.org/Graphics/JPEG/</td></tr>
</table>
<i>end note</i>.]</td></tr>
<tr><td>Root Namespace:</td><td>Not applicable</td></tr>
<tr><td>Source Relationship:</td><td>http://purl.oclc.org/ooxml/officeDocument/relationships/image</td></tr>
</table>

An image can be stored in a package as a ZIP item. Image ZIP items shall be identified by an image part relationship and the appropriate content type.

A package is permitted to contain zero or more Image parts, and each such part shall be the target of an explicit relationship from a Comments (§11.3.2), Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), Drawing (§12.3.8), or Main Document (§11.3.10) part in a WordprocessingML package or a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10 part in a PresentationML package.

[*Example*: The following PresentationML's package-relationship item contains one relationship, for the slide template jpeg image stored in the ZIP item ../media/image1.jpeg:

```
<Relationships xmlns="…">
  <Relationship Id="rId8"
    Type="http://…/image" Target="../media/image1.jpeg"/>
</Relationships>
```

end example]

An Image part can be located within or external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element can be `Internal` or `External`).

An Image part shall not have implicit or explicit relationships to other parts defined by ISO/IEC 29500.

A producer that wants interoperability should use one of the following standard formats:

- image/png ISO/IEC 15948:2003, http://www.libpng.org/pub/png/spec/
- image/jpeg, http://www.w3.org/Graphics/JPEG

15.2.15 Printer Settings Part

Content Type:	application/vnd.openxmlformats-officedocument.spreadsheetml.printerSettings (in SpreadsheetML documents) application/vnd.openxmlformats-officedocument.wordprocessingml.printerSettings (in WordprocessingML documents) application/vnd.openxmlformats-officedocument.presentationml.printerSettings (in PresentationML documents)
Root Namespace:	not applicable
Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings

An instance of this part type contains information about the initialization and environment of a printer or a display device. The layout of this information is application-defined.

[*Note*: It is recommended that a Printer Settings Part contain well documented XML content for improved interoperability; however, there is no requirement on the format of the content contained in a Printer Settings Part. *end note*]

[*Example*: An Office Open XML producer on Windows might store the DEVMODE structure defined here: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/gdi/prntspol_8nle.asp, while an application on the Mac OS might choose to store the print record defined here: http://developer.apple.com/documentation/Printing/index.html. *end example*]

©ISO/IEC 2012 – All rights reserved

A SpreadsheetML package is permitted to contain at most one Printer Settings part per Chartsheet, Dialogsheet, or Worksheet part, and that part shall be the target of an implicit relationship from a Chartsheet (§12.3.2), Dialogsheet (§12.3.7), or Worksheet (§12.3.24) part. A WordprocessingML package is permitted to contain zero or more Printer Settings parts, one per sectPr element, each a target of an explicit relationship from a Main Document (§11.3.10) or Glossary Document (§11.3.8) part. A PresentationML package is permitted to contain at most one Printer Settings part, and that part shall be the target of an implicit relationship from a Presentation (§13.3.6) part.

[*Example*: The following SpreadsheetML Worksheet part-relationship item contains a relationship to a Printer Settings part, which is stored in the ZIP item ../printerSettings/printerSettings1.xml:

```
<Relationships xmlns="…">
  <Relationship Id="rId4"
    Type="http://…/printerSettings"
    Target="../printerSettings/printerSettings1.xml"/>
</Relationships>
```

where the contents of PrinterSettings1.xml contains the following XML:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<PrinterSettings xmlns="…">
    <PrinterSetting name="PropertyName" value="PropertyValue" />
</PrinterSettings>
```

end example]

A Printer Settings part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be Internal).

A Printer Settings part shall not have implicit or explicit relationships to any other part defined by ISO/IEC 29500.

15.2.16 Thumbnail Part

<table>
<tr><td>Content Type:</td><td>Any supported image type.

[<i>Note</i>: Some example content types are:
<table>
<tr><td>image/gif</td><td>http://www.w3.org/Graphics/GIF/spec-gif89a.txt</td></tr>
<tr><td>image/png</td><td>ISO/IEC 15948:2003 http://www.libpng.org/pub/png/spec/</td></tr>
<tr><td>image/tiff</td><td>http://partners.adobe.com/public/developer/tiff/index.html#spec</td></tr>
<tr><td>image/pict</td><td>http://developer.apple.com/documentation/mac/QuickDraw/QuickDraw-2.html</td></tr>
<tr><td>image/jpeg</td><td>http://www.w3.org/Graphics/JPEG/</td></tr>
</table>
<i>end note</i>.]</td></tr>
<tr><td>Root Namespace:</td><td>Not applicable</td></tr>
</table>

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/metadata/thumbnail

To help end-users identify parts of a package or the package as a whole, images, called *thumbnails*, can be stored in that package. Each thumbnail image is generated by the package producer and is stored in the package as a ZIP item. There are no limitations on the size or dimensions of the thumbnail produced, and applications are free to scale the images as desired.

Thumbnail ZIP items shall be identified by either a package-relationship item or a part-relationship item. Packages shall not contain more than one thumbnail relationship associated with the package as a whole, or more than one thumbnail relationship per package part.

[*Example*: The following PresentationML's package-relationship item contains one relationship, for the metafile image stored in the ZIP item thumbnail.wmf:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/thumbnail" Target="docProps/thumbnail.wmf"/>
</Relationships>
```

end example]

A Thumbnail part shall be located within the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element shall be `Internal`).

A Thumbnail part shall not have implicit or explicit relationships to other parts defined by ISO/IEC 29500.

15.2.17 Video Part

<table>
<tr><td>Content Type:</td><td>Any supported video type.
[<i>Note</i>: Some example content types are:
<table>
<tr><td>video/avi</td><td>http://www.the-labs.com/Video/odmlff2-avidef.pdf</td></tr>
<tr><td>video/mpg</td><td>ISO/IEC 13818</td></tr>
<tr><td>video/mpeg</td><td>ISO/IEC 13818</td></tr>
<tr><td>video/ogg</td><td>http://www.theora.org/doc/Theora.pdf</td></tr>
<tr><td>video/quicktime</td><td>http://developer.apple.com/documentation/QuickTime/</td></tr>
<tr><td>video/vc1</td><td>http://tools.ietf.org/html/rfc4425</td></tr>
</table>
<i>end note</i>]</td></tr>
<tr><td>Root Namespace:</td><td>not applicable</td></tr>
<tr><td>Source Relationship:</td><td>http://purl.oclc.org/ooxml/officeDocument/relationships/video</td></tr>
</table>

©ISO/IEC 2012 – All rights reserved

An instance of this part type contains a video file.

A PresentationML package is permitted to contain zero or more Video parts, each of which shall be the target of an explicit relationship in a Handout Master (§13.3.3), Notes Slide (§13.3.5), Notes Master (§13.3.4), Slide (§13.3.8), Slide Layout (§13.3.9), or Slide Master (§13.3.10) part. A WordprocessingML package is permitted to contain zero or more Video parts, each of which shall be the target of an explicit relationship from a Comments (§11.3.2), Endnotes (§11.3.4), Footer (§11.3.6), Footnotes (§11.3.7), Header (§11.3.9), or Main Document (§11.3.10) part.

[*Example*: The following Slide part-relationship item contains a relationship to a Video part, which is stored as the file E:\Video demo.avi:

```
<Relationships xmlns="…">
  <Relationship Id="rId2"
    Type="http://…/video"
    Target="file:///E:\Video%20demo.avi" TargetMode="External"/>
</Relationships>
```

end example]

A Video part is not stored as XML; instead, it involves a relationship target that is a video clip.

A Video part can be located within or external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element can be `Internal` or `External`).

A Video part shall not have implicit or explicit relationships to other parts defined by ISO/IEC 29500.

A producer that wants interoperability should use the following standard format:

- video/mpeg ISO/IEC 13818

15.3 Hyperlinks

Source Relationship:	http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink

A hyperlink can be stored in a package as a relationship. Hyperlinks shall be identified by containing a target which specifies the destination of the given hyperlink.

[*Example*: The following WordprocessingML Footnote part's relationship part contains one relationship, for the hyperlink http://schemas.openxmlformats.org/wordprocessingml/:

```
<Relationships xmlns="…">
  <Relationship Id="rId1"
    Type="http://…/hyperlink"
    Target="http://schemas.openxmlformats.org/wordprocessingml/"
    TargetMode="External"/>
</Relationships>
```

end example]

A hyperlink target can be located within or external to the package containing the relationships part (expressed syntactically, the TargetMode attribute of the Relationship element can be `Internal` or `External`).

©ISO/IEC 2012 – All rights reserved

16. Part Overview

This clause is informative.

For convenience, the following subclauses specify the root elements for each part and, when applicable, link to the appropriate subclause that defines the root element for that part within the package. [*Note*: The terms part and package, as used here, are defined in ISO/IEC 29500-2. *end note*] Note that the contents of some of these parts are not defined in this Part of ISO/IEC 29500, in which case, the root element is listed as "Not applicable" and the Reference is "n/a". For those parts where the Root Element is listed as "Not Applicable", a definition of the part's use and how its content shall be structured is defined in this Part 1 of ISO/IEC 29500.

16.1 WordprocessingML Summary

Part	Root Element	Ref.
Alternative Format Import	Not applicable	n/a
Comments	comments	§17.13.4.6
Document Settings	settings	§17.15.1.78
Endnotes	endnotes	§17.11.8
Font Table	fonts	§17.8.3.11
Footer	ftr	§17.10.3
Footnotes	footnotes	§17.11.15
Glossary Document	glossaryDocument	§17.12.10
Header	hdr	§17.10.4
Mail Merge Recipient Data	recipients	§17.14.29
Main Document	document	§17.2.3
Numbering Definitions	numbering	§17.9.16
Style Definitions	styles	§17.7.4.18
Web Settings	webSettings	§17.15.2.45

16.2 SpreadsheetML Summary

Part	Root Element	Ref.
Calculation Chain	calcChain	§18.6.2
Chartsheet	chartsheet	§18.3.1.12
Comments	comments	§18.7.6
Connections	connections	§18.13.2
Custom Property	Not applicable	n/a

Part	Root Element	Ref.
Custom XML Mappings	MapInfo	§18.16.3
Dialogsheet	dialogsheet	§18.3.1.34
Drawing	wsDr	§20.5.2.35
External Workbook References	externalLink	§18.14.8
Metadata	metadata	§18.9.8
Pivot Table	pivotTableDefinition	§18.10.1.73
Pivot Table Cache Definition	pivotCacheDefinition	§18.10.1.67
Pivot Table Cache Records	pivotCacheRecords	§18.10.1.68
Query Table	queryTable	§18.12.2
Shared String Table	sst	§18.4.9
Shared Workbook Revision Headers	headers	§18.11.1.1
Shared Workbook Revision Log	revisions	§18.11.1.16
Shared Workbook User Data	users	§18.11.2.2
Single Cell Table Definitions	singleXmlCells	§18.5.2.2
Styles	styleSheet	§18.8.39
Table Definition	table	§18.5.1.2
Volatile Dependencies	volTypes	§18.15.6
Workbook	workbook	§18.2.27
Worksheet	worksheet	§18.3.1.99

16.3 PresentationML Summary

Part	Root Element	Ref.
Comment Authors	cmAuthorLst	§19.4.3
Comments	cmLst	§19.4.4
Handout Master	handoutMaster	§19.3.1.24
Notes Master	notesMaster	§19.3.1.27
Notes Slide	notes	§19.3.1.26
Presentation	presentation	§19.2.1.26
Presentation Properties	presentationPr	§19.2.1.27
Slide	sld	§19.3.1.38
Slide Layout	sldLayout	§19.3.1.39
Slide Master	sldMaster	§19.3.1.42
Slide Synchronization Data	sldSyncPr	§19.6.1
User-Defined Tags	tagLst	§19.3.3.2

©ISO/IEC 2012 – All rights reserved

Part	Root Element	Ref.
View Properties	viewPr	§19.2.2.18

16.4 DrawingML Summary

Part	Root Element	Ref.
Chart	chartSpace	§21.2.2.29
Chart Drawing	userShapes	§21.2.2.220
Diagram Colors	colorsDef	§21.4.4.3
Diagram Data	dataModel	§21.4.2.10
Diagram Layout Definition	layoutDef	§21.4.2.16
Diagram Style	styleDef	§21.4.5.7
Theme	officeStyleSheet	§20.1.6.9
Theme Override	themeOverride	§20.1.6.12
Table Styles	tblStyleLst	§20.1.4.2.27

16.5 Shared Summary

Part	Root Element	Ref.
Additional Characteristics	additionalCharacteristics	§22.7.2.1
Audio	Not applicable	n/a
Bibliography	Sources	§22.6.2.60
Custom XML Data Storage	Not applicable	n/a
Custom XML Data Storage Properties	datastoreItem	§22.5.2.1
Digital Signature Origin	Not applicable	n/a
Digital Signature XML Signature	Signature	Defined in ISO/IEC 29500-2
Embedded Control Persistence	Not applicable	n/a
Embedded Object	Not applicable	n/a
Embedded Package	Not applicable	n/a
File Properties, Core	coreProperties	Defined in ISO/IEC 29500-2
File Properties, Custom	Properties	§22.3.2.1
File Properties, Extended	Properties	§22.2.2.21
Font	Not applicable	n/a

Part	Root Element	Ref.
Image	Not applicable	n/a
Printer Settings	Not applicable	n/a
Thumbnail	Not applicable	n/a
Video	Not applicable	n/a

End of informative text.

©ISO/IEC 2012 – All rights reserved

17. WordprocessingML Reference Material

[*Note*: For further information on the mapping of elements and attributes to OPC parts, see the Bibliography entry, “Information on elements, attributes, and OPC parts in ISO/IEC 29500 (OOXML)”. *end note*]

The subordinate subclauses specify the semantics for the XML markup comprising a WordprocessingML document, as defined by §11 of this Part of ISO/IEC 29500.

17.1 Table of Contents

This subclause is informative.

17.2 Main Document Story ... 189
17.2.1 background (Document Background) ... 189
17.2.2 body (Document Body) ... 194
17.2.3 document (Document) ... 194

17.3 Paragraphs and Rich Formatting ... 195
17.3.1 Paragraphs ... 196
17.3.1.1 adjustRightInd (Automatically Adjust Right Indent When Using Document Grid) ... 196
17.3.1.2 autoSpaceDE (Automatically Adjust Spacing of Latin and East Asian Text) ... 197
17.3.1.3 autoSpaceDN (Automatically Adjust Spacing of East Asian Text and Numbers) ... 198
17.3.1.4 bar (Paragraph Border Between Facing Pages) ... 198
17.3.1.5 between (Paragraph Border Between Identical Paragraphs) ... 199
17.3.1.6 bidi (Right to Left Paragraph Layout) ... 201
17.3.1.7 bottom (Paragraph Border Below Identical Paragraphs) ... 201
17.3.1.8 cnfStyle (Paragraph Conditional Formatting) ... 203
17.3.1.9 contextualSpacing (Ignore Spacing Above and Below When Using Identical Styles) ... 208
17.3.1.10 divId (Associated HTML div ID) ... 209
17.3.1.11 framePr (Text Frame Properties) ... 210
17.3.1.12 ind (Paragraph Indentation) ... 221
17.3.1.13 jc (Paragraph Alignment) ... 226
17.3.1.14 keepLines (Keep All Lines On One Page) ... 227
17.3.1.15 keepNext (Keep Paragraph With Next Paragraph) ... 228
17.3.1.16 kinsoku (Use East Asian Typography Rules for First and Last Character per Line) ... 230
17.3.1.17 left (Left Paragraph Border) ... 233
17.3.1.18 mirrorIndents (Use Left/Right Indents as Inside/Outside Indents) ... 234
17.3.1.19 numPr (Numbering Definition Instance Reference) ... 235
17.3.1.20 outlineLvl (Associated Outline Level) ... 235
17.3.1.21 overflowPunct (Allow Punctuation to Extend Past Text Extents) ... 236
17.3.1.22 p (Paragraph) ... 237
17.3.1.23 pageBreakBefore (Start Paragraph on Next Page) ... 239
17.3.1.24 pBdr (Paragraph Borders) ... 241
17.3.1.25 pPr (Previous Paragraph Properties) ... 241
17.3.1.26 pPr (Paragraph Properties) ... 242

©ISO/IEC 2012 – All rights reserved

17.3.1.27 pStyle (Referenced Paragraph Style) ... 242
17.3.1.28 right (Right Paragraph Border) ... 244
17.3.1.29 rPr (Run Properties for the Paragraph Mark) ... 245
17.3.1.30 rPr (Previous Run Properties for the Paragraph Mark) ... 246
17.3.1.31 shd (Paragraph Shading) ... 247
17.3.1.32 snapToGrid (Use Document Grid Settings for Inter-Line Paragraph Spacing) ... 247
17.3.1.33 spacing (Spacing Between Lines and Above/Below Paragraph) ... 248
17.3.1.34 suppressAutoHyphens (Suppress Hyphenation for Paragraph) ... 254
17.3.1.35 suppressLineNumbers (Suppress Line Numbers for Paragraph) ... 254
17.3.1.36 suppressOverlap (Prevent Text Frames From Overlapping) ... 255
17.3.1.37 tab (Custom Tab Stop) ... 256
17.3.1.38 tabs (Set of Custom Tab Stops) ... 257
17.3.1.39 textAlignment (Vertical Character Alignment on Line) ... 258
17.3.1.40 textboxTightWrap (Allow Surrounding Paragraphs to Tight Wrap to Text Box Contents) ... 259
17.3.1.41 textDirection (Paragraph Text Flow Direction) ... 261
17.3.1.42 top (Paragraph Border Above Identical Paragraphs) ... 262
17.3.1.43 topLinePunct (Compress Punctuation at Start of a Line) ... 263
17.3.1.44 widowControl (Allow First/Last Line to Display on a Separate Page) ... 264
17.3.1.45 wordWrap (Allow Line Breaking At Character Level) ... 265
17.3.2 Run ... 266
17.3.2.1 b (Bold) ... 266
17.3.2.2 bCs (Complex Script Bold) ... 267
17.3.2.3 bdo (Bidirectional Override) ... 267
17.3.2.4 bdr (Text Border) ... 268
17.3.2.5 caps (Display All Characters As Capital Letters) ... 269
17.3.2.6 color (Run Content Color) ... 270
17.3.2.7 cs (Use Complex Script Formatting on Run) ... 273
17.3.2.8 dir (Bidirectional Embedding Level) ... 274
17.3.2.9 dstrike (Double Strikethrough) ... 275
17.3.2.10 eastAsianLayout (East Asian Typography Settings) ... 276
17.3.2.11 effect (Animated Text Effect) ... 280
17.3.2.12 em (Emphasis Mark) ... 281
17.3.2.13 emboss (Embossing) ... 281
17.3.2.14 fitText (Manual Run Width) ... 282
17.3.2.15 highlight (Text Highlighting) ... 284
17.3.2.16 i (Italics) ... 285
17.3.2.17 iCs (Complex Script Italics) ... 285
17.3.2.18 imprint (Imprinting) ... 286
17.3.2.19 kern (Font Kerning) ... 286
17.3.2.20 lang (Languages for Run Content) ... 287
17.3.2.21 noProof (Do Not Check Spelling or Grammar) ... 290
17.3.2.22 oMath (Office Open XML Math) ... 290
17.3.2.23 outline (Display Character Outline) ... 291
17.3.2.24 position (Vertically Raised or Lowered Text) ... 292
17.3.2.25 r (Text Run) ... 293
17.3.2.26 rFonts (Run Fonts) ... 294
17.3.2.27 rPr (Previous Run Properties) ... 300
17.3.2.28 rPr (Run Properties) ... 301

17.3.2.29 rStyle (Referenced Character Style) ... 301
17.3.2.30 rtl (Right To Left Text) ... 303
17.3.2.31 shadow (Shadow) ... 304
17.3.2.32 shd (Run Shading) ... 305
17.3.2.33 smallCaps (Small Caps) ... 305
17.3.2.34 snapToGrid (Use Document Grid Settings For Inter-Character Spacing) ... 306
17.3.2.35 spacing (Character Spacing Adjustment) ... 307
17.3.2.36 specVanish (Paragraph Mark Is Always Hidden) ... 308
17.3.2.37 strike (Single Strikethrough) ... 308
17.3.2.38 sz (Non-Complex Script Font Size) ... 309
17.3.2.39 szCs (Complex Script Font Size) ... 310
17.3.2.40 u (Underline) ... 311
17.3.2.41 vanish (Hidden Text) ... 315
17.3.2.42 vertAlign (Subscript/Superscript Text) ... 315
17.3.2.43 w (Expanded/Compressed Text) ... 316
17.3.2.44 webHidden (Web Hidden Text) ... 317
17.3.3 Run Content ... 318
17.3.3.1 br (Break) ... 318
17.3.3.2 contentPart (Content Part) ... 320
17.3.3.3 control (Embedded Control) ... 322
17.3.3.4 cr (Carriage Return) ... 324
17.3.3.5 dayLong (Date Block - Long Day Format) ... 325
17.3.3.6 dayShort (Date Block - Short Day Format) ... 326
17.3.3.7 delText (Deleted Text) ... 326
17.3.3.8 dirty (Invalidated Field Cache) ... 327
17.3.3.9 drawing (DrawingML Object) ... 327
17.3.3.10 hps (Phonetic Guide Text Font Size) ... 328
17.3.3.11 hpsBaseText (Phonetic Guide Base Text Font Size) ... 329
17.3.3.12 hpsRaise (Distance Between Phonetic Guide Text and Phonetic Guide Base Text) ... 330
17.3.3.13 lastRenderedPageBreak (Position of Last Calculated Page Break) ... 330
17.3.3.14 lid (Language ID for Phonetic Guide) ... 331
17.3.3.15 monthLong (Date Block - Long Month Format) ... 332
17.3.3.16 monthShort (Date Block - Short Month Format) ... 332
17.3.3.17 movie (Embedded Video) ... 333
17.3.3.18 noBreakHyphen (Non Breaking Hyphen Character) ... 334
17.3.3.19 object (Embedded Object) ... 335
17.3.3.20 objectEmbed (Embedded Object Properties) ... 337
17.3.3.21 objectLink (Linked Object Properties) ... 339
17.3.3.22 pgNum (Page Number Block) ... 341
17.3.3.23 ptab (Absolute Position Tab Character) ... 342
17.3.3.24 rt (Phonetic Guide Text) ... 344
17.3.3.25 ruby (Phonetic Guide) ... 344
17.3.3.26 rubyAlign (Phonetic Guide Text Alignment) ... 345
17.3.3.27 rubyBase (Phonetic Guide Base Text) ... 346
17.3.3.28 rubyPr (Phonetic Guide Properties) ... 347
17.3.3.29 softHyphen (Optional Hyphen Character) ... 347
17.3.3.30 sym (Symbol Character) ... 348
17.3.3.31 t (Text) ... 350

17.3.3.32 tab (Tab Character) ... 351
17.3.3.33 yearLong (Date Block - Long Year Format) ... 351
17.3.3.34 yearShort (Date Block - Short Year Format) ... 352
17.3.4 Border Properties (CT_Border) ... 353
17.3.5 Shading Properties (CT_Shd) ... 359

17.4 Tables ... 366

17.4.1 bidiVisual (Visually Right to Left Table) ... 367
17.4.2 bottom (Table Cell Bottom Margin Exception) ... 369
17.4.3 bottom (Table Cell Bottom Border) ... 369
17.4.4 bottom (Table Bottom Border) ... 370
17.4.5 bottom (Table Cell Bottom Margin Default) ... 371
17.4.6 cantSplit (Table Row Cannot Break Across Pages) ... 372
17.4.7 cnfStyle (Table Row Conditional Formatting) ... 373
17.4.8 cnfStyle (Table Cell Conditional Formatting) ... 379
17.4.9 divId (Associated HTML div ID) ... 384
17.4.10 end (Table Cell Trailing Margin Exception) ... 386
17.4.11 end (Table Cell Trailing Margin Default) ... 387
17.4.12 end (Table Cell Trailing Edge Border) ... 388
17.4.13 end (Table Trailing Edge Border) ... 389
17.4.14 gridAfter (Grid Columns After Last Cell) ... 389
17.4.15 gridBefore (Grid Columns Before First Cell) ... 391
17.4.16 gridCol (Grid Column Definition) ... 392
17.4.17 gridSpan (Grid Columns Spanned by Current Table Cell) ... 393
17.4.18 header (Header Cell Reference) ... 395
17.4.19 headers (Header Cells Associated With Table Cell) ... 397
17.4.20 hidden (Hidden Table Row Marker) ... 399
17.4.21 hideMark (Ignore End Of Cell Marker In Row Height Calculation) ... 400
17.4.22 insideH (Table Inside Horizontal Edges Border) ... 401
17.4.23 insideH (Table Cell Inside Horizontal Edges Border) ... 401
17.4.24 insideV (Table Inside Vertical Edges Border) ... 402
17.4.25 insideV (Table Cell Inside Vertical Edges Border) ... 403
17.4.26 jc (Table Alignment Exception) ... 404
17.4.27 jc (Table Row Alignment) ... 405
17.4.28 jc (Table Alignment) ... 406
17.4.29 noWrap (Don't Wrap Cell Content) ... 407
17.4.30 shd (Table Shading Exception) ... 408
17.4.31 shd (Table Shading) ... 409
17.4.32 shd (Table Cell Shading) ... 410
17.4.33 start (Table Cell Leading Edge Border) ... 411
17.4.34 start (Table Cell Leading Margin Default) ... 412
17.4.35 start (Table Cell Leading Margin Exception) ... 412
17.4.36 start (Table Leading Edge Border) ... 413
17.4.37 tbl (Table) ... 414
17.4.38 tblBorders (Table Borders) ... 415
17.4.39 tblBorders (Table Borders Exceptions) ... 417
17.4.40 tblCaption (Table Caption) ... 418
17.4.41 tblCellMar (Table Cell Margin Exceptions) ... 419

17.4.42 tblCellMar (Table Cell Margin Defaults) 420
17.4.43 tblCellSpacing (Table Row Cell Spacing) 421
17.4.44 tblCellSpacing (Table Cell Spacing Exception) 422
17.4.45 tblCellSpacing (Table Cell Spacing Default) 423
17.4.46 tblDescription (Table Description) 424
17.4.47 tblGrid (Previous Table Grid) 425
17.4.48 tblGrid (Table Grid) 426
17.4.49 tblHeader (Repeat Table Row on Every New Page) 427
17.4.50 tblInd (Table Indent from Leading Margin) 428
17.4.51 tblInd (Table Indent from Leading Margin Exception) 429
17.4.52 tblLayout (Table Layout) 430
17.4.53 tblLayout (Table Layout Exception) 431
17.4.54 tblLook (Table Style Conditional Formatting Settings Exception) 432
17.4.55 tblLook (Table Style Conditional Formatting Settings) 433
17.4.56 tblOverlap (Floating Table Allows Other Tables to Overlap) 434
17.4.57 tblpPr (Floating Table Positioning) 436
17.4.58 tblPr (Previous Table Properties) 442
17.4.59 tblPr (Table Properties) 443
17.4.60 tblPrEx (Table-Level Property Exceptions) 444
17.4.61 tblPrEx (Previous Table-Level Property Exceptions) 446
17.4.62 tblStyle (Referenced Table Style) 448
17.4.63 tblW (Preferred Table Width) 449
17.4.64 tblW (Preferred Table Width Exception) 450
17.4.65 tc (Table Cell) 451
17.4.66 tcBorders (Table Cell Borders) 453
17.4.67 tcFitText (Fit Text Within Cell) 456
17.4.68 tcMar (Single Table Cell Margins) 456
17.4.69 tcPr (Table Cell Properties) 457
17.4.70 tcPr (Previous Table Cell Properties) 458
17.4.71 tcW (Preferred Table Cell Width) 459
17.4.72 textDirection (Table Cell Text Flow Direction) 461
17.4.73 tl2br (Table Cell Top Left to Bottom Right Diagonal Border) 462
17.4.74 top (Table Cell Top Border) 462
17.4.75 top (Table Cell Top Margin Default) 463
17.4.76 top (Table Top Border) 464
17.4.77 top (Table Cell Top Margin Exception) 465
17.4.78 tr (Table Row) 465
17.4.79 tr2bl (Table Cell Top Right to Bottom Left Diagonal Border) 467
17.4.80 trHeight (Table Row Height) 468
17.4.81 trPr (Table Row Properties) 470
17.4.82 trPr (Previous Table Row Properties) 471
17.4.83 vAlign (Table Cell Vertical Alignment) 472
17.4.84 vMerge (Vertically Merged Cell) 473
17.4.85 wAfter (Preferred Width After Table Row) 475
17.4.86 wBefore (Preferred Width Before Table Row) 476
17.4.87 Table Measurement (CT_TblWidth) 477
17.5 Custom Markup 478

17.5.1 Custom XML and Smart Tags ... 478
17.5.1.1 attr (Custom XML Attribute) ... 481
17.5.1.2 attr (Smart Tag Property) ... 482
17.5.1.3 customXml (Inline-Level Custom XML Element) ... 484
17.5.1.4 customXml (Cell-Level Custom XML Element) ... 485
17.5.1.5 customXml (Row-Level Custom XML Element) ... 487
17.5.1.6 customXml (Block-Level Custom XML Element) ... 488
17.5.1.7 customXmlPr (Custom XML Element Properties) ... 489
17.5.1.8 placeholder (Custom XML Element Placeholder Text) ... 490
17.5.1.9 smartTag (Inline-Level Smart Tag) ... 491
17.5.1.10 smartTagPr (Smart Tag Properties) ... 493
17.5.2 Structured Document Tags ... 493
17.5.2.1 alias (Friendly Name) ... 494
17.5.2.2 bibliography (Bibliography Structured Document Tag) ... 495
17.5.2.3 calendar (Date Picker Calendar Type) ... 496
17.5.2.4 citation (Citation Structured Document Tag) ... 497
17.5.2.5 comboBox (Combo Box Structured Document Tag) ... 497
17.5.2.6 dataBinding (XML Mapping) ... 499
17.5.2.7 date (Date Structured Document Tag) ... 502
17.5.2.8 dateFormat (Date Display Mask) ... 503
17.5.2.9 docPart (Document Part Reference) ... 505
17.5.2.10 docPartCategory (Document Part Category Filter) ... 506
17.5.2.11 docPartGallery (Document Part Gallery Filter) ... 507
17.5.2.12 docPartList (Document Part Gallery Structured Document Tag) ... 508
17.5.2.13 docPartObj (Built-In Document Part Structured Document Tag) ... 509
17.5.2.14 docPartUnique (Built-In Document Part) ... 510
17.5.2.15 dropDownList (Drop-Down List Structured Document Tag) ... 510
17.5.2.16 equation (Equation Structured Document Tag) ... 512
17.5.2.17 group (Group Structured Document Tag) ... 513
17.5.2.18 id (Unique ID) ... 513
17.5.2.19 label (Structured Document Tag Label) ... 514
17.5.2.20 lid (Date Picker Language ID) ... 516
17.5.2.21 listItem (Combo Box List Item) ... 516
17.5.2.22 listItem (Drop-Down List Item) ... 518
17.5.2.23 lock (Locking Setting) ... 520
17.5.2.24 picture (Picture Structured Document Tag) ... 522
17.5.2.25 placeholder (Structured Document Tag Placeholder Text) ... 522
17.5.2.26 richText (Rich Text Structured Document Tag) ... 523
17.5.2.27 rPr (Run Properties For Structured Document Tag Contents) ... 523
17.5.2.28 rPr (Structured Document Tag End Character Run Properties) ... 525
17.5.2.29 sdt (Block-Level Structured Document Tag) ... 525
17.5.2.30 sdt (Row-Level Structured Document Tag) ... 526
17.5.2.31 sdt (Inline-Level Structured Document Tag) ... 527
17.5.2.32 sdt (Cell-Level Structured Document Tag) ... 528
17.5.2.33 sdtContent (Cell-Level Structured Document Tag Content) ... 529
17.5.2.34 sdtContent (Block-Level Structured Document Tag Content) ... 530
17.5.2.35 sdtContent (Row-Level Structured Document Tag Content) ... 531
17.5.2.36 sdtContent (Inline-Level Structured Document Tag Content) ... 532

17.5.2.37 sdtEndPr (Structured Document Tag End Character Properties) ... 533
17.5.2.38 sdtPr (Structured Document Tag Properties) ... 533
17.5.2.39 showingPlcHdr (Current Contents Are Placeholder Text) ... 533
17.5.2.40 storeMappedDataAs (Custom XML Data Date Storage Format) ... 534
17.5.2.41 tabIndex (Structured Document Tag Navigation Order Index) ... 536
17.5.2.42 tag (Programmatic Tag) ... 537
17.5.2.43 temporary (Remove Structured Document Tag When Contents Are Edited) ... 538
17.5.2.44 text (Plain Text Structured Document Tag) ... 539

17.6 Sections ... 540

17.6.1 bidi (Right to Left Section Layout) ... 541
17.6.2 bottom (Bottom Border) ... 542
17.6.3 col (Single Column Definition) ... 551
17.6.4 cols (Column Definitions) ... 552
17.6.5 docGrid (Document Grid) ... 554
17.6.6 formProt (Only Allow Editing of Form Fields) ... 559
17.6.7 left (Left Border) ... 559
17.6.8 lnNumType (Line Numbering Settings) ... 567
17.6.9 paperSrc (Paper Source Information) ... 569
17.6.10 pgBorders (Page Borders) ... 570
17.6.11 pgMar (Page Margins) ... 573
17.6.12 pgNumType (Page Numbering Settings) ... 577
17.6.13 pgSz (Page Size) ... 579
17.6.14 printerSettings (Reference to Printer Settings Data) ... 581
17.6.15 right (Right Border) ... 582
17.6.16 rtlGutter (Gutter on Right Side of Page) ... 589
17.6.17 sectPr (Document Final Section Properties) ... 590
17.6.18 sectPr (Section Properties) ... 592
17.6.19 sectPr (Previous Section Properties) ... 594
17.6.20 textDirection (Text Flow Direction) ... 595
17.6.21 top (Top Border) ... 596
17.6.22 type (Section Type) ... 605
17.6.23 vAlign (Vertical Text Alignment on Page) ... 606

17.7 Styles ... 607

17.7.1 Style Inheritance ... 607
17.7.2 Style Hierarchy ... 608
17.7.3 Toggle Properties ... 609
17.7.4 General Style Properties ... 611
17.7.4.1 aliases (Alternate Style Names) ... 612
17.7.4.2 autoRedefine (Automatically Merge User Formatting Into Style Definition) ... 613
17.7.4.3 basedOn (Parent Style ID) ... 615
17.7.4.4 hidden (Hide Style From User Interface) ... 617
17.7.4.5 latentStyles (Latent Style Information) ... 618
17.7.4.6 link (Linked Style Reference) ... 621
17.7.4.7 locked (Style Cannot Be Applied) ... 623
17.7.4.8 lsdException (Latent Style Exception) ... 624
17.7.4.9 name (Primary Style Name) ... 627
17.7.4.10 next (Style For Next Paragraph) ... 628

©ISO/IEC 2012 – All rights reserved

17.7.4.11 personal (E-Mail Message Text Style) ... 629
17.7.4.12 personalCompose (E-Mail Message Composition Style) ... 630
17.7.4.13 personalReply (E-Mail Message Reply Style) ... 630
17.7.4.14 qFormat (Primary Style) ... 631
17.7.4.15 rsid (Revision Identifier for Style Definition) ... 632
17.7.4.16 semiHidden (Hide Style From Main User Interface) ... 632
17.7.4.17 style (Style Definition) ... 633
17.7.4.18 styles (Style Definitions) ... 638
17.7.4.19 uiPriority (Optional User Interface Sorting Order) ... 639
17.7.4.20 unhideWhenUsed (Remove Semi-Hidden Property When Style Is Used) ... 640
17.7.5 Document Defaults ... 640
17.7.5.1 docDefaults (Document Default Paragraph and Run Properties) ... 641
17.7.5.2 pPr (Paragraph Properties) ... 642
17.7.5.3 pPrDefault (Default Paragraph Properties) ... 643
17.7.5.4 rPr (Run Properties) ... 643
17.7.5.5 rPrDefault (Default Run Properties) ... 644
17.7.6 Table Styles ... 644
17.7.6.1 pPr (Table Style Conditional Formatting Paragraph Properties) ... 648
17.7.6.2 rPr (Table Style Conditional Formatting Run Properties) ... 648
17.7.6.3 tblPr (Table Style Conditional Formatting Table Properties) ... 649
17.7.6.4 tblPr (Style Table Properties) ... 649
17.7.6.5 tblStyleColBandSize (Number of Columns in Column Band) ... 650
17.7.6.6 tblStylePr (Style Conditional Table Formatting Properties) ... 651
17.7.6.7 tblStyleRowBandSize (Number of Rows in Row Band) ... 653
17.7.6.8 tcPr (Table Style Conditional Formatting Table Cell Properties) ... 654
17.7.6.9 tcPr (Style Table Cell Properties) ... 655
17.7.6.10 trPr (Table Style Conditional Formatting Table Row Properties) ... 655
17.7.6.11 trPr (Style Table Row Properties) ... 656
17.7.7 Numbering Styles ... 656
17.7.8 Paragraph Styles ... 657
17.7.8.1 Numbering in Paragraph Styles ... 658
17.7.8.2 pPr (Style Paragraph Properties) ... 660
17.7.9 Run (Character) Styles ... 661
17.7.9.1 rPr (Run Properties) ... 662

17.8 Fonts ... 662

17.8.1 Font Embedding ... 663
17.8.2 Font Substitution ... 664
17.8.3 Elements ... 664
17.8.3.1 altName (Alternate Names for Font) ... 664
17.8.3.2 charset (Character Set Supported By Font) ... 666
17.8.3.3 embedBold (Bold Style Font Style Embedding) ... 666
17.8.3.4 embedBoldItalic (Bold Italic Font Style Embedding) ... 669
17.8.3.5 embedItalic (Italic Font Style Embedding) ... 671
17.8.3.6 embedRegular (Regular Font Style Embedding) ... 673
17.8.3.7 embedSystemFonts (Embed Common System Fonts) ... 675
17.8.3.8 embedTrueTypeFonts (Embed TrueType Fonts) ... 675
17.8.3.9 family (Font Family) ... 675

17.8.3.10 font (Properties for a Single Font) 676
17.8.3.11 fonts (Font Table Root Element) 677
17.8.3.12 notTrueType (Not a TrueType outline Font) 678
17.8.3.13 panose1 (Panose-1 Typeface Classification Number) 678
17.8.3.14 pitch (Font Pitch) 679
17.8.3.15 saveSubsetFonts (Subset Fonts When Embedding) 680
17.8.3.16 sig (Supported Unicode Subranges and Code Pages) 680

17.9 Numbering 684

17.9.1 abstractNum (Abstract Numbering Definition) 686
17.9.2 abstractNumId (Abstract Numbering Definition Reference) 687
17.9.3 ilvl (Numbering Level Reference) 689
17.9.4 isLgl (Display All Levels Using Arabic Numerals) 690
17.9.5 lvl (Numbering Level Override Definition) 691
17.9.6 lvl (Numbering Level Definition) 694
17.9.7 lvlJc (Justification) 697
17.9.8 lvlOverride (Numbering Level Definition Override) 698
17.9.9 lvlPicBulletId (Picture Numbering Symbol Definition Reference) 700
17.9.10 lvlRestart (Restart Numbering Level Symbol) 701
17.9.11 lvlText (Numbering Level Text) 704
17.9.12 multiLevelType (Abstract Numbering Definition Type) 705
17.9.13 name (Abstract Numbering Definition Name) 706
17.9.14 nsid (Abstract Numbering Definition Identifier) 707
17.9.15 num (Numbering Definition Instance) 708
17.9.16 numbering (Numbering Definitions) 711
17.9.17 numFmt (Numbering Format) 711
17.9.18 numId (Numbering Definition Instance Reference) 712
17.9.19 numIdMacAtCleanup (Last Reviewed Abstract Numbering Definition) 713
17.9.20 numPicBullet (Picture Numbering Symbol Definition) 714
17.9.21 numStyleLink (Numbering Style Reference) 715
17.9.22 pPr (Numbering Level Associated Paragraph Properties) 717
17.9.23 pStyle (Paragraph Style's Associated Numbering Level) 717
17.9.24 rPr (Numbering Symbol Run Properties) 719
17.9.25 start (Starting Value) 720
17.9.26 startOverride (Numbering Level Starting Value Override) 721
17.9.27 styleLink (Numbering Style Definition) 723
17.9.28 suff (Content Between Numbering Symbol and Paragraph Text) 724
17.9.29 tmpl (Numbering Template Code) 725

17.10 Headers and Footers 726

17.10.1 evenAndOddHeaders (Different Even/Odd Page Headers and Footers) 727
17.10.2 footerReference (Footer Reference) 728
17.10.3 ftr (Footer) 731
17.10.4 hdr (Header) 733
17.10.5 headerReference (Header Reference) 735
17.10.6 titlePg (Different First Page Headers and Footers) 738

17.11 Footnotes and Endnotes 739

17.11.1 continuationSeparator (Continuation Separator Mark) 741

17.11.2 endnote (Endnote Content) 742
17.11.3 endnote (Special Endnote List) 744
17.11.4 endnotePr (Document-Wide Endnote Properties) 745
17.11.5 endnotePr (Section-Wide Endnote Properties) 746
17.11.6 endnoteRef (Endnote Reference Mark) 747
17.11.7 endnoteReference (Endnote Reference) 749
17.11.8 endnotes (Document Endnotes) 751
17.11.9 footnote (Special Footnote List) 752
17.11.10 footnote (Footnote Content) 753
17.11.11 footnotePr (Section-Wide Footnote Properties) 755
17.11.12 footnotePr (Document-Wide Footnote Properties) 756
17.11.13 footnoteRef (Footnote Reference Mark) 757
17.11.14 footnoteReference (Footnote Reference) 759
17.11.15 footnotes (Document Footnotes) 761
17.11.16 noEndnote (Suppress Endnotes In Document) 762
17.11.17 numFmt (Endnote Numbering Format) 762
17.11.18 numFmt (Footnote Numbering Format) 764
17.11.19 numRestart (Footnote and Endnote Numbering Restart Location) 765
17.11.20 numStart (Footnote and Endnote Numbering Starting Value) 766
17.11.21 pos (Footnote Placement) 768
17.11.22 pos (Endnote Placement) 770
17.11.23 separator (Footnote/Endnote Separator Mark) 771

17.12 Glossary Document 772

17.12.1 behavior (Entry Insertion Behavior) 773
17.12.2 behaviors (Entry Insertion Behaviors) 776
17.12.3 category (Entry Categorization) 776
17.12.4 description (Description for Entry) 777
17.12.5 docPart (Glossary Document Entry) 778
17.12.6 docPartBody (Contents of Glossary Document Entry) 779
17.12.7 docPartPr (Glossary Document Entry Properties) 781
17.12.8 docParts (List of Glossary Document Entries) 781
17.12.9 gallery (Gallery Associated With Entry) 782
17.12.10 glossaryDocument (Glossary Document Root Element) 783
17.12.11 guid (Entry ID) 784
17.12.12 name (Category Associated With Entry) 784
17.12.13 name (Entry Name) 786
17.12.14 style (Associated Paragraph Style Name) 787
17.12.15 type (Entry Type) 788
17.12.16 types (Entry Types) 789

17.13 Annotations 790

17.13.1 Inline Annotations 790
17.13.2 "Cross Structure" Annotations 791
17.13.3 Property Annotations 792
17.13.4 Comments 793
17.13.4.1 annotationRef (Comment Information Block) 795
17.13.4.2 comment (Comment Content) 795
17.13.4.3 commentRangeEnd (Comment Anchor Range End) 798

17.13.4.4 commentRangeStart (Comment Anchor Range Start) ... 800
17.13.4.5 commentReference (Comment Content Reference Mark) ... 802
17.13.4.6 comments (Comments Collection) ... 804
17.13.5 Revisions ... 804
17.13.5.1 cellDel (Table Cell Deletion) ... 806
17.13.5.2 cellIns (Table Cell Insertion) ... 808
17.13.5.3 cellMerge (Vertically Merged/Split Table Cells) ... 811
17.13.5.4 customXmlDelRangeEnd (Custom XML Markup Deletion End) ... 815
17.13.5.5 customXmlDelRangeStart (Custom XML Markup Deletion Start) ... 817
17.13.5.6 customXmlInsRangeEnd (Custom XML Markup Insertion End) ... 819
17.13.5.7 customXmlInsRangeStart (Custom XML Markup Insertion Start) ... 821
17.13.5.8 customXmlMoveFromRangeEnd (Custom XML Markup Move Source End) ... 824
17.13.5.9 customXmlMoveFromRangeStart (Custom XML Markup Move Source Start) ... 826
17.13.5.10 customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End) ... 829
17.13.5.11 customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start) ... 831
17.13.5.12 del (Deleted Table Row) ... 834
17.13.5.13 del (Deleted Math Control Character) ... 836
17.13.5.14 del (Deleted Run Content) ... 837
17.13.5.15 del (Deleted Paragraph) ... 839
17.13.5.16 ins (Inserted Math Control Character) ... 842
17.13.5.17 ins (Inserted Table Row) ... 843
17.13.5.18 ins (Inserted Run Content) ... 845
17.13.5.19 ins (Inserted Numbering Properties) ... 847
17.13.5.20 ins (Inserted Paragraph) ... 850
17.13.5.21 moveFrom (Move Source Paragraph) ... 852
17.13.5.22 moveFrom (Move Source Run Content) ... 854
17.13.5.23 moveFromRangeEnd (Move Source Location Container - End) ... 856
17.13.5.24 moveFromRangeStart (Move Source Location Container - Start) ... 860
17.13.5.25 moveTo (Move Destination Run Content) ... 867
17.13.5.26 moveTo (Move Destination Paragraph) ... 870
17.13.5.27 moveToRangeEnd (Move Destination Location Container - End) ... 872
17.13.5.28 moveToRangeStart (Move Destination Location Container - Start) ... 875
17.13.5.29 pPrChange (Revision Information for Paragraph Properties) ... 882
17.13.5.30 rPrChange (Revision Information for Run Properties on the Paragraph Mark) ... 884
17.13.5.31 rPrChange (Revision Information for Run Properties) ... 886
17.13.5.32 sectPrChange (Revision Information for Section Properties) ... 888
17.13.5.33 tblGridChange (Revision Information for Table Grid Column Definitions) ... 890
17.13.5.34 tblPrChange (Revision Information for Table Properties) ... 891
17.13.5.35 tblPrExChange (Revision Information for Table-Level Property Exceptions) ... 893
17.13.5.36 tcPrChange (Revision Information for Table Cell Properties) ... 895
17.13.5.37 trPrChange (Revision Information for Table Row Properties) ... 897
17.13.6 Bookmarks ... 899
17.13.6.1 bookmarkEnd (Bookmark End) ... 900
17.13.6.2 bookmarkStart (Bookmark Start) ... 902
17.13.7 Range Permissions ... 909
17.13.7.1 permEnd (Range Permission End) ... 909
17.13.7.2 permStart (Range Permission Start) ... 912
17.13.8 Spelling and Grammar ... 918

17.13.8.1 proofErr (Proofing Error Anchor).. 919
17.14 Mail Merge 921
17.14.1 active (Record Is Included in Mail Merge)... 922
17.14.2 activeRecord (Record Currently Displayed In Merged Document) 923
17.14.3 addressFieldName (Column Containing E-mail Address)... 925
17.14.4 checkErrors (Mail Merge Error Reporting Setting)... 926
17.14.5 colDelim (Column Delimiter for Data Source) 927
17.14.6 column (Index of Column Being Mapped).. 928
17.14.7 column (Index of Column Containing Unique Values for Record)... 930
17.14.8 connectString (Data Source Connection String) 931
17.14.9 dataSource (Data Source File Path).. 932
17.14.10 dataType (Data Source Type) 934
17.14.11 destination (Merged Document Destination) 935
17.14.12 doNotSuppressBlankLines (Remove Blank Lines from Merged Documents)............................... 936
17.14.13 dynamicAddress (Use Country-Based Address Field Ordering) 936
17.14.14 fHdr (First Row of Data Source Contains Column Names) 937
17.14.15 fieldMapData (External Data Source to Merge Field Mapping) 938
17.14.16 headerSource (Header Definition File Path)... 938
17.14.17 lid (Merge Field Name Language ID) 940
17.14.18 linkToQuery (Query Contains Link to External Query File).. 941
17.14.19 mailAsAttachment (Merged Document To E-Mail Attachment)... 941
17.14.20 mailMerge (Mail Merge Settings).. 942
17.14.21 mailSubject (Merged E-mail or Fax Subject Line)... 943
17.14.22 mainDocumentType (Source Document Type) 944
17.14.23 mappedName (Predefined Merge Field Name) 945
17.14.24 name (Data Source Name for Column) 947
17.14.25 odso (Office Data Source Object Settings) 948
17.14.26 query (Query For Data Source Records To Merge) 949
17.14.27 recipientData (Data About Single Data Source Record)... 950
17.14.28 recipientData (Reference to Inclusion/Exclusion Data for Data Source) 951
17.14.29 recipients (Inclusion/Exclusion Data for Data Source) 953
17.14.30 src (ODSO Data Source File Path) 954
17.14.31 table (Data Source Table Name)... 955
17.14.32 type (ODSO Data Source Type).. 957
17.14.33 type (Merge Field Mapping) 957
17.14.34 udl (UDL Connection String) 958
17.14.35 uniqueTag (Unique Value for Record) 960
17.14.36 viewMergedData (View Merged Data Within Document) 960
17.15 Settings...961
17.15.1 Document Settings 962
17.15.1.1 activeWritingStyle (Grammar Checking Settings) 962
17.15.1.2 alignBordersAndEdges (Align Paragraph and Table Borders with Page Border)....................... 964
17.15.1.3 alwaysMergeEmptyNamespace (Do Not Mark Custom XML Elements With No Namespace As Invalid) 965
17.15.1.4 alwaysShowPlaceholderText (Use Custom XML Element Names as Default Placeholder Text)966
17.15.1.5 attachedSchema (Attached Custom XML Schema) 967
17.15.1.6 attachedTemplate (Attached Document Template) 968

17.15.1.7 autoCaption (Single Automatic Captioning Setting) 969
17.15.1.8 autoCaptions (Automatic Captioning Settings) 971
17.15.1.9 autoFormatOverride (Allow Automatic Formatting to Override Formatting Protection Settings) 972
17.15.1.10 autoHyphenation (Automatically Hyphenate Document Contents When Displayed) 972
17.15.1.11 bookFoldPrinting (Book Fold Printing) 973
17.15.1.12 bookFoldPrintingSheets (Number of Pages Per Booklet) 975
17.15.1.13 bookFoldRevPrinting (Reverse Book Fold Printing) 977
17.15.1.14 bordersDoNotSurroundFooter (Page Border Excludes Footer) 979
17.15.1.15 bordersDoNotSurroundHeader (Page Border Excludes Header) 979
17.15.1.16 caption (Single Caption Type Definition) 980
17.15.1.17 captions (Caption Settings) 989
17.15.1.18 characterSpacingControl (Character-Level Whitespace Compression) 992
17.15.1.19 clickAndTypeStyle (Paragraph Style Applied to Automatically Generated Paragraphs) 992
17.15.1.20 clrSchemeMapping (Theme Color Mappings) 994
17.15.1.21 compat (Compatibility Settings) 999
17.15.1.22 consecutiveHyphenLimit (Maximum Number of Consecutively Hyphenated Lines) 999
17.15.1.23 decimalSymbol (Radix Point for Field Code Evaluation) 1000
17.15.1.24 defaultTableStyle (Default Table Style for Newly Inserted Tables) 1002
17.15.1.25 defaultTabStop (Distance Between Automatic Tab Stops) 1003
17.15.1.26 displayBackgroundShape (Display Background Objects When Displaying Document) 1004
17.15.1.27 displayHorizontalDrawingGridEvery (Distance between Horizontal Gridlines) 1005
17.15.1.28 displayVerticalDrawingGridEvery (Distance between Vertical Gridlines) 1006
17.15.1.29 documentProtection (Document Editing Restrictions) 1008
17.15.1.30 documentType (Document Classification) 1013
17.15.1.31 docVar (Single Document Variable) 1014
17.15.1.32 docVars (Document Variables) 1015
17.15.1.33 doNotAutoCompressPictures (Do Not Automatically Compress Images) 1015
17.15.1.34 doNotDemarcateInvalidXml (Do Not Show Visual Indicator For Invalid Custom XML Markup) 1016
17.15.1.35 doNotDisplayPageBoundaries (Do Not Display Visual Boundary For Header/Footer or Between Pages) 1016
17.15.1.36 doNotEmbedSmartTags (Remove Smart Tags When Saving) 1018
17.15.1.37 doNotHyphenateCaps (Do Not Hyphenate Words in ALL CAPITAL LETTERS) 1018
17.15.1.38 doNotIncludeSubdocsInStats (Do Not Include Content in Text Boxes, Footnotes, and Endnotes in Document Statistics) 1019
17.15.1.39 doNotShadeFormData (Do Not Show Visual Indicator For Form Fields) 1020
17.15.1.40 doNotTrackFormatting (Do Not Track Formatting Revisions When Tracking Revisions) 1020
17.15.1.41 doNotTrackMoves (Do Not Use Move Syntax When Tracking Revisions) 1022
17.15.1.42 doNotUseMarginsForDrawingGridOrigin (Do Not Use Margins for Drawing Grid Origin) .. 1023
17.15.1.43 doNotValidateAgainstSchema (Do Not Validate Custom XML Markup Against Schemas). 1023
17.15.1.44 drawingGridHorizontalOrigin (Drawing Grid Horizontal Origin Point) 1023
17.15.1.45 drawingGridHorizontalSpacing (Drawing Grid Horizontal Grid Unit Size) 1025
17.15.1.46 drawingGridVerticalOrigin (Drawing Grid Vertical Origin Point) 1026
17.15.1.47 drawingGridVerticalSpacing (Drawing Grid Vertical Grid Unit Size) 1027
17.15.1.48 forceUpgrade (Upgrade Document on Open) 1029
17.15.1.49 formsDesign (Structured Document Tag Placeholder Text Should be Resaved) 1029
17.15.1.50 gutterAtTop (Position Gutter At Top of Page) 1030

17.15.1.51 hideGrammaticalErrors (Do Not Display Visual Indication of Grammatical Errors)........... 1031
17.15.1.52 hideSpellingErrors (Do Not Display Visual Indication of Spelling Errors) 1031
17.15.1.53 hyphenationZone (Hyphenation Zone) 1031
17.15.1.54 ignoreMixedContent (Ignore Mixed Content When Validating Custom XML Markup) 1033
17.15.1.55 linkStyles (Automatically Update Styles From Document Template)................................ 1034
17.15.1.56 listSeparator (List Separator for Field Code Evaluation) 1035
17.15.1.57 mirrorMargins (Mirror Page Margins)... 1036
17.15.1.58 noLineBreaksAfter (Custom Set of Characters Which Cannot End a Line)......................... 1037
17.15.1.59 noLineBreaksBefore (Custom Set Of Characters Which Cannot Begin A Line) 1038
17.15.1.60 noPunctuationKerning (Never Kern Punctuation Characters) 1040
17.15.1.61 printFormsData (Only Print Form Field Content) 1040
17.15.1.62 printFractionalCharacterWidth (Print Fractional Character Widths) 1042
17.15.1.63 printPostScriptOverText (Print PostScript Codes With Document Text)............................ 1042
17.15.1.64 printTwoOnOne (Print Two Pages Per Sheet) 1042
17.15.1.65 proofState (Spelling and Grammatical Checking State) 1044
17.15.1.66 readModeInkLockDown (Freeze Document Layout).. 1046
17.15.1.67 removeDateAndTime (Remove Date and Time from Annotations) 1048
17.15.1.68 removePersonalInformation (Remove Personal Information from Document Properties) 1049
17.15.1.69 revisionView (Visibility of Annotation Types)... 1049
17.15.1.70 rsid (Single Session Revision Save ID) 1051
17.15.1.71 rsidRoot (Original Document Revision Save ID) 1052
17.15.1.72 rsids (Listing of All Revision Save ID Values)... 1053
17.15.1.73 saveFormsData (Only Save Form Field Content) 1054
17.15.1.74 saveInvalidXml (Allow Saving Document As XML File When Custom XML Markup Is Invalid) 1055
17.15.1.75 savePreviewPicture (Generate Thumbnail For Document On Save)................................. 1055
17.15.1.76 saveThroughXslt (Custom XSL Transform To Use When Saving As XML File) 1056
17.15.1.77 saveXmlDataOnly (Only Save Custom XML Markup) 1057
17.15.1.78 settings (Document Settings) 1058
17.15.1.79 showEnvelope (Show E-Mail Message Header) 1058
17.15.1.80 showXMLTags (Show Visual Indicators for Custom XML Markup Start/End Locations) 1059
17.15.1.81 smartTagType (Supplementary Smart Tag Information).. 1059
17.15.1.82 strictFirstAndLastChars (Use Strict Kinsoku Rules for Japanese Text)............................... 1061
17.15.1.83 styleLockQFSet (Prevent Replacement of Styles Part) 1061
17.15.1.84 styleLockTheme (Prevent Modification of Themes Part) 1061
17.15.1.85 stylePaneFormatFilter (Suggested Filtering for List of Document Styles).......................... 1062
17.15.1.86 stylePaneSortMethod (Suggested Sorting for List of Document Styles)............................ 1064
17.15.1.87 summaryLength (Percentage of Document to Use When Generating Summary) 1064
17.15.1.88 themeFontLang (Theme Font Languages)... 1065
17.15.1.89 trackRevisions (Track Revisions to Document).. 1068
17.15.1.90 updateFields (Automatically Recalculate Fields on Open) 1070
17.15.1.91 useXSLTWhenSaving (Save Document as XML File through Custom XSL Transform)......... 1071
17.15.1.92 view (Document View Setting) 1071
17.15.1.93 writeProtection (Write Protection) 1072
17.15.1.94 zoom (Magnification Setting) 1076
17.15.2 Web Page Settings.. 1078
17.15.2.1 allowPNG (Allow PNG as Graphic Format) 1078
17.15.2.2 blockQuote (Data for HTML blockquote Element)... 1079

17.15.2.3 bodyDiv (Data for HTML body Element).. 1080
17.15.2.4 bottom (Bottom Border for HTML div).. 1081
17.15.2.5 color (Frameset Splitter Color) 1082
17.15.2.6 div (Information About Single HTML div Element).. 1085
17.15.2.7 divBdr (Set of Borders for HTML div).. 1088
17.15.2.8 divs (Information about HTML div Elements).. 1089
17.15.2.9 divsChild (Child div Elements Contained within Current div).. 1091
17.15.2.10 doNotOrganizeInFolder (Do Not Place Supporting Files in Subdirectory) 1093
17.15.2.11 doNotRelyOnCSS (Do Not Rely on CSS for Font Face Formatting) 1093
17.15.2.12 doNotSaveAsSingleFile (Recommend Web Page Format over Single File Web Page Format) 1094
17.15.2.13 doNotUseLongFileNames (Do Not Use File Names Longer than 8.3 Characters) 1095
17.15.2.14 encoding (Output Encoding When Saving as Web Page) 1095
17.15.2.15 flatBorders (Frameset Splitter Border Style) 1096
17.15.2.16 frame (Single Frame Properties).. 1097
17.15.2.17 frameLayout (Frameset Layout) 1099
17.15.2.18 frameset (Nested Frameset Definition).. 1100
17.15.2.19 frameset (Root Frameset Definition).. 1102
17.15.2.20 framesetSplitbar (Frameset Splitter Properties) 1103
17.15.2.21 left (Left Border for HTML div) 1104
17.15.2.22 linkedToFile (Maintain Link to Existing File) 1105
17.15.2.23 longDesc (Frame Long Description).. 1106
17.15.2.24 marBottom (Bottom Margin for HTML div).. 1109
17.15.2.25 marH (Top and Bottom Margin for Frame) 1110
17.15.2.26 marLeft (Left Margin for HTML div).. 1112
17.15.2.27 marRight (Right Margin for HTML div) 1113
17.15.2.28 marTop (Top Margin for HTML div).. 1115
17.15.2.29 marW (Left and Right Margin for Frame) 1117
17.15.2.30 name (Frame Name).. 1118
17.15.2.31 noBorder (Do Not Display Frameset Splitters) 1120
17.15.2.32 noResizeAllowed (Frame Cannot Be Resized) 1122
17.15.2.33 optimizeForBrowser (Disable Features Not Supported by Target Web Profile) 1123
17.15.2.34 pixelsPerInch (Pixels per Inch for Graphics/Images) 1124
17.15.2.35 right (Right Border for HTML div).. 1125
17.15.2.36 saveSmartTagsAsXml (Save Smart Tag Data in XML Property Bag) 1126
17.15.2.37 scrollbar (Scrollbar Display Option).. 1127
17.15.2.38 sourceFileName (Source File for Frame) 1129
17.15.2.39 sz (Frame Size) 1131
17.15.2.40 sz (Nested Frameset Size).. 1133
17.15.2.41 targetScreenSz (Target Screen Size for Web Page) 1135
17.15.2.42 title (Frame or Frameset Title).. 1135
17.15.2.43 top (Top Border for HTML div) 1137
17.15.2.44 w (Frameset Splitter Width) 1138
17.15.2.45 webSettings (Web Page Settings).. 1140
17.15.3 Language Compatibility Settings 1140
17.15.3.1 adjustLineHeightInTable (Add Document Grid Line Pitch To Lines in Table Cells).. 1141
17.15.3.2 applyBreakingRules (Use Legacy Ethiopic and Amharic Line Breaking Rules).. 1142
17.15.3.3 balanceSingleByteDoubleByteWidth (Balance Single Byte and Double Byte Characters)...... 1143

17.15.3.4 compatSetting (Custom Compatibility Setting) ... 1144
17.15.3.5 doNotExpandShiftReturn (Don't Justify Lines Ending in Soft Line Break) ... 1146
17.15.3.6 doNotLeaveBackslashAlone (Display Backslash As Yen Sign) ... 1146
17.15.3.7 spaceForUL (Add Additional Space Below Baseline For Underlined East Asian Text) ... 1147
17.15.3.8 ulTrailSpace (Underline All Trailing Spaces) ... 1148

17.16 Fields and Hyperlinks ... 1149

17.16.1 Syntax ... 1150
17.16.2 XML representation ... 1157
17.16.3 Formulas and expressions ... 1159
17.16.3.1 Constants ... 1160
17.16.3.2 Bookmarks ... 1160
17.16.3.3 Operators ... 1160
17.16.3.4 Functions ... 1161
17.16.3.5 Table cell references ... 1162
17.16.4 Field formatting ... 1163
17.16.4.1 Date and time formatting ... 1163
17.16.4.2 Numeric formatting ... 1179
17.16.4.3 General formatting ... 1181
17.16.5 Field definitions ... 1190
17.16.5.1 ADDRESSBLOCK ... 1192
17.16.5.2 ADVANCE ... 1194
17.16.5.3 ASK ... 1195
17.16.5.4 AUTHOR ... 1196
17.16.5.5 AUTOTEXT ... 1197
17.16.5.6 AUTOTEXTLIST ... 1197
17.16.5.7 BIBLIOGRAPHY ... 1198
17.16.5.8 CITATION ... 1199
17.16.5.9 COMMENTS ... 1200
17.16.5.10 COMPARE ... 1201
17.16.5.11 CREATEDATE ... 1202
17.16.5.12 DATABASE ... 1203
17.16.5.13 DATE ... 1204
17.16.5.14 DOCPROPERTY ... 1205
17.16.5.15 DOCVARIABLE ... 1207
17.16.5.16 EDITTIME ... 1207
17.16.5.17 FILENAME ... 1208
17.16.5.18 FILESIZE ... 1208
17.16.5.19 FILLIN ... 1209
17.16.5.20 FORMCHECKBOX ... 1210
17.16.5.21 FORMDROPDOWN ... 1210
17.16.5.22 FORMTEXT ... 1211
17.16.5.23 GOTOBUTTON ... 1211
17.16.5.24 GREETINGLINE ... 1212
17.16.5.25 HYPERLINK ... 1212
17.16.5.26 IF ... 1213
17.16.5.27 INCLUDEPICTURE ... 1214
17.16.5.28 INCLUDETEXT ... 1214

©ISO/IEC 2012 – All rights reserved

17.16.5.29 INDEX 1216
17.16.5.30 KEYWORDS 1218
17.16.5.31 LASTSAVEDBY 1219
17.16.5.32 LINK 1219
17.16.5.33 LISTNUM 1220
17.16.5.34 MACROBUTTON 1224
17.16.5.35 MERGEFIELD 1225
17.16.5.36 MERGEREC 1226
17.16.5.37 MERGESEQ 1226
17.16.5.38 NEXT 1227
17.16.5.39 NEXTIF 1227
17.16.5.40 NOTEREF 1228
17.16.5.41 NUMCHARS 1228
17.16.5.42 NUMPAGES 1229
17.16.5.43 NUMWORDS 1229
17.16.5.44 PAGE 1230
17.16.5.45 PAGEREF 1230
17.16.5.46 PRINT 1231
17.16.5.47 PRINTDATE 1232
17.16.5.48 PRIVATE 1232
17.16.5.49 QUOTE 1233
17.16.5.50 RD 1233
17.16.5.51 REF 1234
17.16.5.52 REVNUM 1235
17.16.5.53 SAVEDATE 1235
17.16.5.54 SECTION 1236
17.16.5.55 SECTIONPAGES 1237
17.16.5.56 SEQ 1237
17.16.5.57 SET 1239
17.16.5.58 SKIPIF 1239
17.16.5.59 STYLEREF 1240
17.16.5.60 SUBJECT 1241
17.16.5.61 SYMBOL 1242
17.16.5.62 TA 1243
17.16.5.63 TC 1244
17.16.5.64 TEMPLATE 1245
17.16.5.65 TIME 1245
17.16.5.66 TITLE 1246
17.16.5.67 TOA 1246
17.16.5.68 TOC 1247
17.16.5.69 USERADDRESS 1249
17.16.5.70 USERINITIALS 1250
17.16.5.71 USERNAME 1251
17.16.5.72 XE 1251
17.16.6 calcOnExit (Recalculate Fields When Current Field Is Modified) 1253
17.16.7 checkBox (Checkbox Form Field Properties) 1254
17.16.8 checked (Checkbox Form Field State) 1254
17.16.9 ddList (Drop-Down List Form Field Properties) 1255

17.16.10 default (Default Text Box Form Field String) 1255
17.16.11 default (Default Drop-Down List Item Index) 1257
17.16.12 default (Default Checkbox Form Field State) 1258
17.16.13 delInstrText (Deleted Field Code) 1258
17.16.14 enabled (Form Field Enabled) 1259
17.16.15 entryMacro (Script Function to Execute on Form Field Entry) 1260
17.16.16 exitMacro (Script Function to Execute on Form Field Exit) 1261
17.16.17 ffData (Form Field Properties) 1262
17.16.18 fldChar (Complex Field Character) 1263
17.16.19 fldSimple (Simple Field) 1267
17.16.20 format (Text Box Form Field Formatting) 1269
17.16.21 helpText (Associated Help Text) 1270
17.16.22 hyperlink (Hyperlink) 1272
17.16.23 instrText (Field Code) 1275
17.16.24 label (Form Field Label) 1276
17.16.25 listEntry (Drop-Down List Entry) 1278
17.16.26 maxLength (Text Box Form Field Maximum Length) 1279
17.16.27 name (Form Field Name) 1280
17.16.28 result (Drop-Down List Selection) 1281
17.16.29 size (Checkbox Form Field Size) 1283
17.16.30 sizeAuto (Automatically Size Form Field) 1283
17.16.31 statusText (Associated Status Text) 1284
17.16.32 tabIndex (Form Field Navigation Order Index) 1286
17.16.33 textInput (Text Box Form Field Properties) 1287
17.16.34 type (Text Box Form Field Type) 1287

17.17 Miscellaneous Topics 1288

17.17.1 Subdocuments 1288
17.17.1.1 subDoc (Anchor for Subdocument Location) 1290
17.17.2 Alternative Format Import 1293
17.17.2.1 altChunk (Anchor for Imported External Content) 1294
17.17.2.2 altChunkPr (External Content Import Properties) 1295
17.17.2.3 matchSrc (Keep Source Formatting on Import) 1296
17.17.3 Roundtripping Alternate Content 1297
17.17.4 Boolean Property (CT_OnOff) 1299

17.18 Simple Types 1299

17.18.1 ST_AnnotationVMerge (Table Cell Vertical Merge Revision Type) 1299
17.18.2 ST_Border (Border Styles) 1300
17.18.3 ST_BrClear (Line Break Text Wrapping Restart Location) 1353
17.18.4 ST_BrType (Break Types) 1355
17.18.5 ST_CaptionPos (Automatic Caption Positioning Values) 1356
17.18.6 ST_ChapterSep (Chapter Separator Types) 1358
17.18.7 ST_CharacterSpacing (Character-Level Whitespace Compression Settings) 1359
17.18.8 ST_CombineBrackets (Two Lines in One Enclosing Character Type) 1359
17.18.9 ST_DateTime (Standard Date and Time Storage Format) 1360
17.18.10 ST_DecimalNumber (Decimal Number Value) 1361
17.18.11 ST_DecimalNumberOrPercent (Percentage Measurement) 1361
17.18.12 ST_Direction (Bidirectional Direction Types) 1362

17.18.13 ST_DisplacedByCustomXml (Location of Custom XML Markup Displacing an Annotation) 1362
17.18.14 ST_DocGrid (Document Grid Types) 1363
17.18.15 ST_DocPartBehavior (Insertion Behavior Types) 1365
17.18.16 ST_DocPartGallery (Entry Gallery Types) 1366
17.18.17 ST_DocPartType (Entry Types) 1369
17.18.18 ST_DocProtect (Document Protection Types) 1371
17.18.19 ST_DocType (Document Classification Values) 1372
17.18.20 ST_DropCap (Text Frame Drop Cap Location) 1372
17.18.21 ST_EdGrp (Range Permision Editing Group) 1373
17.18.22 ST_EdnPos (Endnote Positioning Location) 1374
17.18.23 ST_EighthPointMeasure (Measurement in Eighths of a Point) 1375
17.18.24 ST_Em (Emphasis Mark Type) 1375
17.18.25 ST_FFHelpTextVal (Help Text Value) 1376
17.18.26 ST_FFName (Form Field Name Value) 1377
17.18.27 ST_FFStatusTextVal (Status Text Value) 1378
17.18.28 ST_FFTextType (Text Box Form Field Type Values) 1379
17.18.29 ST_FldCharType (Complex Field Character Type) 1380
17.18.30 ST_FontFamily (Font Family Value) 1380
17.18.31 ST_FrameLayout (Frameset Layout Order) 1381
17.18.32 ST_FrameScrollbar (Frame Scrollbar Visibility) 1383
17.18.33 ST_FtnEdn (Footnote or Endnote Type) 1384
17.18.34 ST_FtnPos (Footnote Positioning Location) 1386
17.18.35 ST_HAnchor (Horizontal Anchor Location) 1387
17.18.36 ST_HdrFtr (Header or Footer Type) 1388
17.18.37 ST_HeightRule (Height Rule) 1389
17.18.38 ST_HexColor (Color Value) 1390
17.18.39 ST_HexColorAuto ('Automatic' Color Value) 1390
17.18.40 ST_HighlightColor (Text Highlight Colors) 1391
17.18.41 ST_Hint (Font Type Hint) 1393
17.18.42 ST_HpsMeasure (Measurement in Half-Points) 1394
17.18.43 ST_InfoTextType (Help or Status Text Type) 1394
17.18.44 ST_Jc (Horizontal Alignment Type) 1395
17.18.45 ST_JcTable (Table Alignment Type) 1398
17.18.46 ST_LevelSuffix (Content Between Numbering Symbol and Paragraph Text) 1399
17.18.47 ST_LineNumberRestart (Line Numbering Restart Position) 1400
17.18.48 ST_LineSpacingRule (Line Spacing Rule) 1401
17.18.49 ST_Lock (Locking Types) 1402
17.18.50 ST_LongHexNumber (Eight Digit Hexadecimal Value) 1403
17.18.51 ST_MacroName (Script Subroutine Name Value) 1403
17.18.52 ST_MailMergeDataType (Mail Merge Data Source Type Values) 1404
17.18.53 ST_MailMergeDest (Merged Document Destination Types) 1405
17.18.54 ST_MailMergeDocType (Source Document Types) 1406
17.18.55 ST_MailMergeOdsoFMDFieldType (Merge Field Mapping Types) 1407
17.18.56 ST_MailMergeSourceType (Mail Merge ODSO Data Source Types) 1407
17.18.57 ST_Merge (Merged Cell Type) 1409
17.18.58 ST_MultiLevelType (Numbering Definition Type) 1411
17.18.59 ST_NumberFormat (Numbering Format) 1412
17.18.60 ST_ObjectDrawAspect (Embedded Object Representations) 1455

17.18.61 ST_ObjectUpdateMode (Embedded Object Update Modes) 1455
17.18.62 ST_PageBorderDisplay (Page Border Display Options) 1455
17.18.63 ST_PageBorderOffset (Page Border Positioning Base) 1456
17.18.64 ST_PageBorderZOrder (Page Border Z-Order) 1457
17.18.65 ST_PageOrientation (Page Orientation) 1458
17.18.66 ST_Pitch (Font Pitch Value) 1459
17.18.67 ST_PixelsMeasure (Measurement in Pixels) 1459
17.18.68 ST_PointMeasure (Measurement in Points) 1459
17.18.69 ST_Proof (Proofing State Values) 1460
17.18.70 ST_ProofErr (Proofing Error Type) 1460
17.18.71 ST_PTabAlignment (Absolute Position Tab Alignment) 1461
17.18.72 ST_PTabLeader (Absolute Position Tab Leader Character) 1462
17.18.73 ST_PTabRelativeTo (Absolute Position Tab Positioning Base) 1464
17.18.74 ST_RestartNumber (Footnote/Endnote Numbering Restart Locations) 1464
17.18.75 ST_RubyAlign (Phonetic Guide Text Alignment) 1465
17.18.76 ST_SdtDateMappingType (Date Storage Format Types) 1467
17.18.77 ST_SectionMark (Section Type) 1468
17.18.78 ST_Shd (Shading Patterns) 1471
17.18.79 ST_ShortHexNumber (Four Digit Hexadecimal Value) 1485
17.18.80 ST_SignedHpsMeasure (Signed Measurement in Half-Points) 1486
17.18.81 ST_SignedTwipsMeasure (Signed Measurement in Twentieths of a Point) 1486
17.18.82 ST_StyleSort (Style Sort Settings) 1487
17.18.83 ST_StyleType (Style Types) 1488
17.18.84 ST_TabJc (Custom Tab Stop Type) 1489
17.18.85 ST_TabTlc (Custom Tab Stop Leader Character) 1490
17.18.86 ST_TargetScreenSz (Target Screen Sizes for Generated Web Pages) 1492
17.18.87 ST_TblLayoutType (Table Layout Type) 1493
17.18.88 ST_TblOverlap (Table Overlap Setting) 1496
17.18.89 ST_TblStyleOverrideType (Conditional Table Style Formatting Types) 1498
17.18.90 ST_TblWidth (Table Width Units) 1499
17.18.91 ST_TextAlignment (Vertical Text Alignment Types) 1500
17.18.92 ST_TextboxTightWrap (Lines To Tight Wrap Within Text Box) 1501
17.18.93 ST_TextDirection (Text Flow Direction) 1502
17.18.94 ST_TextEffect (Animated Text Effects) 1504
17.18.95 ST_TextScale (Text Expansion/Compression Percentage) 1505
17.18.96 ST_Theme (Theme Font) 1505
17.18.97 ST_ThemeColor (Theme Color) 1506
17.18.98 ST_UcharHexNumber (Two Digit Hexadecimal Value) 1508
17.18.99 ST_Underline (Underline Patterns) 1508
17.18.100 ST_VAnchor (Vertical Anchor Location) 1512
17.18.101 ST_VerticalJc (Vertical Alignment Type) 1513
17.18.102 ST_View (Document View Values) 1514
17.18.103 ST_WmlColorSchemeIndex (Theme Color Reference) 1515
17.18.104 ST_Wrap (Text Wrapping around Text Frame Type) 1516
17.18.105 ST_Zoom (Magnification Preset Values) 1518
17.18.106 ST_TextScalePercent (Text Expansion/Compression Percentage) 1518
17.18.107 ST_MeasurementOrPercent (Measurement or Percentage Value) 1519

End of informative text.

17.2 Main Document Story

As defined in §4, a WordprocessingML document contains the markup for a rendition of an Office Open XML document of category Wordprocessing. Syntactically, the document consists of a compilation of two kinds of information, which are combined to create this rendition:

- Properties [*Example*: styles, numbering definitions, etc. *end example*]
- Stories [*Example*: main document, comments, headers, etc. *end example*]

In WordprocessingML, *stories* are unique containers for one or more paragraphs, as defined by the parent elements of the p element (§17.3.1.22). Stories contain the document's content. The properties of the document are applied to the contents of each story to create the rendition. Most of the content in a WordprocessingML document is located in the main document story, which is stored inside the body element within the Main Document part (§11.3.10).

[*Example*: Consider a document with a single paragraph in the main document story. This document would require the following WordprocessingML in its main document part:

```
<w:document>
  <w:body>
    <w:p/>
  </w:body>
</w:document>
```

The fact that the paragraph is inside the body element makes it part of the main document story. *end example*]

17.2.1 background (Document Background)

This element specifies the background for every page of the document containing the background element. A document's *background* is the image or fill for the entire page surface, behind all other document content.

The drawing §17.3.3.9 child elements of the background element allows any DrawingML effect to be applied to the document's background.

For solid color fill backgrounds, however, the attributes on this element allow the use of any RGB or theme color value (the latter a reference to the document's themes part).

[*Example*: Consider a document which utilizes a gradient fill background moving between black and the accent3 theme color, as follows:

This background would require the following WordprocessingML markup:

```
<w:background w:themeColor="accent3">

  <w:drawing>
    <wp:inline … >
      <a:graphic>
        <a:graphicData … >
          …
        </a:graphicData>
      </a:graphic>
    </wp:inline>
  </w:drawing></w:background>
```

The resulting background consists of a single color fill of the `accent3` theme color from the themeColor attribute, layered under a `gradientCenter` fill. *end example*]

Attributes	Description
color (Background Color)	Specifies the color for the background of the document. This value can be defined as either: • A color value using the RGB color model whose red, green, and blue values are written as numbers in the range 0 to 255, hex encoded, and concatenated. [*Example*: Full intensity red would be 255 red, 0 green, 0 blue, encoded to FF, 00, 00, and concatenated to `FF0000`. *end example*]. RGB colors are specified in the sRGB color space. • `auto` to allow a consumer to automatically determine the background color in order to make the document's text readable. [*Example*: A document with white text and a background color of `auto` might result in the use of a black background, in order to ensure legibility of the content. *end example*]

Attributes	Description
	If the background specifies the use of a theme color via the themeColor attribute, this value is ignored. [*Note*: Applications are discouraged from specifying both the color and themeColor attributes on the same parent element. *end note*] [*Example*: Consider a background color with value `2C34FF`, as follows: `<w:background … w:color="2C34FF"/>` The background color is therefore the color with RGB value 44,52,255 (the decimal decoding of the hex value above). *end example*] The possible values for this attribute are defined by the ST_HexColor simple type (§17.18.38).

themeColor (Background Theme Color)	Specifies the base theme color used to generate the background color. The background color is the RGB value associated with themeColor as further transformed by themeTint or themeShade (if one is present), else the background color is the RGB value associated with themeColor.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part (§14.2.7 and §20.1.6.9), which allows color information to be set centrally in the document.
	
	If the color attribute is specified, its value shall be ignored in favor of the color resulting from the use of this attribute with any appropriate themeTint and themeShade attribute value calculations applied.
	
	To determine the color to display, the following actions are performed:
	• Using the mapping specified in the ST_ThemeColor simple type (§17.18.97), the appropriate attribute on the clrSchemeMapping element (§17.15.1.20) is read.
	• Using that value and the mapping specified in the ST_ColorSchemeIndex simple type (§17.18.103), the appropriate element in the document's Theme part is read to get the base theme color.
	• The specified color is modified based on the presence of the themeTint or themeShade attribute.
	
	[*Example*: Consider a background configured to use the `accent5` theme color, resulting in the following WordprocessingML markup:
	
	`<w:background w:themeColor="accent5" />`
	
	If the Settings part contained the following markup:
	
	`<w:clrSchemeMapping … w:accent5="accent5"/>`
	
	and the Theme part contained the following XML markup:

Attributes	Description

	`  <a:accent5>`
	`    <a:srgbClr val="BCBCBC"/>`
	`  </a:accent5>`
	
	the resulting background color would be `BCBCBC`. *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§17.18.97).

themeShade (Background Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this background. If the themeColor attribute is not specified, this attribute shall not be specified. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this background. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current background. [*Example*: Consider a shade of 60% applied to a background in a document. This shade is calculated as follows: $\begin{aligned} S_{xml} &= 0.6 * 255 \\ &= 153 \\ &= 99(hex) \end{aligned}$ The resulting themeShade value in the file format would be `99`. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48{,}0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $\begin{aligned} L' &= 0.53 * 0.75 \\ &= 0.39698 \end{aligned}$

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48{,}0.39698\right)$and converting back to RGB, we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:background w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).
themeTint (Background Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this background. If the themeColor attribute is not specified, this attribute shall not be specified. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to the document's background. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current background. [*Example*: Consider a tint of 60% applied to a background in a document. This tint is calculated as follows: $$\begin{aligned} T_{xml} &= 0.4 * 255 \\ &= 102 \\ &= 66(hex) \end{aligned}$$ The resulting themeTint value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: - Convert the color to the HSL color format (values from 0 to 1) - Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ - Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45{,}0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get:

Attributes	Description
	$$L' = 0.53 * 0.6 + (1 - .6) \\ = 0.71$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get 95B3D7. This transformed value can be seen in the resulting background's color attribute: `<w:background w:color="95B3D7" w:themeColor="accent2" w:themeTint="99"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Background) is located in §A.1. *end note*]

17.2.2 body (Document Body)

This element specifies the contents of the body of the document - the main document editing surface.

The document body contains what is referred to as *block-level markup* - markup which can exist as a sibling element to paragraphs in a WordprocessingML document.

[*Example*: Consider a document with a single paragraph in the main document story. This document would require the following WordprocessingML in its main document part:

```
<w:document>
  <w:body>
    <w:p/>
  </w:body>
</w:document>
```

The fact that the paragraph is inside the body element makes it part of the main document story. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Body) is located in §A.1. *end note*]

17.2.3 document (Document)

This element specifies the contents of a main document part in a WordprocessingML document.

[*Example*: Consider the basic structure of the main document part in a basic WordprocessingML document, as follows:

```
<w:document>
  <w:body>
    <w:p/>
  </w:body>
</w:document>
```

All of the contents of the main document part are contained beneath the document element. *end example*]

Attributes	Description

conformance (Document Conformance Class)	Specifies the conformance class (§2.1) to which the WordprocessingML document conforms.
	If this attribute is omitted, its default value is `transitional`.
	[*Example*: Consider the following WordprocessingML Main Document part markup:
	`<w:document w:conformance="strict">`
	`  …`
	`</w:document>`
	This document has a conformance attribute value of `strict`, therefore it conforms to the WML Strict conformance class. *end example*]
	The possible values for this attribute are defined by the ST_ConformanceClass simple type (§22.9.2.2).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Document) is located in §A.1. *end note*]

17.3 Paragraphs and Rich Formatting

The basis of a WordprocessingML document is its actual text contents. Those text contents can be stored in many contexts (tables, text boxes, etc.), but the most basic form of text contents in WordprocessingML is the paragraph, specified using the p element (§17.3.1.22).

Within the paragraph, all rich formatting at the paragraph level is stored within the pPr element (§17.3.1.25; §17.3.1.26). [*Note*: Some examples of paragraph properties are alignment, border, hyphenation override, indentation, line spacing, shading, text direction, and widow/orphan control. *end note*]

Within the paragraph, text is grouped into one or more runs, represented by the r element (§17.3.2.25), which define a region of text with a common set of properties.

Just as a paragraph can have rich formatting, so too can a run. All of the elements inside an r element have their properties controlled by a corresponding optional rPr run properties element (§17.7.9.1; §17.3.2.27). [*Note*: Some examples of run properties are bold, underlined, or visible. *end note*]

Within runs, run content is the set of possible objects and characters which can be displayed in the document.

17.3.1 Paragraphs

The most basic unit of block-level content within a WordprocessingML document, *paragraphs* are stored using the p element (§17.3.1.22). A paragraph defines a distinct division of content with a WordprocessingML document which begins on a new line.

[*Example*: Consider the paragraph fragment "*The quick brown fox jumped …* " which is centered on a paragraph. The justification property is a paragraph level property, and therefore is expressed on the paragraph properties as follows:

```
<w:p>
  <w:pPr>
    <w:jc w:val="center"/>
    <w:rPr>
      <w:i/>
    </w:rPr>
  </w:pPr>
  <w:r>
    <w:rPr>
      <w:i/>
    </w:rPr>
    <w:t xml:space="preserve">The quick brown fox jumped … </w:t>
  </w:r>
</w:p>
```

Notice that each run specifies the character formatting information for its contents, and the paragraph specifies the paragraph level formatting (the center-justification). It is also notable that since leading and trailing whitespace is not normally significant in XML; some runs require an attribute specifying that their whitespace is significant via the xml:space element. *end example*]

A paragraph's properties are specified via the pPr element (§17.3.1.25; §17.3.1.26). [*Note*: Some examples of paragraph properties are alignment, border, hyphenation override, indentation, line spacing, shading, text direction, and widow/orphan control. *end note*]

17.3.1.1 adjustRightInd (Automatically Adjust Right Indent When Using Document Grid)

This element specifies whether the right indent shall be automatically adjusted for the given paragraph when a document grid has been defined for the current section using the docGrid element (§17.6.5), modifying of the current right indent used on this paragraph.

[*Note*: This setting is used in order to ensure that the line breaking for that paragraph is not determined by the width of the final character on the line. *end note*]

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, its value is assumed to be `true`.

[*Example*: Consider a paragraph in which the right indent on the current paragraph should not be automatically determined based on the character pitch set in the document grid. This setting would be specified using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    …
    <w:adjustRightInd w:val="false" />
  </w:pPr>
  …
</w:p>
```

By explicitly setting the val to `false`, this paragraph uses its specified right indent settings regardless of the presence of the document grid for the parent section. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.2 autoSpaceDE (Automatically Adjust Spacing of Latin and East Asian Text)

This element specifies whether inter-character spacing shall automatically be adjusted between regions of Latin text and regions of East Asian text in the current paragraph. These regions shall be determined by the Unicode character values of the text content within the paragraph.

[*Note*: This property is used to ensure that the spacing between regions of Latin text and adjoining East Asian text is sufficient on each side such that the Latin text can be easily read within the East Asian text. *end note*]

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, its value is assumed to be `true`.

[*Example*: Consider a paragraph in which the spacing should not be automatically adjusted based on the presence of Latin and East Asian text. This setting would be specified using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    …
    <w:autoSpaceDE w:val="false" />
  </w:pPr>
  …
```

```
</w:p>
```

By explicitly setting the val to `false`, this paragraph must not automatically adjust the spacing of adjoining Latin and East Asian text. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.3 autoSpaceDN (Automatically Adjust Spacing of East Asian Text and Numbers)

This element specifies whether inter-character spacing shall automatically be adjusted between regions of numbers and regions of East Asian text in the current paragraph. These regions shall be determined by the Unicode character values of the text content within the paragraph.

[*Note*: This property is used to ensure that the spacing between regions of numbers and adjoining East Asian text is sufficient on each side such that the numbers can be easily read within the East Asian text. *end note*]

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, its value is assumed to be `true`.

[*Example*: Consider a paragraph in which the spacing should not be automatically adjusted based on the presence of numbers and East Asian text. This setting would be specified using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    …
    <w:autoSpaceDN w:val="false" />
  </w:pPr>
  …
</w:p>
```

By explicitly setting the val to `false`, this paragraph automatically adjusts the spacing of adjoining numbers and East Asian text. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.4 bar (Paragraph Border Between Facing Pages)

This element specifies the border which can be displayed on the inside edge of the paragraph when the parent's section settings specify that the section shall be printed using mirrored margins using the mirrorMargins element (§17.15.1.57). [*Note*: This information is present in the WordprocessingML for the purposes of legacy document format compatibility, and it can be removed and/or ignored as required. *end note*]

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no bar border shall be applied to the current paragraph.

[*Example*: Consider the following paragraph's WordprocessingML definition for its paragraph borders:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
        w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
        w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
        w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
        w:themeColor="accent3" w:themeTint="99" />
      <w:bar w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
        w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>Sample paragraph.</w:t>
  </w:r>
</w:p>
```

This paragraph has a single line bar border as defined by the bar element. *end example*]

This element’s content model is defined by the common border properties definition in §17.3.4.

17.3.1.5 between (Paragraph Border Between Identical Paragraphs)

This element specifies the border which shall be displayed between each paragraph in a set of paragraphs which have the same set of paragraph border settings.

To determine if any two adjoining paragraphs should have a between border or an individual top and bottom border, the set of borders on the two adjoining paragraphs are compared. If the border information on those two paragraphs is identical for all possible paragraphs borders, then the between border is displayed. Otherwise, each paragraph shall use its bottom and top border, respectively. If this border specifies a space attribute, that value is ignored - this border is always located at the bottom of each paragraph with an identical following paragraph, taking into account any space after the line pitch.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no between border shall be applied between identical paragraphs.

[*Example*: Consider the following two paragraphs' WordprocessingML definition:

```
<w:p>
  <w:pPr>
```

```
      <w:pBdr>
        <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
  w:themeColor="accent2" w:themeTint="33" />
        <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
  w:themeColor="accent6" w:themeShade="BF" />
        <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
  w:themeColor="accent2" w:themeTint="33" />
        <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
  w:themeColor="accent3" w:themeTint="99" />
        <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
  w:themeColor="accent1" />
      </w:pBdr>
    </w:pPr>
    <w:r>
      <w:t>First paragraph.</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:pPr>
      <w:pBdr>
        <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
  w:themeColor="accent2" w:themeTint="33" />
        <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
  w:themeColor="accent6" w:themeShade="BF" />
        <w:bottom w:val="single" w:sz="24" w:space="0" w:color="F2DCDB"
  w:themeColor="accent2" w:themeTint="33" />
        <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
  w:themeColor="accent3" w:themeTint="99" />
        <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
  w:themeColor="accent1" />
      </w:pBdr>
    </w:pPr>
    <w:r>
      <w:t>Second paragraph.</w:t>
    </w:r>
  </w:p>
```

Since the bottom paragraph border is different between the two paragraphs (the bottom space value goes from 1 to 0), these paragraphs do not use the between border, and instead paragraph one uses its bottom border, and paragraph two uses its top border. If those values were identical, then paragraph one would have a between border below it, and paragraph two would have no top border. *end example*]

This element’s content model is defined by the common border properties definition in §17.3.4.

17.3.1.6 bidi (Right to Left Paragraph Layout)

This element specifies that this paragraph shall be displayed from right to left. This property only affects the following set of paragraph-level properties:

- ind (§17.3.1.12)
- jc (§17.3.1.13)
- tab (§17.3.1.37)
- textDirection (§17.3.1.41)

This setting alone does not affect the ordering of text within the paragraph – see the rtl element (§17.3.2.30) for a detailed description.

This element specifies that the base direction of the text within the paragraph is right-to-left (see HL1 in the Unicode Standard Annex #9). See also Part 1, §I.2.

[*Example*: Consider a paragraph with the bidi property set as follows:

```
<w:p>
    <w:pPr>
      <w:bidi/>
    </w:pPr>
    …
</w:p>
```

This paragraph direction is now right to left, which means that all paragraph properties are displayed right to left (e.g. the paragraph marker glyph (if any) is displayed on the right, and indentation for the first line of the paragraph occurs on the right side of the page). *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.7 bottom (Paragraph Border Below Identical Paragraphs)

This element specifies the border which shall be displayed below a set of paragraphs which have the same paragraph border settings.

To determine if any two adjoining paragraphs shall have an individual top and bottom border or a between border, the set of borders on the two adjoining paragraphs are compared. If the border information on those two paragraphs is different, then the first paragraph shall use its bottom border and the following paragraph shall use its top border. Otherwise, the between border is used. If this border specifies a space attribute, that value determines the space after the bottom of the text (ignoring any space below) which should be left before this border is drawn, specified in points.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no between border shall be applied below identical paragraphs.

[*Example*: Consider the following two paragraphs' WordprocessingML definition:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB" 
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034" 
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB" 
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B" 
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD" 
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>First paragraph.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB" 
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034" 
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="0" w:color="F2DCDB" 
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B" 
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD" 
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>Second paragraph.</w:t>
  </w:r>
</w:p>
```

Since the paragraph border is different between the two paragraphs (the bottom space value goes from 1 to 0), paragraph one uses its bottom border, which is located one point below the text in that paragraph. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.3.1.8 cnfStyle (Paragraph Conditional Formatting)

This element specifies the set of conditional table style formatting properties which have been applied to this paragraph, if this paragraph is contained within a table cell. [*Note*: This property is an optimization which can be used by consumers to determine if a given property on a paragraph is the result of the table style properties vs. direct formatting on the paragraph itself. *end note*]

If this property is specified on a paragraph which is not contained within a table cell, then its contents shall be ignored when reading the contents of the document.

[*Example*: Consider a paragraph in the top right corner of a table with a table style applied and where the table is formatted as left to right. This paragraph would need to specify the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:cnfStyle w:firstRow="true" w:lastColumn="true"
w:firstRowLastColumn="true" />
    …
  </w:pPr>
  …
</w:p>
```

This paragraph specifies that it has the conditional properties from the table style for the first column, first row, and the top right corner of the parent table by setting the appropriate attributes. *end example*]

Attributes	Description

evenHBand (Even Numbered Horizontal Band)	Specifies that the object has inherited the conditional properties applied to the even numbered horizontal bands of the parent object.
	[*Example*: Consider a paragraph in the second row of a table with a table style applied, and where the band width is one row. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:evenHBand="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the even numbered horizontal bands of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

Attributes	Description

evenVBand (Even Numbered Vertical Band)	Specifies that the object has inherited the conditional properties applied to the even numbered vertical bands of the parent object.
	[*Example*: Consider a paragraph in the second column of a table with a table style applied, and where the band width is one column. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:evenVBand="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the even numbered vertical bands of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstColumn (First Column)	Specifies that the object has inherited the conditional properties applied to the first column of the parent object.
	[*Example*: Consider a paragraph in the first column of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:firstColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the first column of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstRow (First Row)	Specifies that the object has inherited the conditional properties applied to the first row of the parent object.
	[*Example*: Consider a paragraph in the top row of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:firstRow="true" />`

Attributes	Description

	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the first row of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstRowFirstColumn (First Row and First Column)	Specifies that the object has inherited the conditional properties applied to the cell that is in the first row and first column of the parent object.
	
	[*Example*: Consider a paragraph in the first row and first column of a table. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:firstRow="true" w:firstColumn="true"`
	`w:firstRowFirstColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the cell in the first row and first column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstRowLastColumn (First Row and Last Column)	Specifies that the object has inherited the conditional properties applied to the cell that is in the first row and last column of the parent object.
	
	[*Example*: Consider a paragraph in the first row and last column of a table. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:firstRow="true" w:lastColumn="true"`
	`w:firstRowLastColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the cell in the first row and last column of the parent table. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastColumn (Last Column)	Specifies that the object has inherited the conditional properties applied to the last column of the parent object.
	[*Example*: Consider a paragraph in the last column of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	  `<w:pPr>`
	    `<w:cnfStyle w:lastColumn="true" />`
	    …
	  `</w:pPr>`
	  …
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the last column of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastRow (Last Row)	Specifies that the object has inherited the conditional properties applied to the last row of the parent object.
	[*Example*: Consider a paragraph in the bottom row of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	  `<w:pPr>`
	    `<w:cnfStyle w:lastRow="true" />`
	    …
	  `</w:pPr>`
	  …
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the last row of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastRowFirstColumn (Last Row and First Column)	Specifies that the object has inherited the conditional properties applied to the cell that is in the last row and first column of the parent object.
	[*Example*: Consider a paragraph in the last row and first column of a table. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	  `<w:pPr>`
	    `<w:cnfStyle w:lastRow="true" w:firstColumn="true"`

Attributes	Description

	`w:lastRowFirstColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the cell in the last row and first column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastRowLastColumn (Last Row and Last Column)	Specifies that the object has inherited the conditional properties applied to the cell that is in the last row and last column of the parent object.
	
	[*Example*: Consider a paragraph in the last row and last column of a table. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:lastRow="true" w:lastColumn="true"`
	`w:lastRowLastColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the cell in the last row and last column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

oddHBand (Odd Numbered Horizontal Band)	Specifies that the object has inherited the conditional properties applied to the odd numbered horizontal bands of the parent object.
	
	[*Example*: Consider a paragraph in the third row of a table with a table style applied, and where the band width is one column. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:oddHBand="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the odd numbered horizontal bands of the parent table. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

oddVBand (Odd Numbered Vertical Band)	Specifies that the object has inherited the conditional properties applied to the odd numbered vertical bands of the parent object.
	[*Example*: Consider a paragraph in the third column of a table with a table style applied, and where the band width is one column. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:oddVBand="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the odd numbered vertical bands of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Cnf) is located in §A.1. *end note*]

17.3.1.9 contextualSpacing (Ignore Spacing Above and Below When Using Identical Styles)

This element specifies that any space specified before or after this paragraph, specified using the spacing element (§17.3.1.33), should not be applied when the preceding and following paragraphs are of the same paragraph style, affecting the top and bottom spacing respectively. [*Example*: This value is typically used for paragraphs in lists, in which any space between subsequent list items, even if inherited from another style, is not desirable. *end example*]

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then spacing is not ignored. If it is present, then the spacing above or below on this paragraph is subtracted from the spacing which would have been present if contextual spacing was off, never going below zero.

[*Example*: Consider two paragraphs defined as follows:

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="TestParagraphStyle" />
    <w:spacing w:after="200"/>
```

```
      <w:contextualSpacing/>
    </w:pPr>
    …
  </w:p>
  <w:p>
    <w:pPr>
      <w:pStyle w:val="TestParagraphStyle" />
      <w:spacing w:before="240"/>
    </w:pPr>
    …
  </w:p>
```

The first paragraph specifies a spacing after of 10 points, and the second paragraph specifies a spacing before of 12 points, therefore according to the rules on the spacing element, the net paragraph spacing should be 12 points. However, since the first paragraph specifies that its spacing should be omitted between paragraphs of the same style, and the two paragraphs use the same TestParagraphStyle, that value is subtracted from the total, therefore the paragraphs are spaced by 2 points. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.10 divId (Associated HTML div ID)

This element specifies that this paragraph should be located within the specified HTML div tag when this document is saved in HTML format. This ID is then used to look up the associated div stored in the divs (§17.15.2.8) element. [*Note*: This element is used to preserve the fidelity of existing HTML documents when saved in the WordprocessingML format. *end note*].

If the paragraph does not specify this element, then any div referenced by the previous paragraph is closed, and this paragraph shall not belong to any div when saved as HTML. If this specified id does not exist in the collection of divs the current document, then any div referenced by the previous paragraph is closed, and this paragraph shall not belong to any div when saved as HTML.

[*Example*: Consider the following WordprocessingML paragraph fragment:

```
  <w:p>
    <w:pPr>
      <w:divId w:val="1512645511" />
    </w:pPr>
  </w:p>
```

This paragraph specifies that it belongs to the HTML div with id 1512645511, stored in the divs element. *end example*]

Attributes	Description
val (Decimal	Specifies that the contents of this attribute contains a decimal number.

Attributes	Description
Number Value)	The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.3.1.11 framePr (Text Frame Properties)

This element specifies information about the current paragraph with regard to *text frames*. *Text frames* are paragraphs of text in a document which are positioned in a separate region or frame in the document, and can be positioned with a specific size and position relative to non-frame paragraphs in the current document.

The first piece of information specified by the framePr element is that the current paragraph is actually part of a text frame in the document. This information is specified simply by the presence of the framePr element in paragraph's properties. If the framePr element is omitted, the paragraph shall not be part of any text frame in the document.

The second piece of information concerns the set of paragraphs which are part of the current text frame in the document. This is determined based on the attributes on the framePr element. If the set of attribute values specified on two adjacent paragraphs is identical, then those two paragraphs shall be considered to be part of the same text frame and rendered within the same frame in the document.

[*Example*: Consider a document in which the following two paragraphs are located adjacent to one another:

```
<w:p>
  <w:pPr>
    <w:framePr w:w="2191" w:h="811" w:hRule="exact" w:hSpace="180"
w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1921"/>
  </w:pPr>
  <w:r>
    <w:t>Paragraph One</w:t>
  </w:r>
```

```
</w:p>
<w:p>
  <w:pPr>
    <w:framePr w:w="2191" w:h="810" w:hRule="exact" w:hSpace="180"
w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1921"/>
  </w:pPr>
  <w:r>
    <w:t>Paragraph Two.</w:t>
  </w:r>
</w:p>
```

These two paragraphs, although each is a part of a text frame due to the presence of the framePr element, are different text frames because of the differing h value - `810` vs. `811`. *end example*]

The positioning of the frame relative to the properties stored on its attribute values shall be calculated relative to the next paragraphs in the document which is itself not part of a text frame.

[*Example*: Consider a document in which the following three paragraphs are located adjacent to one another:

```
<w:p>
  <w:pPr>
    <w:framePr w:w="2191" w:h="811" w:hRule="exact" w:hSpace="180"
w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1921"/>
  </w:pPr>
  <w:r>
    <w:t>Paragraph One</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:framePr w:w="2191" w:h="811" w:hRule="exact" w:hSpace="180"
w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1921"/>
  </w:pPr>
  <w:r>
    <w:t>Paragraph Two.</w:t>
  </w:r>
</w:p>
<w:p/>
```

The first two paragraphs form a single text frame, which is anchored using its attribute values relative to the first non-frame paragraph following it (the third paragraph in the example). *end example*]

Attributes	Description
anchorLock (Lock	Specifies that the frame shall always remain in the same logical position relative to the

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>Frame Anchor to Paragraph)</td><td>non-frame paragraphs which precede and follow it in this document.

This means that consumers which modify this document shall ensure that this text frame remains directly above the non-frame paragraph which it is currently above, by adjusting the frame's positioning properties as needed as the paragraph is moved throughout the document rather than moving the frame's logical location within the paragraphs in the document, if that would be more appropriate.

If this attribute is omitted, then this frame shall not have a locked anchor position.

[<i>Example</i>: Consider the following WordprocessingML paragraph contained in a text frame:
<pre>
<w:p>
  <w:pPr>
    <w:framePr w:w="2419" w:h="2189" w:hRule="exact"
w:hSpace="187" w:wrap="around" w:vAnchor="text"
w:hAnchor="page" w:x="1643" w:y="73" w:anchorLock="1"/>
  </w:pPr>
  <w:r>
    <w:t>Text Frame Content.</w:t>
  </w:r>
</w:p>
</pre>This text frame has a locked anchor using the anchorLock attribute. If the text frame is moved down in the document, the text frame properties must be adjusted to be relative to the parent paragraph's same logical position - the paragraph cannot be relocated in the document, which results in changes to the frame's properties as follows:
<pre>
<w:p>
  <w:pPr>
    <w:framePr w:w="2419" w:h="2189" w:hRule="exact"
w:hSpace="187" w:wrap="around" w:vAnchor="text"
w:hAnchor="page" w:x="1643" w:y="-5247" w:anchorLock="1"/>
  </w:pPr>
  <w:r>
    <w:t>Text Frame Content.</w:t>
  </w:r>
</w:p>
</pre>The non-frame paragraph was relocated 5320 twentieths of a point below its original location in the document, and the frame's vertical positioning properties were adjusted to ensure its logical location within the paragraph ordering was constant while its visual location was changed. <i>end example</i>]

The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>dropCap (Drop Cap Frame)</td><td>

Specifies that the current frame contains a drop cap to be located at the beginning of the next non-frame paragraph in the document. Its contents shall be used to specify how that drop cap should be positioned relative to that paragraph.

If this attribute is omitted, then this frame shall not be considered a drop cap frame.

[*Note*: Although a drop cap is simply a text frame, this element is used to determine how the cap should be positioned relative to the following non-frame paragraph in relative terms (see possible values), rather than relying on absolute sizing. *end note*]

[*Example*: Consider the following paragraph containing a text frame which should be positioned as a drop cap:

```
  <w:p>
    <w:pPr>
      <w:framePr w:dropCap="margin" w:lines="3" w:hSpace="432"
  w:wrap="around" w:vAnchor="text" w:hAnchor="page" />
    </w:pPr>
    <w:r>
      <w:t>A</w:t>
    </w:r>
  </w:p>
```

The dropCap attribute specifies a value of `margin`, so this drop cap is placed outside of the text margin before the start of the current text. *end example*]

The possible values for this attribute are defined by the ST_DropCap simple type (§17.18.20).

</td></tr>
<tr><td>h (Frame Height)</td><td>

Specifies the frame's height.

This height is expressed in twentieths of a point.

If this attribute is omitted, then its value shall be assumed to be `0`.

The meaning of the value of the h attribute is defined based on the value of the hRule attribute for this text frame as follows:

- If the value of hRule is `auto`, then the frame's height should be automatically determined based on the height of its contents. This value is ignored.
- If the value of hRule is `atLeast`, then the frame's height should be at least the value of this attribute.
- If the value of hRule is `exact`, then the frame's height should be exactly the value of this attribute.

[*Example*: Consider the following paragraph containing a text frame:

```
  <w:p>
```

</td></tr>
</table>

Attributes	Description

	`  <w:pPr>`
	`    <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast" w:hSpace="187" w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1643" w:y="73" />`
	`  </w:pPr>`
	`  <w:r>`
	`    <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	The h attribute specifies a value of 2189 twentieths of a point, so this text frame is a minimum of 2189 twentieths of a point high regardless of its contents, since its hRule value is set to atLeast. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

hAnchor (Frame Horizontal Positioning Base)	Specifies the base object from which the horizontal positioning in the x attribute should be calculated.
	
	A text frame can be horizontally positioned relative to:
	• The vertical edge of the page before any runs of text (the left edge for left-to-right paragraphs, the right edge for right-to-left paragraphs)
	• The vertical edge of the text margin before any runs of text (the left edge for left-to-right paragraphs, the right edge for right-to-left paragraphs)
	• The vertical edge of the text margin for the column in which the anchor paragraph is located
	
	If this attribute is omitted, then its value shall be assumed to be page.
	
	[*Example*: Consider a text frame which should be positioned one inch to the right of its column in a left-to-right document. This text frame would be specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:framePr … w:x="1440" w:hAnchor="column" />`
	`</w:pPr>`
	
	These frame properties specify that they are relative to the anchor paragraph's column, and that relative to that column, the frame should be 1440 twentieths of a point in the direction of the flow of text (right, in this case). *end example*]
	
	The possible values for this attribute are defined by the ST_HAnchor simple type (§17.18.35).

hRule (Frame Height Type)	Specifies the meaning of the height specified for this frame. The meaning of the value of the h attribute is defined based on the value of the hRule

Attributes	Description

	attribute for this text frame as follows:
	• If the value of hRule is `auto`, then the frame's height should be automatically determined based on the height of its contents. The h value is ignored.
	• If the value of hRule is `atLeast`, then the frame's height should be at least the value the h attribute.
	• If the value of hRule is `exact`, then the frame's height should be exactly the value of the h attribute.
	
	If this attribute is omitted, then its value shall be assumed to be `auto`.
	
	[*Example*: Consider the following paragraph containing a text frame:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast"`
	`w:hSpace="187" w:wrap="around" w:vAnchor="text"`
	`w:hAnchor="page" w:x="1643" w:y="73" />`
	`  </w:pPr>`
	`  <w:r>`
	`    <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	The h attribute specifies a value of 2189 twentieths of a point, so this text frame is a minimum of 2189 twentieths of a point high regardless of its contents, since its hRule value is set to `atLeast`. *end example*]
	
	The possible values for this attribute are defined by the ST_HeightRule simple type (§17.18.37).

hSpace (Horizontal Frame Padding)	Specifies the minimum distance which shall be maintained between the current text frame and any non-frame text which has been allowed to flow around this object when the wrap attribute on this text frame is set to around.
	
	This distance is expressed in twentieths of a point.
	
	If the wrap value is not set to around, this value shall be ignored. If this attribute is omitted, its value shall be assumed to be `0`.
	
	[*Example*: Consider a text frame which should have a minimum of a one-half inch spacing from any non-frame text on its left and right sides. This constraint would be specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:framePr … w:hSpace="720" w:wrap="around" />`
	`</w:pPr>`

Attributes	Description
	The wrap value of around allows text to wrap around this text frame, and the hSpace attribute specifies that the spacing between text and this frame must be a minimum of 720 twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

lines (Drop Cap Vertical Height in Lines)	Specifies the number of lines in the non-frame paragraph to which this text frame is anchored which should be used to calculate the drop cap's height.
	If the current frame is not a drop cap (the parent framePr element does not have the dropCap attribute), this value is ignored. If the current text frame is a dropped cap and this attribute is present, then any other vertical positioning information shall be ignored.
	If this attribute is omitted, then its value shall be considered to be 1.
	[*Example*: Consider the following paragraph containing a text frame which should be positioned as a drop cap:
	`<w:p>`
	`  <w:pPr>`
	`    <w:framePr w:dropCap="margin" w:lines="3" w:hSpace="432" w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:y="400" w:yAlign="text" />`
	`  </w:pPr>`
	`  <w:r>`
	`    <w:t>O</w:t>`
	`  </w:r>`
	`</w:p>`
	Since this frame is being used as a dropped cap, the y and yAlign attributes are ignored and the height of the drop cap is the first three lines of the anchor paragraph. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

vAnchor (Frame Vertical Positioning Base)	Specifies the base object from which the horizontal positioning in the y attribute should be calculated. A text frame can be horizontally positioned relative to: • The horizontal edge of the page before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections) • The horizontal edge of the text margin before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections) • The horizontal edge of the page before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections)

Attributes	Description

	If this attribute is omitted, then its value shall be assumed to be page.
	[*Example*: Consider a text frame which should be positioned two inches below the page top in a top-to-bottom document. This text frame would be specified using the following WordprocessingML:
	`<w:pPr>`
	`  <w:framePr … w:y="2880" w:vAnchor="page" />`
	`</w:pPr>`
	These frame properties specify that they are relative to the anchor page, and that relative to that column, the frame should be `2880` twentieths of a point in the direction of the flow of text (down, in this case). *end example*]
	The possible values for this attribute are defined by the ST_VAnchor simple type (§17.18.100).

vSpace (Vertical Frame Padding)	Specifies the minimum distance which shall be maintained between the current text frame and any non-frame text which is above or below this text frame.
	This distance is expressed in twentieths of a point.
	If this attribute is omitted, its value shall be assumed to be `0`.
	[*Example*: Consider a text frame which should have a minimum of a one-half inch spacing from any non-frame text on its top and bottom sides. This constraint would be specified using the following WordprocessingML:
	`<w:pPr>`
	`  <w:framePr … w:vSpace="720" />`
	`</w:pPr>`
	The vspace attribute specifies that the spacing between text and this frame must be a minimum of `720` twentieths of a point. *end example*]
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

w (Frame Width)	Specifies the exact value for this text frame's width. This value is specified in twentieths of a point. When this attribute is present, the text frame shall be rendered to the exact width specified. If this attribute is omitted, the text frame width shall be automatically determined by the maximum line width of the content within the text frame. [*Example*: Consider the following WordprocessingML fragment specifying a text frame:

Attributes	Description

	`<w:p>`
	`  <w:pPr>`
	`    <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast"`
	`w:hSpace="187" w:wrap="around" w:vAnchor="text"`
	`w:hAnchor="page" w:x="1643" w:y="73" />`
	`  </w:pPr>`
	`  <w:r>`
	`    <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	This text frame specifies that its width must be exactly 2419 twips. If this attribute was removed, the text frame would be rendered at the width of the content `Text Frame Content`. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

wrap (Text Wrapping Around Frame)	Specifies the style of text wrapping which should be allowed around the contents of this text frame. This attribute determines if non-frame text shall be allowed to flow around the contents of this frame.
	
	If this attribute is omitted, its value shall be assumed to be `around`.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a text frame:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast"`
	`w:hSpace="187" w:wrap="around" w:vAnchor="text"`
	`w:hAnchor="page" w:x="1643" w:y="73" />`
	`  </w:pPr>`
	`  <w:r>`
	`    <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	This text frame specifies that when the frame is rendered on the page, any non-text frame paragraphs which would normally flow onto the same lines must be allowed to do so. *end example*]
	
	The possible values for this attribute are defined by the ST_Wrap simple type (§17.18.104).

x (Absolute Horizontal Position)	Specifies an absolute horizontal position for the text frame. This absolute position is specified relative to the horizontal anchor specified by the hAnchor attribute for this text frame.

Attributes	Description

	This value is expressed in twentieths of a point. If it is positive, then the text frame is positioned after the anchor object in the direction of horizontal text flow in this document. If it is negative, then the text frame is positioned before the anchor object in the direction of horizontal text flow in this document.
	
	If the xAlign attribute is also specified, then this value is ignored. If this attribute is omitted, then its value shall be assumed to be `0`.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a text frame:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast"`
	`w:hSpace="187" w:wrap="around" w:vAnchor="text"`
	`w:hAnchor="page" w:x="1643" w:y="73" />`
	`  </w:pPr>`
	`  <w:r>`
	`    <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	This text frame specifies that it should be located exactly 1643 twentieths of a point after the vertical edge of the page (from the hAnchor attribute). *end example*]
	
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

xAlign (Relative Horizontal Position)	Specifies a relative horizontal position for the text frame. This relative position is specified relative to the horizontal anchor specified by the hAnchor attribute for this text frame.
	
	If omitted, this attribute is not specified and the value of the x attribute determines the absolute horizontal position of the text frame. If specified, the position for this attribute supersede any value which is specified in the x attribute, and that value is ignored.
	
	[*Example*: Consider the following WordprocessingML fragment specifying a text frame:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast"`
	`w:hSpace="187" w:wrap="around" w:vAnchor="text"`
	`w:hAnchor="page" w:x="1643" w:xAlign="left" w:y="73" />`
	`  </w:pPr>`
	`  <w:r>`
	`    <w:t>Text Frame Content.</w:t>`
	`  </w:r>`
	`</w:p>`

Attributes	Description
	This text frame specifies that it has a horizontal placement of exactly 1643 twentieths of a point relative to the page, but that exact placement is overridden by the presence of the xAlign attribute to place the frame on the left side of the page. *end example*] The possible values for this attribute are defined by the ST_XAlign simple type (§22.9.2.18).

y (Absolute Vertical Position)	Specifies an absolute vertical position for the text frame. This absolute position is specified relative to the vertical anchor specified by the vAnchor attribute for this text frame.
	This value is expressed in twentieths of a point. If it is positive, then the text frame is positioned after the anchor object in the direction of vertical text flow in this document. If it is negative, then the text frame is positioned before the anchor object in the direction of vertical text flow in this document.
	If the yAlign attribute is also specified, then this value is ignored. If this attribute is omitted, then its value shall be assumed to be `0`.
	[*Example*: Consider the following WordprocessingML fragment specifying a text frame:
	`<w:p>`
	`<w:pPr>`
	`<w:framePr w:w="2419" w:h="2189" w:hRule="atLeast" w:hSpace="187" w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1643" w:y="73" />`
	`</w:pPr>`
	`<w:r>`
	`<w:t>Text Frame Content.</w:t>`
	`</w:r>`
	`</w:p>`
	This text frame specifies that it should be located exactly 79 twentieths of a point below the top vertical edge of the anchor's paragraph's text (from the vAnchor attribute), assuming that the vertical text direction is top to bottom. *end example*]
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

yAlign (Relative Vertical Position)	Specifies a relative vertical position for the text frame. This relative position is specified relative to the vertical anchor specified by the vAnchor attribute for this text frame. If omitted, this attribute is not specified and the value of the y attribute determines the absolute horizontal position of the text frame. If specified, the position for this attribute supersedes any value which is specified in the y attribute, and that value is ignored, unless the vAnchor is set to `text`, in which case any relative positioning is not allowed, and is itself ignored.

©ISO/IEC 2012 – All rights reserved

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

[*Example*: Consider the following WordprocessingML fragment specifying a text frame:

```
<w:p>
  <w:pPr>
    <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast"
w:hSpace="187" w:wrap="around" w:vAnchor="margin"
w:hAnchor="page" w:x="1643" w:y="73" w:yAlign="center" />
  </w:pPr>
  <w:r>
    <w:t>Text Frame Content.</w:t>
  </w:r>
</w:p>
```

This text frame specifies that it has a vertical placement of exactly `73` twentieths of a point relative to the top margin, but that exact placement is overridden by the presence of the yAlign attribute to place the frame in the center of the margin. *end example*]

The possible values for this attribute are defined by the ST_YAlign simple type (§22.9.2.20).

</td></tr>
</table>

[*Note*: The W3C XML Schema definition of this element's content model (CT_FramePr) is located in §A.1. *end note*]

17.3.1.12 ind (Paragraph Indentation)

This element specifies the set of indentation properties applied to the current paragraph.

Indentation settings are overriden on an individual basis - if any single attribute on this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If any single attribute on this element is never specified in the style hierarchy, then no indentation of that indentation type is applied to the paragraph.

[*Example*: Consider a paragraph which should have a one inch indentation from the text margins on both the left and the right sides, except for the first line in each paragraph, which should only be indented one quarter of an inch from the text margin (on the side which begins the flow of text for this paragraph). This set of indentations is specified using the following WordprocessingML:

```
<w:pPr>
  <w:ind w:start="1440" w:end="1440" w:hanging="1080" />
</w:pPr>
```

This set of indentation properties specifies that a `1440` twentieths of a point indentation should be provided on both the left and the right side of the text margins for this paragraph, and that a `1080` twentieths of a point hanging indent (towards the text margin) should be applied to the text in the first paragraph, giving it a net one-quarter inch indent from the text margin. *end example*]

Attributes	Description

end (End Indentation)	Specifies the indentation which shall be placed at the end of this paragraph – between the right text margin for this paragraph and the right edge of that paragraph's content in a left to right paragraph, and the left text margin and the left edge of that paragraph's text in a right to left paragraph. If the mirrorIndents property (§17.3.1.18) is specified for this paragraph, then this indent is used for the outside page edge - the left page edge for odd numbered pages and the right page edge for even numbered pages.
	
	If this attribute is omitted, its value shall be assumed to be zero.
	
	All other values for this element are relative to the trailing text margin, Negative values are defined such that the text is moved past the text margin, positive values move the text inside the text margin. As well, if the endChars attribute is specified, then this value is ignored.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:ind w:start="720" w:end="-1440" />`
	`</w:pPr>`
	
	This set of paragraph indentations specifies that this paragraph's text should be indented `1440` twentieths of a point (one inch) into the right text margin in this document, assuming this is a left to right paragraph. *end example*]
	
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

endChars (End Indentation in Character Units)	Specifies the indentation which shall be placed at the end of this paragraph – between the right text margin for this paragraph and the right edge of that paragraph's content in a left to right paragraph, and the left text margin and the left edge of that paragraph's text in a right to left paragraph. If the mirrorIndents property (§17.3.1.18) is specified for this paragraph, then this indent is used for the outside page edge - the left page edge for odd numbered pages and the right page edge for even numbered pages. This value is specified in hundredths of a character unit. If this attribute is omitted, its value shall be assumed to be zero. All other values for this element are relative to the trailing text margin, negative values are defined such that the text is moved past the text margin, positive values move the text inside the text margin. As well, if the end attribute is specified, then its value is ignored, and is superseded by this value. [*Example*: Consider the following WordprocessingML fragment:

Attributes	Description

	`<w:pPr>`
	`  <w:ind w:endChars="250" />`
	`</w:pPr>`
	This set of paragraph indentations specifies that this paragraph's text should be indented two and a half character units from the right text margin in this document, assuming this is a left to right paragraph. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

firstLine (Additional First Line Indentation)	Specifies the additional indentation which shall be applied to the first line of the parent paragraph. This additional indentation is specified relative to the paragraph indentation which is specified for all other lines in the parent paragraph.
	The firstLine and hanging attributes are mutually exclusive, if both are specified, then the firstLine value is ignored. If the firstLineChars attribute is also specified, then this value is ignored. If this attribute is omitted, then its value shall be assumed to be zero (if needed).
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:ind w:start="1440" w:end="720" w:firstLine="1440" />`
	`</w:pPr>`
	This set of indentations specifies that the first line should be indented `1440` twentieths of a point (one inch) from the indentation specified for all remaining paragraphs, which is the `1440` twentieths of a point, as specified by the start attribute. This gives the first line a two inch indentation from the text margin. *end example*]
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

firstLineChars (Additional First Line Indentation in Character Units)	Specifies the additional indentation which shall be applied to the first line of the parent paragraph. This additional indentation is specified relative to the paragraph indentation which is specified for all other lines in the parent paragraph. It is specified in one hundredths of a character unit. The firstLineChars and hangingChars attributes are mutually exclusive, if both are specified, then the firstLineChars value is ignored. If the firstLine attribute is also specified, then this value supersedes its other value. If this attribute is omitted, then its value shall be assumed to be zero (if needed). [*Example*: Consider the following WordprocessingML fragment:

Attributes	Description

	`<w:pPr>`
	`  <w:ind w:start="1440" w:end="720" w:firstLineChars="140" />`
	`</w:pPr>`
	This set of indentations specifies that the first line should be indented `140` hundredths of a character units from the indentation specified for all remaining paragraphs, which is the `1440` twentieths of a point specified by the start attribute. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

hanging (Indentation Removed from First Line)	Specifies the indentation which shall be removed from the first line of the parent paragraph, by moving the indentation on the first line back towards the beginning of the direction of text flow.
	This indentation is specified relative to the paragraph indentation which is specified for all other lines in the parent paragraph.
	The firstLine and hanging attributes are mutually exclusive, if both are specified, then the firstLine value is ignored. If the hangingChars attribute is also specified, then this value is ignored. If this attribute is omitted, its value shall be assumed to be zero (if needed).
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:ind w:start="1440" w:end="720" w:hanging="720" />`
	`</w:pPr>`
	This set of indentations specifies that the first line should be indented `720` twentieths of a point (one inch) towards the text margin from the indentation specified for all remaining paragraphs, which is the `1440` twentieths of a point specified by the start attribute. This gives the first line a one-half inch indentation from the text margin. *end example*]
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

hangingChars (Indentation Removed From First Line in Character Units)	Specifies the indentation which shall be removed from the first line of the parent paragraph, by moving the indentation on the first line back towards the beginning of the direction of text flow. This indentation is specified relative to the paragraph indentation which is specified for all other lines in the parent paragraph. It is specified in one hundredths of a character unit. The firstLineChars and hangingChars attributes are mutually exclusive, if both are specified, then the firstLine value is ignored. If the hanging attribute is also specified,

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	then its value is superseded by this value. If this attribute is omitted, its value shall be assumed to be zero (if needed).
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:ind w:start="1440" w:end="720" w:hangingChars="100" />`
	`</w:pPr>`
	This set of indentations specifies that the first line should be indented one character unit towards the text margin from the indentation specified for all remaining paragraphs, which is the `1440` twentieths of a point specified by the start attribute. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

start (Start Indentation)	Specifies the indentation which shall be placed at the start of this paragraph – between the left text margin for this paragraph and the left edge of that paragraph's content in a left to right paragraph, and the right text margin and the right edge of that paragraph's text in a right to left paragraph. If the mirrorIndents property (§17.3.1.18) is specified for this paragraph, then this indent is used for the inside page edge - the right page edge for odd numbered pages and the left page edge for even numbered pages.
	If this attribute is omitted, its value shall be assumed to be zero.
	All other values for this element are relative to the leading text margin, Negative values are defined such that the text is moved past the text margin, positive values move the text inside the text margin. This value can be superseded for the first line only via use of the firstLine or hanging attributes. As well, if the startChars attribute is specified, then this value is ignored.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:ind w:start="720" w:end="2880" />`
	`</w:pPr>`
	This set of paragraph indentations specifies that this paragraph's text should be indented `720` twentieths of a point (one half inch) from the left text margin in this document, assuming this is a left to right paragraph. *end example*]
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

startChars (Start Indentation in Character Units)	Specifies the indentation which shall be placed at the start of this paragraph – between the left text margin for this paragraph and the left edge of that paragraph's content in a left to right paragraph, and the right text margin and the right edge of that paragraph's text in a right to left paragraph. If the mirrorIndents property (§17.3.1.18) is specified

Attributes	Description

	for this paragraph, then this indent is used for the inside page edge - the right page edge for odd numbered pages and the left page edge for even numbered pages.
	This value is specified in hundredths of a character unit.
	If this attribute is omitted, its value shall be assumed to be zero.
	All other values for this element are relative to the leading text margin, Negative values are defined such that the text is moved past the text margin, positive values move the text inside the text margin. This value can be superseded for the first line only via use of the firstLine or hanging attributes. As well, if the start attribute is specified, then its value is ignored, and is superseded by this value.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:ind w:startChars="250" />`
	`</w:pPr>`
	This set of paragraph indentations specifies that this paragraph's text should be indented two and a half character units from the left text margin in this document, assuming this is a left to right paragraph. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Ind) is located in §A.1. *end note*]

17.3.1.13 jc (Paragraph Alignment)

This element specifies the paragraph alignment which shall be applied to text in this paragraph.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no alignment is applied to the paragraph.

[*Example*: Consider a paragraph which should be right justified to the right page side paragraph extents within a document. This constraint is specified in the following WordprocessingML content:

```
<w:pPr>
  <w:jc w:val="end" />
</w:pPr>
```

The paragraph is now `right` justified on the page. *end example*]

Attributes	Description

val (Alignment Type)	Specifies the justification which should be applied to the parent object within a document.
	The possible values (see below) for this attribute are always specified with `left` specifying justification relative to the leading edge of the paragraph, and therefore change semantic between right-to-left and left-to-right documents.
	[*Example*: Consider the following WordprocessingML fragment for a paragraph in a document:
	`<w:pPr>`
	`  <w:jc w:val="end" />`
	`</w:pPr>`
	This paragraph is now right justified on the page for a left-to-right paragraph, left justfied for a right-to-left paragraph. *end example*]
	The possible values for this attribute are defined by the ST_Jc simple type (§17.18.44).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Jc) is located in §A.1. *end note*]

17.3.1.14 keepLines (Keep All Lines On One Page)

This element specifies that when rendering this document in a page view, all lines of this paragraph are maintained on a single page whenever possible.

This means that if the contents of the current paragraph would normally span across two pages due to the placement of the paragraph's text, all lines in this paragraph shall be moved onto the next page to ensure they are displayed together. If this is not possible because all lines in the paragraph would exceed a single page in any case, then lines in this paragraph shall start on a new page, with page breaks as needed afterwards.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this property shall not be applied.

[*Example*: Consider a WordprocessingML document in which a code fragment (such as the schema fragments in this document) are defined such that they should never be broken across a page boundary in order to improve readability. This constraint would be specified using the following paragraph properties in WordprocessingML:

```
<w:pPr>
  <w:keepLines />
  …
</w:pPr>
```

This setting ensures that the schema fragment is displayed on one page if possible. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

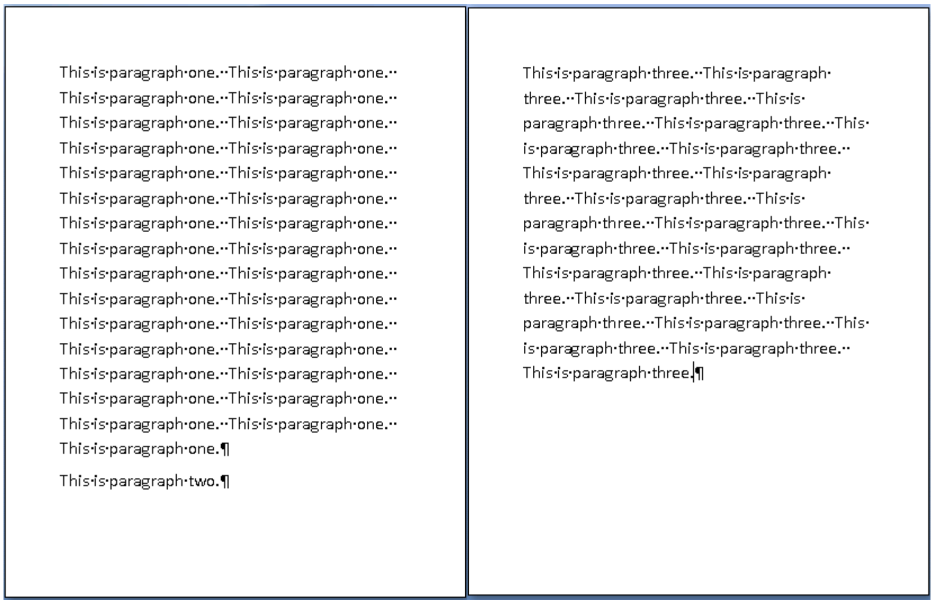
17.3.1.15 keepNext (Keep Paragraph With Next Paragraph)

This element specifies that when rendering this document in a paginated view, the contents of this paragraph are at least partly rendered on the same page as the following paragraph whenever possible.

This means that if the contents of the current paragraph would normally be completely rendered on a different page than the following paragraph (because only one of the two paragraphs would fit on the remaining space on the first page), then both paragraphs shall be rendered on a single page. This property can be chained between multiple paragraphs to ensure that all paragraphs are rendered on a single page without any intervening page boundaries. If this is not possible the entire set of paragraphs that are grouped together using this property would exceed a single page in any case, then the set of "keep with next" paragraphs shall start on a new page, with page breaks as needed afterwards.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this property shall not be applied.

[*Example*: Consider the following document with three paragraphs:



This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.¶

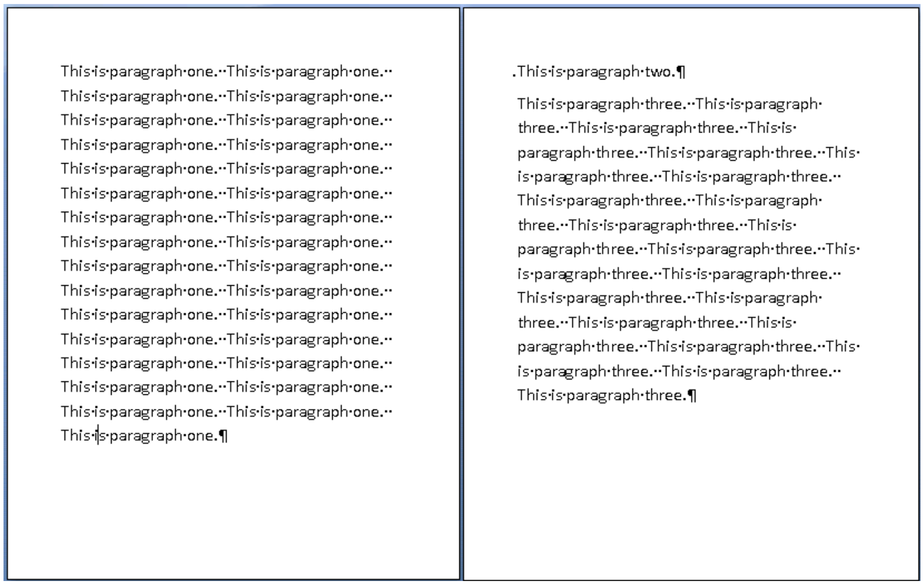
This·is·paragraph·two.¶

This·is·paragraph·three.··This·is·paragraph·
three.··This·is·paragraph·three.··This·is·
paragraph·three.··This·is·paragraph·three.··This·
is·paragraph·three.··This·is·paragraph·three.··
This·is·paragraph·three.··This·is·paragraph·
three.··This·is·paragraph·three.··This·is·
paragraph·three.··This·is·paragraph·three.··This·
is·paragraph·three.··This·is·paragraph·three.··
This·is·paragraph·three.··This·is·paragraph·
three.··This·is·paragraph·three.··This·is·
paragraph·three.··This·is·paragraph·three.··This·
is·paragraph·three.··This·is·paragraph·three.··
This·is·paragraph·three.¶

As shown above, the second paragraph and third paragraph are being rendered on two separate pages. However, a producer can specify that the second paragraph should not be displayed without any part of the third by setting the keepNext element as follows:

```
<w:pPr>
  <w:keepNext/>
  …
</w:pPr>
```

This would ensure that the second paragraph is displayed on the same page as the third paragraph:

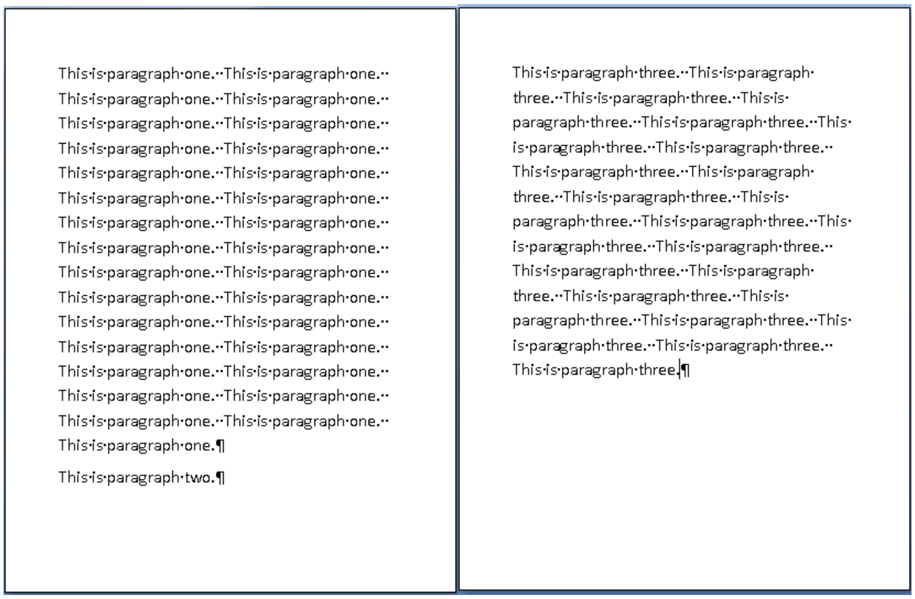


This·is·paragraph·one.··This·is·paragraph·one.¶

.This·is·paragraph·two.¶

This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.¶

However, if the first paragraph was also set to keepNext, then this set of grouped paragraphs would exceed a page, therefore the set of paragraphs would be grouped and start on the first page, resulting in the second paragraph reappearing on page one:



This·is·paragraph·one.··This·is·paragraph·one.¶

This·is·paragraph·two.¶

This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.··This·is·paragraph·three.¶

Since the paragraphs cannot all be put on one page, they all start on page one and flow as needed. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.16 kinsoku (Use East Asian Typography Rules for First and Last Character per Line)

This element specifies whether East Asian typography and line-breaking rules shall be applied to text in this paragraph to determine which characters can begin and end each line. This property only applies to Simplified Chinese, Traditional Chinese, and Japanese text in this paragraph.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this property shall be applied to Simplified Chinese, Traditional Chinese, and Japanese text in this paragraph.

If these rules are set on the current paragraph, then the following rules are applied to the all first and last characters in the paragraph except the first and last character in the paragraph. By default, the following settings are used for kinsoku paragraphs:

Chinese (Simplified)

- Cannot start a line:
  !%),.:;>?]}¢¨°·ˇˉ―‖’”…‰′″›℃∶、。〃〉》」』】〕〗〞︶︺︾﹀﹄﹚﹜﹞！＂％＇），．：；？］｀｜｝～￠

  (Unicode character values: U+0021, U+0025, U+0029, U+002C, U+002E, U+003A, U+003B, U+003E, U+003F, U+005D, U+007D, U+00A2, U+00A8, U+00B0, U+00B7, U+02C7, U+02C9, U+2015, U+2016, U+2019, U+201D, U+2026, U+2030, U+2032, U+2033, U+203A, U+2103, U+2236, U+3001, U+3002, U+3003, U+3009, U+300B, U+300D, U+300F, U+3011, U+3015, U+3017, U+301E, U+FE36, U+FE3A, U+FE3E, U+FE40, U+FE44, U+FE5A, U+FE5C, U+FE5E, U+FF01, U+FF02, U+FF05, U+FF07, U+FF09, U+FF0C, U+FF0E, U+FF1A, U+FF1B, U+FF1F, U+FF3D, U+FF40, U+FF5C, U+FF5D, U+FF5E, and U+FFE0, respectively)

- Cannot end a line:
  $([{£¥·‘“〈《「『【〔〖〝﹙﹛﹝＄（．［｛￡￥

  (Unicode character values: U+0024, U+0028, U+005B, U+007B, U+00A3, U+00A5, U+00B7, U+2018, U+201C, U+3008, U+300A, U+300C, U+300E, U+3010, U+3014, U+3016, U+301D, U+FE59, U+FE5B, U+FE5D, U+FF04, U+FF08, U+FF0E, U+FF3B, U+FF5B, U+FFE1, and U+FFE5, respectively)

Chinese (Traditional)

- Cannot start a line:
  !),.:;?]}¢·–—’”•‥…‧′╴、。〉》」』】〕〞︰︱︲︳︴︶︸︺︼︾﹀﹂﹄﹏﹐﹑﹒﹔﹕﹖﹗﹚﹜﹞！），．：；？］｜｝､

  (Unicode character values: U+0021, U+0029, U+002C, U+002E, U+003A, U+003B, U+003F, U+005D, U+007D, U+00A2, U+00B7, U+2013, U+2014, U+2019, U+201D, U+2022, U+2025, U+2026, U+2027, U+2032, U+2574, U+3001, U+3002, U+3009, U+300B, U+300D, U+300F, U+3011, U+3015, U+301E, U+FE30, U+FE31, U+FE33, U+FE34, U+FE36, U+FE38, U+FE3A, U+FE3C, U+FE3E, U+FE40, U+FE42, U+FE44, U+FE4F, U+FE50, U+FE51, U+FE52, U+FE54, U+FE55, U+FE56, U+FE57, U+FE5A, U+FE5C, U+FE5E, U+FF01, U+FF09, U+FF0C, U+FF0E, U+FF1A, U+FF1B, U+FF1F, U+FF3D, U+FF5C, U+FF5D, and U+FF64, respectively)

- Cannot end a line:
  ([{£¥‘“‵〈《「『【〔〝︵︷︹︻︽︿﹁﹃﹙﹛﹝（｛

  (Unicode character values: U+0028, U+005B, U+007B, U+00A3, U+00A5, U+2018, U+201C, U+2035, U+3008, U+300A, U+300C, U+300E, U+3010, U+3014, U+301D, U+FE35, U+FE37, U+FE39, U+FE3B, U+FE3D, U+FE3F, U+FE41, U+FE43, U+FE59, U+FE5B, U+FE5D, U+FF08, and U+FF5B, respectively)

Japanese

- Cannot start a line:
  !%),.:;?]}¢°’”‰′″℃、。々〉》」』】〕゛゜ゝゞ・ヽヾ！％），．：；？］｝｡｣､･ﾞﾟ￠

  (Unicode character values: U+0021, U+0025, U+0029, U+002C, U+002E, U+003A, U+003B, U+003F, U+005D, U+007D, U+00A2, U+00B0, U+2019, U+201D, U+2030, U+2032, U+2033, U+2103, U+3001, U+3002, U+3005, U+3009, U+300B, U+300D, U+300F, U+3011, U+3015, U+309B, U+309C, U+309D, U+309E, U+30FB, U+30FD, U+30FE, U+FF01, U+FF05, U+FF09, U+FF0C, U+FF0E, U+FF1A, U+FF1B, U+FF1F, U+FF3D, U+FF5D, U+FF61, U+FF63, U+FF64, U+FF65, U+FF9E, U+FF9F, and U+FFE0, respectively)

- Cannot end a line:
  $([\{£¥‘“〈《「『【〔＄（［｛｢￡￥

  (Unicode character values: U+0024, U+0028, U+005B, U+005C, U+007B, U+00A3, U+00A5, U+2018, U+201C, U+3008, U+300A, U+300C, U+300E, U+3010, U+3014, U+FF04, U+FF08, U+FF3B, U+FF5B, U+FF62, U+FFE1, and U+FFE5, respectively)

Korean

- Cannot start a line:
  !%),.:;?]}¢°’”′″℃〉》」』】〕！％），．：；？］｝￠

  (Unicode character values: U+0021, U+0025, U+0029, U+002C, U+002E, U+003A, U+003B, U+003F, U+005D, U+007D, U+00A2, U+00B0, U+2019, U+201D, U+2032, U+2033, U+2103, U+3009, U+300B, U+300D, U+300F, U+3011, U+3015, U+FF01, U+FF05, U+FF09, U+FF0C, U+FF0E, U+FF1A, U+FF1B, U+FF1F, U+FF3D, U+FF5D, and U+FFE0, respectively)

- Cannot end a line:
  $([\{£¥‘“〈《「『【〔＄（［｛￡￥￦

(Unicode character values: U+0024, U+0028, U+005B, U+005C, U+007B, U+00A3, U+00A5, U+2018, U+201C, U+3008, U+300A, U+300C, U+300E, U+3010, U+3014, U+FF04, U+FF08, U+FF3B, U+FF5B, U+FFE1, U+FFE5, and U+FFE6, respectively)

If the strictFirstAndLastChars property (§17.15.1.82) is set in the Document Settings part, then the following settings supersede the defaults for Japanese:

- Cannot start a line:
  !%),.:;?]}¢°’”‰′″℃、。々〉》」』】〕ぁぃぅぇぉっゃゅょゎ゛゜ゝゞァィゥェォッャュョヮヵヶ・ーヽヾ！％），．：；？］｝｡｣､･ｧｨｩｪｫｬｭｮｯｰﾞﾟ￠

  (Unicode character values: U+0021, U+0025, U+0029, U+002C, U+002E, U+003A, U+003B, U+003F, U+005D, U+007D, U+00A2, U+00B0, U+2019, U+201D, U+2030, U+2032, U+2033, U+2103, U+3001, U+3002, U+3005, U+3009, U+300B, U+300D, U+300F, U+3011, U+3015, U+3041, U+3043, U+3045, U+3047, U+3049, U+3063, U+3083, U+3085, U+3087, U+308E, U+309B, U+309C, U+309D, U+309E, U+30A1, U+30A3, U+30A5, U+30A7, U+30A9, U+30C3, U+30E3, U+30E5, U+30E7, U+30EE, U+30F5, U+30F6, U+30FB, U+30FC, U+30FD, U+30FE, U+FF01, U+FF05, U+FF09, U+FF0C, U+FF0E, U+FF1A, U+FF1B, U+FF1F, U+FF3D, U+FF5D, U+FF61, U+FF63, U+FF64, U+FF65, U+FF67, U+FF68, U+FF69, U+FF6A, U+FF6B, U+FF6C, U+FF6D, U+FF6E, U+FF6F, U+FF70, U+FF9E, U+FF9F, and U+FFE0, respectively)

- Cannot end a line:
  $([\{£¥‘“〈《「『【〔＄（［｛｢￡￥

  (Unicode character values: U+0024, U+0028, U+005B, U+005C, U+007B, U+00A3, U+00A5, U+2018, U+201C, U+3008, U+300A, U+300C, U+300E, U+3010, U+3014, U+FF04, U+FF08, U+FF3B, U+FF5B, U+FF62, U+FFE1, and U+FFE5, respectively)

If the noLineBreaksBefore property (§17.15.1.59) is set in the Document Settings part, then the characters it specifies cannot begin a line for the specified language. If the noLineBreaksAfter property (§17.15.1.58) is set in the Document Settings part, then the characters it specifies cannot end a line for the specified language. In both cases, those settings shall supersede the defaults specified above.

[*Example*: Consider a document with a paragraph which should not use the kinsoku line breaking properties. This paragraph would define the following WordprocessingML:

```
<w:pPr>
  <w:kinsoku w:val="off" />
</w:pPr>
```

This paragraph would now be exempt from any kinsoku line breaking rules, and the characters specified above are allowed to begin and end lines as they normally would. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.17 left (Left Paragraph Border)

This element specifies the border which shall be displayed on the left side of the page around the specified paragraph. This shall not change based on the paragraph direction.

To determine if any two adjoining paragraphs should have a left border which spans the full line height or not, the left border shall be drawn between the top border or between border at the top (whichever would be rendered for the current paragraph), and the bottom border or between border at the bottom (whichever would be rendered for the current paragraph).

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no left border shall be applied.

[*Example*: Consider the following two paragraphs' WordprocessingML definition:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>First paragraph.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
```

```
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>Second paragraph.</w:t>
  </w:r>
</w:p>
```

Since the paragraph border set is identical between the two paragraphs, the paragraphs are connected by a between border. These paragraphs therefore draw the left border between the top and between borders for the first paragraph, and the between and bottom borders for the second paragraph. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.3.1.18 mirrorIndents (Use Left/Right Indents as Inside/Outside Indents)

This element specifies whether the paragraph indents should be interpreted as mirrored indents. When this element is present, the start indent shall become the inside indent (the one closest to the binding) and the end indent shall become the outside indent (the one furthest from the binding). [*Note*: This mirroring is typically used when the contents of the document are used to generate *signatures* – combinations of pages which are then placed in a binding. When signatures are printed in a left-to-right document, the first, third, etc. pages are printed on the left side of the combined sheet, and the second, fourth, etc. are printed on its right side, then bound and folded. For a right-to-left document, the first, third, etc. pages are printed on the right side of the combined sheet, and the second, fourth, etc. are printed on its left side. *end note*]

If the mirrorIndents property is specified for this paragraph, then the inside page edge is the end page edge for odd numbered pages and the start page edge for even numbered pages. Conversely, the outside page edge is the start page edge for odd numbered pages and the end page edge for even numbered pages. Odd and even numbering in the text above refers to the ordinal position of the page in the paginated document, not to the page number which may appear on each page.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this property shall not be applied.

[*Example*: Consider a paragraph on the first page of a document which should have a one inch indentation from the text margins on the inside edge when the resulting document is printed and bound. This means that the paragraph has a one inch right border if it is on an odd numbered page, and a one inch left border if it is on an even numbered page. This set of indentations is specified using the following WordprocessingML:

```
<w:pPr>
  <w:ind w:start="1440" />
```

```
    <w:mirrorIndents />
  </w:pPr>
```

This set of indentation properties specifies that a 1440 twip indentation should be provided on the leading side of the text margins for this paragraph. However, since the mirrorIndents property is set, the start indent is really the inside indent, and if this paragraph is on page one, must result in a one inch right indent from the text margin. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.19 numPr (Numbering Definition Instance Reference)

This element specifies that the current paragraph uses numbering information that is defined by a particular *numbering definition instance*.

The presence of this element specifies that the paragraph inherits the properties specified by the numbering definition in the num element (§17.9.15) at the level specified by the level specified in the lvl element (§17.9.6) and shall have an associated number positioned before the beginning of the text flow in this paragraph. When this element appears as part of the paragraph formatting for a paragraph style, then any numbering level defined using the ilvl element shall be ignored, and the pStyle element (§17.9.23) on the associated abstract numbering definition shall be used instead.

[*Example*: Consider a paragraph in a document which should be associated with level 4 of a numbering definition with ID 0. Associating the paragraph with this numbering definition would be specified using the following WordprocessingML:

```
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="4" />
      <w:numId w:val="0" />
    </w:numPr>
  </w:pPr>
```

The numPr element specifies that this paragraph must contain numbering information, and its children specify that the numbering definition for that numbering information must have a numId of 0 and an ilvl of 4 within that numbering definition. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_NumPr) is located in §A.1. *end note*]

17.3.1.20 outlineLvl (Associated Outline Level)

This element specifies the *outline level* which shall be associated with the current paragraph in the document. The *outline level* specifies an integer which defines the level of the associated text. This level shall not affect the appearance of the text in the document, but shall be used to calculate the TOC field (§17.16.5.68) if the appropriate field switches have been set, and can be used by consumers to provide additional application behavior.

The outline level of text in the document (specified using the val attribute) can be from 0 to 9, where 9 specifically indicates that there is no outline level specifically applied to this paragraph. If this element is omitted, then the outline level of the content is assumed to be 9 (no level).

[*Example*: Consider a paragraph in a document which has outline level 1 applied to it. This paragraph would specify the following WordprocessingML:

```
<w:pPr>
  <w:outlineLvl w:val="0" />
</w:pPr>
```

This paragraph is now of outline level 1, and if a table of contents field is inserted that utilizes outlines levels, the text in this paragraph is at level one in the TOC. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.3.1.21 overflowPunct (Allow Punctuation to Extend Past Text Extents)

This element specifies that the text in this paragraph shall be allowed to extend one character beyond the extents applied by any indents/margins when the character that extends past those extents is a punctuation character.

Omitting this element sets its value to true.

[*Example*: Consider a WordprocessingML document with the following string at the end of a line:

```
"This is some text in quotation marks"
```

Typically, if the text extents would normally fall between the letter s and the closing quotation mark, the quotation mark would be allowed to extend past the end of the line by one character even though the punctuation is not part of the word marks (since the omission of overflowPunct is equivalent to setting its val attribute to true).

However, if this behavior should not be applied to this paragraph, a producer can specify this by setting the property in the WordprocessingML:

```
<w:pPr>
  <w:overflowPunct w:val="0" />
</w:pPr>
```

The line would now break after the letter s, regardless of the fact that the next character is a quotation mark. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.22 p (Paragraph)

This element specifies a paragraph of content in the document.

The contents of a paragraph in a WordprocessingML document shall consist of any combination of the following four kinds of content:

- Paragraph properties
- Annotations (bookmarks, comments, revisions)
- Custom markup
- Run level content (fields, hyperlinks, runs)

[*Example*: Consider a basic WordprocessingML document with a single paragraph. This paragraph would be expressed as follows:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Text</w:t>
      </w:r>
      <w:fldSimple w:instr="AUTHOR">
        <w:r>
          <w:t>Author Name</w:t>
        </w:r>
      </w:fldSimple>
    </w:p>
  </w:body>
</w:document>
```

The p element is the container for all of the content in the paragraph, which in this example includes both a text run and a simple field. *end example*]

Attributes	Description
rsidDel (Revision Identifier for Paragraph Deletion)	Specifies an identifier used to track the editing session when the paragraph was deleted from the main document. All rsid* attributes throughout this document with the same value, if present, shall indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidP (Revision Identifier for Paragraph Properties)	This attribute specifies an identifier used to track the editing session when the paragraph's properties were last modified in this document. All rsid* attributes throughout this document with the same value, if present, shall indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidR (Revision Identifier for Paragraph)	This attribute specifies an identifier used to track the editing session when the paragraph was added to the main document. All rsid* attributes throughout this document with the same value, if present, shall indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

©ISO/IEC 2012 – All rights reserved

Attributes	Description
rsidRDefault (Default Revision Identifier for Runs)	This attribute specifies an identifier used for all runs in this paragraph which do not explicitly declare an rsidR attribute. This attribute allows consumers to optimize the locations where rsid* values are written in this document. All rsid* attributes throughout this document with the same value, if present, shall indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidRPr (Revision Identifier for Paragraph Glyph Formatting)	This attribute specifies an identifier used to track the editing session when the glyph character representing the paragraph mark was last modified in the main document. All rsid* attributes throughout this document with the same value, if present, shall indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element's content model (CT_P) is located in §A.1. *end note*]

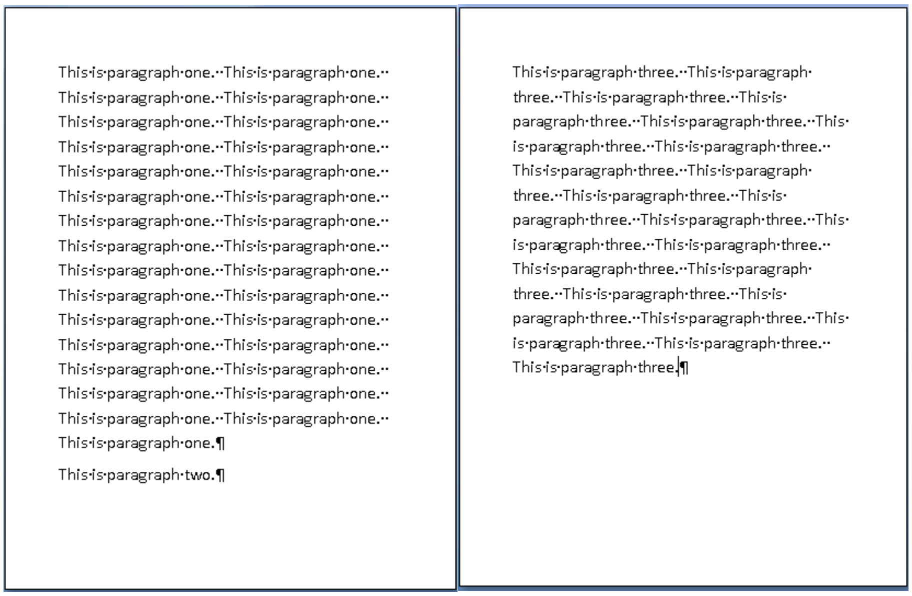
17.3.1.23 pageBreakBefore (Start Paragraph on Next Page)

This element specifies that when rendering this document in a paginated view, the contents of this paragraph are rendered on the start of a new page in the document.

This means that if the contents of the current paragraph would normally be rendered on the middle of a page in the host document, then the paragraph shall be rendered on a new page as if the paragraph was preceded by a page break in the WordprocessingML contents of the document. This property supersedes any use of the keepNext property, so that if any paragraph wishes to be on the same page as this paragraph, they are still be separated by a page break.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this property shall not be applied.

[*Example*: Consider the following document with three paragraphs:



This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.¶

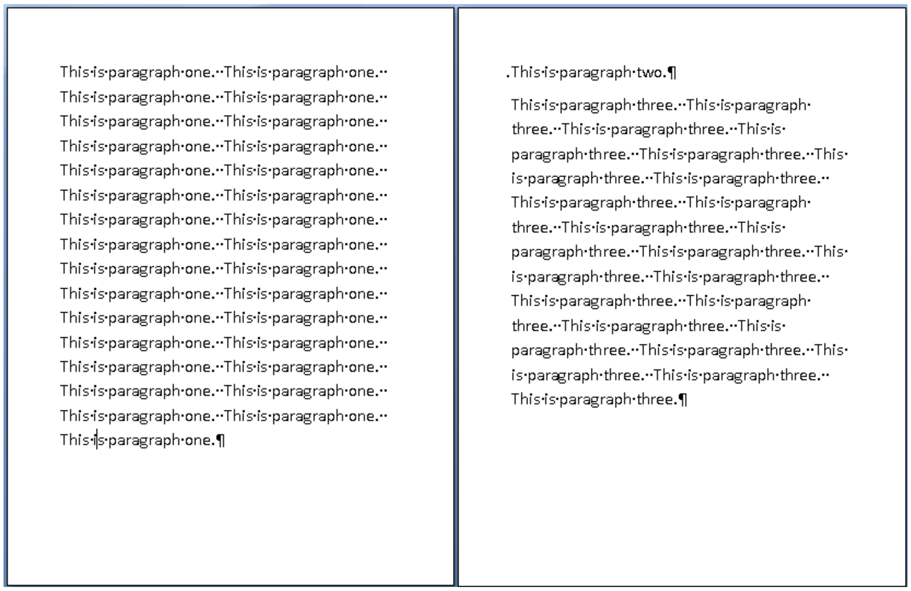
This·is·paragraph·two.¶

This·is·paragraph·three.··This·is·paragraph·
three.··This·is·paragraph·three.··This·is·
paragraph·three.··This·is·paragraph·three.··This·
is·paragraph·three.··This·is·paragraph·three.··
This·is·paragraph·three.··This·is·paragraph·
three.··This·is·paragraph·three.··This·is·
paragraph·three.··This·is·paragraph·three.··This·
is·paragraph·three.··This·is·paragraph·three.··
This·is·paragraph·three.··This·is·paragraph·
three.··This·is·paragraph·three.··This·is·
paragraph·three.··This·is·paragraph·three.··This·
is·paragraph·three.··This·is·paragraph·three.··
This·is·paragraph·three.¶

As shown above, the second paragraph is rendered at the bottom of page one. However, a producer can specify that the second paragraph should be displayed at the top of a new page by setting the pageBreakBefore element as follows:

```
<w:pPr>
  <w:pageBreakBefore/>
</w:pPr>
```

This would ensure that the second paragraph is displayed on a new page:



This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.··This·is·paragraph·one.··
This·is·paragraph·one.¶

.This·is·paragraph·two.¶

This·is·paragraph·three.··This·is·paragraph·
three.··This·is·paragraph·three.··This·is·
paragraph·three.··This·is·paragraph·three.··This·
is·paragraph·three.··This·is·paragraph·three.··
This·is·paragraph·three.··This·is·paragraph·
three.··This·is·paragraph·three.··This·is·
paragraph·three.··This·is·paragraph·three.··This·
is·paragraph·three.··This·is·paragraph·three.··
This·is·paragraph·three.··This·is·paragraph·
three.··This·is·paragraph·three.··This·is·
paragraph·three.··This·is·paragraph·three.··This·
is·paragraph·three.··This·is·paragraph·three.··
This·is·paragraph·three.¶

Since the paragraph is specified to start on a new page, it begins page two even though it could have fit on page one. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.24 pBdr (Paragraph Borders)

This element specifies the borders for the parent paragraph. Each child element shall specify a specific kind of border (left, right, bottom, top, and between).

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no paragraph borders shall be applied.

[*Example*: Consider a pairing of paragraphs which have a three point red border around them, and a six point border between them. These paragraphs would each have the following set of paragraph borders:

```
<w:pBdr>
  <w:top w:val="single" w:sz="24" w:space="1" w:color="FF0000" />
  <w:left w:val="single" w:sz="24" w:space="4" w:color="FF0000" />
  <w:bottom w:val="single" w:sz="24" w:space="1" w:color="FF0000" />
  <w:right w:val="single" w:sz="24" w:space="4" w:color="FF0000" />
  <w:between w:val="single" w:sz="48" w:space="1" w:color="4D5D2C" />
</w:pBdr>
```

The resulting paragraphs have identical pBdr values, therefore they would use the top, left, bottom, and right borders around them as a units, and the between border between each other. This matching heuristic is further discussed in the child elements of the pBdr element. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_PBdr) is located in §A.1. *end note*]

17.3.1.25 pPr (Previous Paragraph Properties)

This element specifies a set of paragraph properties which shall be attributed to a revision by a particular author and at a particular time. This element contains the set of properties which have been tracked as a specific set of revisions by one author.

[*Example*: Consider a paragraph which should have a set of paragraph formatting properties that were added with revision tracking turned on. This set of revised properties is specified in the paragraph properties as follows:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:bottom w:val="single" w:sz="8" w:space="4" w:color="4F81BD" />
    </w:pBdr>
    <w:pPrChange w:author="user1" … >
```

```
        <w:pPr>
          <w:spacing w:after="300" />
          <w:contextualSpacing />
        </w:pPr>
      </w:pPrChange>
    </w:pPr>
  </w:p>
```

The pPr element under pPrChange specifies the properties which are applied to the current paragraph with revision tracking turned on - in this case, spacing after the paragraph using the spacing element (§17.3.1.33), and that spacing should be ignored for paragraphs above/below of the same style using the contextualSpacing element (§17.3.1.9). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_PPrBase) is located in §A.1. *end note*]

17.3.1.26 pPr (Paragraph Properties)

This element specifies a set of paragraph properties which shall be applied to the contents of the parent paragraph after all style/numbering/table properties have been applied to the text. These properties are defined as *direct formatting*, since they are directly applied to the paragraph and supersede any formatting from styles.

[*Example*: Consider a paragraph which should have a set of paragraph formatting properties. This set of properties is specified in the paragraph properties as follows:

```
  <w:p>
    <w:pPr>
      <w:pBdr>
        <w:bottom w:val="single" w:sz="8" w:space="4" w:color="4F81BD" />
      </w:pBdr>
      <w:spacing w:after="300" />
      <w:contextualSpacing />
    </w:pPr>
  </w:p>
```

The pPr element specifies the properties which are applied to the current paragraph - in this case, a bottom paragraph border using the bottom element (§17.3.1.7), spacing after the paragraph using the spacing element (§17.3.1.33), and that spacing should be ignored for paragraphs above/below of the same style using the contextualSpacing element (§17.3.1.9). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_PPr) is located in §A.1. *end note*]

17.3.1.27 pStyle (Referenced Paragraph Style)

This element specifies the style ID of the paragraph style which shall be used to format the contents of this paragraph.

This formatting is applied at the following location in the *style hierarchy*:

- Document defaults
- Table styles
- Numbering styles
- Paragraph styles (this element)
- Character styles
- Direct Formatting

This means that all properties specified in the style element (§17.7.4.17) with a styleId which corresponds to the value in this element's val attribute are applied to the paragraph at the appropriate level in the hierarchy.

If this element is omitted, or it references a style which does not exist, then no paragraph style shall be applied to the current paragraph. As well, this property is ignored if the paragraph properties are part of a paragraph style.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:pPr>
  <w:pStyle w:val="TestParagraphStyle" />
  <w:ind w:start="1440" />
</w:pPr>
```

This paragraph specifies that it inherits all of the paragraph properties specified by the paragraph style with a styleId of TestParagraphStyle, which then has any indentation properties overridden with a start indentation of 1440 twentieths of a point, and no indentation for any other value. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`

Attributes	Description
	`</w:sdtPr>` In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.3.1.28 right (Right Paragraph Border)

This element specifies the border which shall be displayed on the right side of the page around the specified paragraph. This shall not change based on the paragraph direction.

To determine if any two adjoining paragraphs should have a right border which spans the full line height or not, the right border shall be drawn between the top border or between border at the top (whichever would be rendered for the current paragraph), and the bottom border or between border at the bottom (whichever would be rendered for the current paragraph).

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no right border shall be applied.

[*Example*: Consider the following two paragraphs' WordprocessingML definition:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
  <w:r>
    <w:t>First paragraph.</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:pPr>
      <w:pBdr>
        <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB" 
  w:themeColor="accent2" w:themeTint="33" />
        <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034" 
  w:themeColor="accent6" w:themeShade="BF" />
        <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB" 
  w:themeColor="accent2" w:themeTint="33" />
        <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B" 
  w:themeColor="accent3" w:themeTint="99" />
        <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD" 
  w:themeColor="accent1" />
      </w:pBdr>
    </w:pPr>
    <w:r>
      <w:t>Second paragraph.</w:t>
    </w:r>
  </w:p>
```

Since the paragraph border set is identical between the two paragraphs, the paragraphs are connected by a between border. These paragraphs therefore draw the right border between the top and between borders for the first paragraph, and the between and bottom borders for the second paragraph. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.3.1.29 rPr (Run Properties for the Paragraph Mark)

This element specifies the set of run properties applied to the glyph used to represent the physical location of the paragraph mark for this paragraph. This paragraph mark, being a physical character in the document, can be formatted, and therefore shall be capable of representing this formatting like any other character in the document.

If this element is not present, the paragraph mark is unformatted, as with any other run of text.

[*Example*: Consider a run of text displayed as follows, including a display format using the pilcrow sign ¶ for the paragraph mark glyph:

```
  This is some text and the paragraph mark.¶
```

If we format the display formatting for the paragraph mark by making it red and giving it a 72 point font size, then the WordprocessingML must reflect this formatting on the paragraph as follows:

```
  <w:pPr>
```

```
    <w:rPr>
      <w:color w:val="FF0000" />
      <w:sz w:val="144" />
    </w:rPr>
  </w:pPr>
```

The paragraph glyph's formatting is stored in the rPr element under the paragraph properties, since there is no run saved for the paragraph mark itself. *end example*]

The W3C XML Schema definition of this element's content model (CT_ParaRPr) is located in §A.1. Each child element from the above table shall not occur more than once. [*Note*: This restriction is not reflected in the element's content model due to limitations of W3C XML Schema language. *end note*]

17.3.1.30 rPr (Previous Run Properties for the Paragraph Mark)

This element specifies a set of run properties applied to the glyph used to represent the physical location of the paragraph mark for this paragraph which shall be attributed to a revision by a particular author and at a particular time. This element contains the set of properties which have been tracked as a specific set of revisions by one author.

[*Example*: Consider a run which has a set of run formatting properties that were added with revision tracking turned on. This set of revised properties is specified in the run properties as follows:

```
  <w:p>
    <w:pPr>
      <w:rPr>
        <w:b />
        <w:imprint />
        <w:lang w:val="en-CA" />
        <w:rPrChange w:id="1" w:author="user1">
          <w:rPr>
            <w:i />
            <w:dstrike w:val="false" />
          </w:rPr>
        </w:rPrChange>
      </w:rPr>
    </w:pPr>
  </w:p>
```

The rPr element under rPrChange specifies the properties which are applied to the run representing the paragraph mark before the revision tracking was turned on - in this case, italics using the i element (§17.3.2.16), and that any double strikethrough which was applied based on the style hierarchy must be turned off using the dstrike element (§17.3.2.9). *end example*]

The W3C XML Schema definition of this element's content model (CT_ParaRPrOriginal) is located in §A.1. Each child element from the above table shall not occur more than once. [*Note*: This restriction is not reflected in the element's content model due to limitations of W3C XML Schema language. *end note*]

17.3.1.31 shd (Paragraph Shading)

This element specifies the shading applied to the contents of the paragraph.

This shading consists of three components:

- Background Color
- (optional) Pattern
- (optional) Pattern Color

The resulting shading is applied by setting the background color behind the paragraph, then applying the pattern color using the mask supplied by the pattern over that background.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no paragraph shading shall be applied.

[*Example*: Consider a paragraph which must have a background consisting of a theme color `accent3` with a theme color `accent6` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:

```
<w:pPr>
  <w:shd w:val="pct20" w:themeColor="accent6" w:themeFill="accent3" />
</w:pPr>
```

The resulting paragraph uses the background color `accent3` under the foreground pattern color `accent6` as specified by the `pct20` pattern mask. *end example*]

This element's content model is defined by the common shading properties definition in §17.3.5.

17.3.1.32 snapToGrid (Use Document Grid Settings for Inter-Line Paragraph Spacing)

This element specifies whether the current paragraph should use the document grid lines per page settings defined in the docGrid element (§17.6.5) when laying out the contents in the paragraph. This setting determines whether the additional line pitch specified in the document grid shall be added to each line in this paragraph as specified by the document grid.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the paragraph shall use the document grid to lay out text when a document grid is defined for this document.

[*Example*: Consider two single-spaced paragraphs in a section with a document grid set to allow 15 lines per page. This document grid would effectively specifies that an additional line pitch of 45.6 points must be added to each line in order to ensure that the resulting page contains only 15 lines of text.

If this property is set on the first paragraph, but turned off on the second paragraph, as follows:

```
<w:p>
  <w:pPr>
    <w:snapToGrid w:val="off" />
  </w:pPr>
  …
</w:p>
<w:p>
  …
</w:p>
```

The resulting document must have 45.6 points of additional line pitch added to each line in paragraph two, but zero lines of additional line pitch added to each line in paragraph one, since the snapToGrid property is turned off. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.33 spacing (Spacing Between Lines and Above/Below Paragraph)

This element specifies the inter-line and inter-paragraph spacing which shall be applied to the contents of this paragraph when it is displayed by a consumer.

If this element is omitted on a given paragraph, each of its values is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the paragraph shall have no spacing applied to its lines, or above and below its contents.

[*Example*: Consider the following WordprocessingML paragraph:

```
<w:pPr>
  <w:spacing w:after="200" w:line="276" w:lineRule="auto" />
</w:pPr>
```

This paragraph specifies that it must have at least 200 twentieths of a point after the last line in each paragraph, and that the spacing in each line should be automatically calculated based on a 1.15 times (276 divided by 240) the normal single spacing calculation. *end example*]

When determining the spacing between any two paragraphs, a consumer shall use the maximum of the inter-line spacing in each paragraph, the spacing after the first paragraph and the spacing before the second paragraph to determine the net spacing between the paragraphs.

[*Example*: Consider two consecutive single-spaced paragraphs in a document, the first of which specifies spacing below of 12 points, the second of which specifies spacing above of 4 points. These constraints are expressed using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:spacing w:after="240" />
  </w:pPr>
  …
</w:p>
<w:p>
  <w:pPr>
    <w:spacing w:before="80" />
  </w:pPr>
  …
</w:p>
```

The resulting spacing between the first and second paragraph is 12 points, since that is the largest spacing requested between the two paragraphs. *end example*]

Attributes	Description

after (Spacing Below Paragraph)	Specifies the spacing that should be added after the last line in this paragraph in the document in absolute units.
	
	If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the paragraph shall have no spacing applied below its contents.
	
	If the afterLines attribute or the afterAutoSpacing attribute is also specified, then this attribute value is ignored.
	
	[*Example*: Consider the following WordprocessingML paragraph:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:spacing w:after="240" />`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph must have a minimum spacing below its final lines of 240 twentieths of a point, although the actual spacing can be determined by the inter-line spacing or the spacing above the following paragraph, if either are greater. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type

Attributes	Description
	(§22.9.2.14).

afterAutospacing (Automatically Determine Spacing Below Paragraph)	Specifies whether a consumer shall automatically determine the spacing after this paragraph based on its contents.
	This automatic spacing shall match the spacing which would be applied to the paragraph in an HTML document where no explicit spacing before/after is specified.
	If this attribute is specified, then any value in after or afterLines is ignored, and the spacing is automatically determined by the consumer.
	If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then automatic spacing is turned off (not applied).
	[*Example*: Consider a paragraph in a document whose spacing below must automatically be determined by the consumer based on the paragraph's contents. This constraint would be specified by the following WordprocessingML:
	`<w:pPr>`
	`  <w:spacing … w:afterAutospacing="on" />`
	`</w:pPr>`
	The resulting paragraph must have the spacing below its last line determined automatically by the consumer to match an HTML document as specified. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

afterLines (Spacing Below Paragraph in Line Units)	Specifies the spacing that should be added after the last line in this paragraph in the document in line units.
	The value of this attribute is specified in one hundredths of a line.
	If the afterAutoSpacing attribute is also specified, then this attribute value is ignored. If this setting is never specified in the style hierarchy, then its value shall be zero (if needed).
	[*Example*: Consider the following WordprocessingML paragraph:
	`<w:p>`
	`  <w:pPr>`
	`    <w:spacing w:afterLines="300" />`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph must have a minimum spacing below its final lines of 3 lines, although the

Attributes	Description
	actual spacing can be determined by the inter-line spacing or the spacing above the following paragraph, if either are greater. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

before (Spacing Above Paragraph)	Specifies the spacing that should be added above the first line in this paragraph in the document in absolute units.
	
	If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the paragraph shall have no spacing applied above its contents.
	
	If the beforeLines attribute or the beforeAutoSpacing attribute is also specified, then this attribute value is ignored.
	
	[*Example*: Consider the following WordprocessingML paragraph:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:spacing w:before="80" />`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph must have a minimum spacing above its first line of `80` twentieths of a point, although the actual spacing can be determined by the inter-line spacing or the spacing below the last line in the preceding paragraph, if either are greater. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

beforeAutospacing (Automatically Determine Spacing Above Paragraph)	Specifies whether a consumer shall automatically determine the spacing before this paragraph based on its contents. This automatic spacing shall match the spacing which would be applied to the paragraph in an HTML document where no explicit spacing before/after is specified. If this attribute is specified, then any value in before or beforeLines is ignored, and the spacing is automatically determined by the consumer. If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then automatic spacing is turned off (not applied). [*Example*: Consider a paragraph in a document whose spacing above must automatically

Attributes	Description

	be determined by the consumer based on the paragraph's contents. This constraint would be specified by the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:spacing … w:beforeAutospacing="on" />`
	`</w:pPr>`
	
	The resulting paragraph must have the spacing above its first line determined automatically by the consumer to match an HTML document as specified. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

beforeLines (Spacing Above Paragraph IN Line Units)	Specifies the spacing that should be added before the first line in this paragraph in the document in line units.
	
	The value of this attribute is specified in one hundredths of a line.
	
	If the beforeAutoSpacing attribute is also specified, then this attribute value is ignored. If this setting is never specified in the style hierarchy, then its value shall be zero (if needed).
	
	[*Example*: Consider the following WordprocessingML paragraph:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:spacing w:beforeLines="100" />`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph must have a minimum spacing above its first line of 1 line, although the actual spacing can be determined by the inter-line spacing or the spacing below the preceding paragraph, if either are greater. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

line (Spacing Between Lines in Paragraph)	This attribute specifies the amount of vertical spacing between lines of text within this paragraph. If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no line spacing shall be applied to lines within this paragraph. If the value of the lineRule attribute is either `atLeast` or `exactly`, then the value of this attribute shall be interpreted as twentieths of a point. When the value of the lineRule attribute is either `exactly`, the text shall be positioned as follows within that

Attributes	Description

	line height:
	• When the line height is too small, the text shall be positioned at the bottom of the line (i.e. clipped from the top down)
	• When the line height is too large, the text shall be centered in the available space.
	
	If the value of the lineRule attribute is `auto`, then the value of the line attribute shall be interpreted as 240ths of a line, in the manner described by the simple type's values.
	
	[*Example*: Consider the following WordprocessingML paragraph which should have an inter-line spacing of 1.15 times the line height. This constraint would be specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:spacing w:line="276" w:lineRule="auto" />`
	`</w:pPr>`
	
	The lineRule attribute value of `auto` specifies that the value of the line attribute is to be interpreted in 240ths of a single line height, which means that the net spacing is 276/240ths of a line or 1.15 lines tall. *end example*]
	
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

lineRule (Spacing Between Lines)	Specifies how the spacing between lines is calculated as stored in the line attribute.
	
	If this attribute is omitted, then it shall be assumed to be of a value `auto` if a line attribute value is present.
	
	If the value of this attribute is either `atLeast` or `exactly`, then the value of the line attribute shall be interpreted as twentieths of a point, in the manner described by the simple type's values.
	
	If the value of this attribute is `auto`, then the value of the line attribute shall be interpreted as 240ths of a line, in the manner described by the simple type's values.
	
	[*Example*: Consider the following WordprocessingML paragraph which should have an inter-line spacing of 1.15 times the line height. This constraint would be specified using the following WordprocessingML:
	
	`<w:pPr>`
	`  <w:spacing w:line="276" w:lineRule="auto" />`
	`</w:pPr>`
	
	The lineRule attribute value of `auto` specifies that the value of the line attribute is to be interpreted in 240ths of a single line height. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_LineSpacingRule simple type (§17.18.48).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Spacing) is located in §A.1. *end note*]

17.3.1.34 suppressAutoHyphens (Suppress Hyphenation for Paragraph)

This element specifies whether any hyphenation shall be performed on this paragraph by the consumer when requested using the autoHyphenation element (§17.15.1.10) in the document's settings. This element specifies whether the current paragraph should be exempted from any hyphenation which is applied by the consumer on this document.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the default hyphenation settings for the document, as specified in the autoHyphenation element, shall apply to the contents of this paragraph.

[*Example*: Consider a document which must be hyphenated automatically by a consumer, since it has the autoHyphenation element set to true in its document settings. If this paragraph should be exempted from that hyphenation pass, this requirement would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:suppressAutoHyphens />
</w:pPr>
```

The paragraph would then be exempted from hyphenation by a consumer at display time, regardless of the hyphenation settings for the document. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.35 suppressLineNumbers (Suppress Line Numbers for Paragraph)

This element specifies whether line numbers shall be calculated for lines in this paragraph by the consumer when line numbering is requested using the lnNumType element (§17.6.8) in the paragraph's parent section settings. This element specifies whether the current paragraph's lines should be exempted from line numbering which is applied by the consumer on this document, not just suppressing the display of the numbering, but removing these lines from the line numbering calculation.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the default line number settings for the section, as specified in the lnNumType element shall apply to each line of this paragraph.

©ISO/IEC 2012 – All rights reserved

[*Example*: Consider a document with three paragraphs, each of which are displayed on five lines , all contained in a section which has the lnNumType element specified. If the second paragraph should be exempted from that line numbering, this requirement would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:suppressLineNumbers />
</w:pPr>
```

The paragraph would then be exempted from line by a consumer at display time, which would result in paragraph one using line numbers one through five, the second paragraph having no line numbers, and the third paragraph using line numbers six through ten. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.3.1.36 suppressOverlap (Prevent Text Frames From Overlapping)

This element specifies whether a text frame which intersects another text frame at display time shall be allowed to overlap the contents of the other text frame. If a text frame cannot overlap other text frames, it shall be repositioned when displayed to prevent this overlap as needed.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then overlap shall be allowed between a text frame which intersects another text frame displayed at the same location.

[*Example*: Consider a document with two text frames which are allowed to overlap each other. If the second text frame should overlap the contents of another text frame, that constraint would be specified via the following WordprocessingML:

```
<w:p>
  …
</w:p>
<w:p>
  <w:pPr>
    <w:framePr … />
    <w:suppressOverlap />
  </w:pPr>
  …
</w:p>
```

The resulting text frame with the suppressOverlap property specified would never overlap any intersecting text frames. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.3.1.37 tab (Custom Tab Stop)

This element specifies a single custom tab stop defined within a set of paragraph properties in a document. A tab stop location shall always be measured relative to the leading edge of the paragraph in which it is used (that is, the left edge for a left-to-right paragraph, and the right edge for a right-to-left paragraph).

[*Example*: Consider a custom tab stops at 1.5" in a WordprocessingML document. This tab stop would be contained within a tab element defining the tab stop as follows:

```
<w:tab w:val="start" w:pos="2160" />
```

The tab element specifies all of the properties for the custom tab stop for the current paragraph property set. *end example*]

Attributes	Description
leader (Tab Leader Character)	Specifies the character which shall be used to fill in the space created by a tab which ends at this custom tab stop. This character shall be repeated as required to completely fill the tab spacing generated by the tab character. If this attribute is omitted, then no tab leader character shall be used. [*Example*: Consider a tab stop which should be preceded by a sequence of underscore characters, as follows: _______________Text at the tab stop This tab stop would have a leader attribute value of `underscore`, indicating that the tab stop must be preceded by underscore characters as needed to fill the tab spacing. *end example*] The possible values for this attribute are defined by the ST_TabTlc simple type (§17.18.85).
pos (Tab Stop Position)	Specifies the position of the current custom tab stop with respect to the current page margins. Negative values are permitted and move the tab stop into the current page margin the specified amount. [*Example*: Consider a custom tab stops at 1.5" in a WordprocessingML document. This tab stop would be contained within a tab element defining the tab stop as follows: `<w:tab w:val="start" w:pos="2160" />` The pos attribute specifies that this custom tab stop must be located `2160` points (1.5 inches) inside the starting text margin. *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple

Attributes	Description
	type (§17.18.81).
val (Tab Stop Type)	Specifies the style of custom tab stop, which determines the behavior of the tab stop and the alignment which shall be applied to text entered at the current custom tab stop. The value of `clear` is unique and specifies that this tab stop shall be removed when the document is next edited by a consumer which supports rendering the document contents. [*Example*: Consider a custom tab stops at 1.5" in a WordprocessingML document. This tab stop would be contained within a tab element defining the tab stop as follows: `<w:tab w:val="start" w:pos="2160" />` The val attribute specifies that this custom tab stop must align all text entered at its location to its left. *end example*] The possible values for this attribute are defined by the ST_TabJc simple type (§17.18.84).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TabStop) is located in §A.1. *end note*]

17.3.1.38 tabs (Set of Custom Tab Stops)

This element specifies a sequence of custom tab stops which shall be used for any tab characters in the current paragraph.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no custom tab stops shall be used for this paragraph.

As well, this property is additive - tab stops at each level in the style hierarchy are added to each other to determine the full set of tab stops for the paragraph. A hanging indent specified via the hanging attribute on the ind element (§17.3.1.12) shall also always implicitly create a custom tab stop at its location.

[*Example*: Consider a paragraph which contains two custom tab stops at 1.5" and 3.5", respectively. These two tab stops would be contained within a tabs element defining the set of tab stops of the paragraph as follows:

```
<w:pPr>
  <w:tabs>
    <w:tab w:val="start" w:pos="2160" />
    <w:tab w:val="start" w:pos="5040" />
  </w:tabs>
</w:pPr>
```

The tabs element specifies all of the customized tab stops for the current paragraph. *end example*]

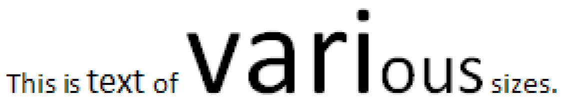
[*Note*: The W3C XML Schema definition of this element's content model (CT_Tabs) is located in §A.1. *end note*]

17.3.1.39 textAlignment (Vertical Character Alignment on Line)

This element specifies the vertical alignment of all text on each line displayed within a paragraph. If the line height (before any added spacing) is larger than one or more characters on the line, all characters are aligned to each other as specified by this element.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the vertical alignment of all characters on the line shall be automatically determined by the consumer.

[*Example*: Consider a paragraph of text of different font sizes, as follows:

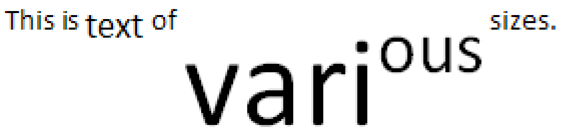


This is text of various sizes.

If the text on this paragraph must be aligned based on the top point of the maximum character height, that requirement would be specified as follows in the WordprocessingML:

```
<w:pPr>
  <w:textAlignment w:val="top" />
</w:pPr>
```

The resulting text would be top aligned, as follows:



This is text of various sizes.

The characters are all aligned to the maximum character extent on the line. *end example*]

Attributes	Description

val (Vertical Character Alignment Position)	Specifies the style of vertical alignment which shall be used to align the characters on each line in the current paragraph.
	[*Example*: Consider a paragraph of text of different font sizes which must be aligned based on the baseline point of each character in each line. This requirement would be specified as follows in the WordprocessingML:
	`<w:pPr>`
	`  <w:textAlignment w:val="baseline" />`

Attributes	Description
	`</w:pPr>` The resulting text would be aligned to the baseline for each character on the line. *end example*] The possible values for this attribute are defined by the ST_TextAlignment simple type (§17.18.91).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextAlignment) is located in §A.1. *end note*]

17.3.1.40 textboxTightWrap (Allow Surrounding Paragraphs to Tight Wrap to Text Box Contents)

This element specifies whether, for paragraphs in a text box, the surrounding text shall be allowed to overlap with the empty text box boundaries and tight wrap to the extents of the text within the text box.

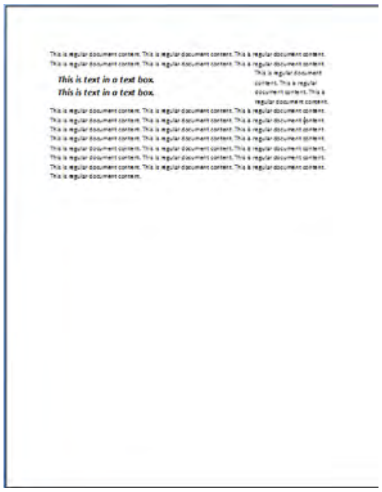
This element shall only be read for paragraphs which are contained within a text box (have a txbxContent ancestor), ignored otherwise.

If the parent text box does not meet the following three criteria, then this property has no effect:

- The text box wrapping shall be set to `tight`
- The text box border shall not be set
- The text box shading shall not be set

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then paragraphs in a text box have no tight wrapping overrides, and text shall wrap to the extents of the text box.

[*Example*: Consider a document with a tight wrapped text box which extends two-thirds of the way across the page, as follows:

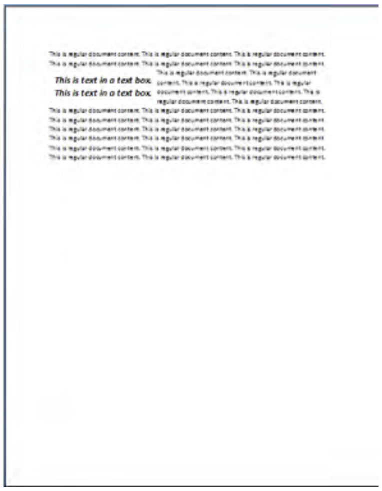


This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is text in a text box.
This is text in a text box.
This is regular document
content. This is regular
document content. This is
regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content.

The surrounding text is tightly wrapped to the extents of the text box. If the consumer must tight wrap to the extents of the text, that requirement would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:textboxTightWrap w:val="allLines" />
</w:pPr>
```

This would result in the following display of the content:



This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is text in a text box.
This is text in a text box.
This is regular document content. This is regular document
content. This is regular document content. This is regular
document content. This is regular document content. This is
regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.
This is regular document content. This is regular document content. This is regular document content.

The resulting paragraphs within the textbox use the textboxTightWrap element to specify that text should be tightly wrapped to the paragraph's extents. *end example*]

Attributes	Description
val (Lines to Tight Wrap to Paragraph Extents)	Specifies the lines in the parent paragraph which shall allow the text to be tight wrapped to the paragraph (and not the text box) extents when displaying the document. [*Example*: Consider a paragraph in a text box which meets the criteria specified above which must allow wrapping to the text extents on its first line only. That requirement

Attributes	Description

	would be specified using the following WordprocessingML:
	`<w:pPr>`
	`  <w:textboxTightWrap w:val="firstLineOnly" />`
	`</w:pPr>`
	The resulting paragraph would allow text to tightly wrap to the contents of its first line only. All other lines would wrap to the text box's extents. *end example*]
	The possible values for this attribute are defined by the ST_TextboxTightWrap simple type (§17.18.92).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextboxTightWrap) is located in §A.1. *end note*]

17.3.1.41 textDirection (Paragraph Text Flow Direction)

This element specifies the direction of the text flow for this paragraph.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the paragraph shall inherit the text flow settings from the parent section.

[*Example*: Consider a document with a paragraph in which text must be oriented vertically, flowing from left to right horizontally on the page.This setting would be specified with the following WordprocessingML:

```
<w:pPr>
  <w:textDirection w:val="lr" />
</w:pPr>
```

The textDirection element specifies via the `lr` value in the val attribute that the text flow must be oriented vertically, with subsequent lines stacked from left to right. *end example*]

Attributes	Description

val (Direction of Text Flow)	Specifies the direction of the text flow for this object.
	[*Example*: Consider a document with a section in which text be oriented vertically, flowing from left to right horizontally on the page.This setting requires the following WordprocessingML:
	`<w:sectPr>`
	`  …`
	`  <w:textDirection w:val="lr" />`
	`</w:sectPr>`

Attributes	Description
	The textDirection element specifies via the lr value in the val attribute that the text flow must be oriented vertically, with subsequent lines stacked from left to right. *end example*] The possible values for this attribute are defined by the ST_TextDirection simple type (§17.18.93).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextDirection) is located in §A.1. *end note*]

17.3.1.42 top (Paragraph Border Above Identical Paragraphs)

This element specifies the border which shall be displayed above a set of paragraphs which have the same set of paragraph border settings.

To determine if any two adjoining paragraphs shall have an individual top and bottom border or a between border, the set of borders on the two adjoining paragraphs are compared. If the border information on those two paragraphs is identical for all possible paragraphs borders, then the between border is displayed. Otherwise, the final paragraph shall use its bottom border and the following paragraph shall use its top border, respectively. If this border specifies a space attribute, that value determines the space above the text (ignoring any spacing above) which should be left before this border is drawn, specified in points.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then no between border shall be applied above identical paragraphs.

[*Example*: Consider the following two paragraphs' WordprocessingML definition:

```
<w:p>
  <w:pPr>
    <w:pBdr>
      <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
w:themeColor="accent6" w:themeShade="BF" />
      <w:bottom w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
w:themeColor="accent2" w:themeTint="33" />
      <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
w:themeColor="accent3" w:themeTint="99" />
      <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
w:themeColor="accent1" />
    </w:pBdr>
  </w:pPr>
```

```
    <w:r>
      <w:t>First paragraph.</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:pPr>
      <w:pBdr>
        <w:top w:val="single" w:sz="24" w:space="1" w:color="F2DCDB"
  w:themeColor="accent2" w:themeTint="33" />
        <w:left w:val="single" w:sz="24" w:space="4" w:color="B97034"
  w:themeColor="accent6" w:themeShade="BF" />
        <w:bottom w:val="single" w:sz="24" w:space="0" w:color="F2DCDB"
  w:themeColor="accent2" w:themeTint="33" />
        <w:right w:val="single" w:sz="24" w:space="4" w:color="C3D69B"
  w:themeColor="accent3" w:themeTint="99" />
        <w:between w:val="single" w:sz="24" w:space="1" w:color="4F81BD"
  w:themeColor="accent1" />
      </w:pBdr>
    </w:pPr>
    <w:r>
      <w:t>Second paragraph.</w:t>
    </w:r>
  </w:p>
```

Since the paragraph border is different between the two paragraphs (the bottom space value goes from 1 to 0), paragraph two uses its top border, which is located one point above the text in that paragraph. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.3.1.43 topLinePunct (Compress Punctuation at Start of a Line)

This element specifies whether punctuation shall be compressed when it appears as the first character in a line, allowing subsequent characters on the line to be move in accordingly.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then punctuation shall not be compressed in this paragraph, even when it appears at the start of a line.

[*Example*: Consider a paragraph which should allow punctuation at the start of a line to be compressed, in order to prevent it from taking up unnecessary space. This constraint is specified using the following WordprocessingML:

```
  <w:pPr>
    <w:topLinePunct w:val="on" />
```

```
</w:pPr>
```

The topLinePunct element specifies that this compression must be allowed when displaying this paragraph. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

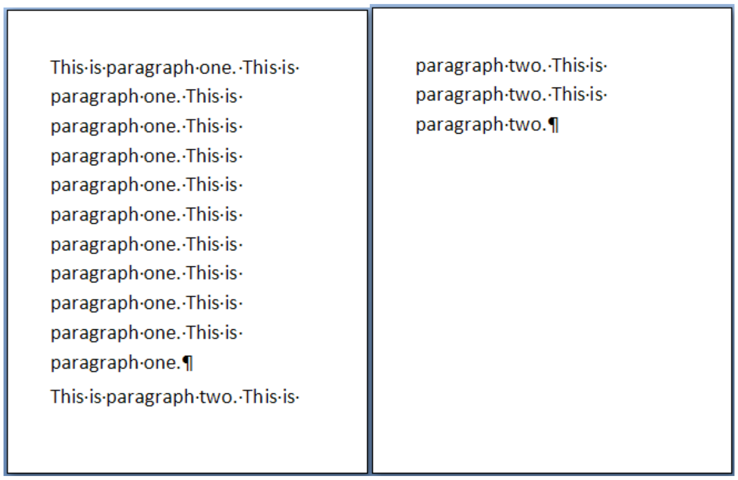
17.3.1.44 widowControl (Allow First/Last Line to Display on a Separate Page)

This element specifies whether a consumer shall prevent a single line of this paragraph from being displayed on a separate page from the remaining content at display time by moving the line onto the following page.

When displaying a paragraph in a page, it is sometimes the case that the first line of that paragraph would display as the last line on one page, and all subsequent lines would display on the following page. This property ensures that a consumer shall move the single line to the following page as well to prevent having one line on its own page. As well, if a single line appears at the top of a page, a consumer shall move the preceding line onto the following page as well, to prevent a single line from being displayed on a separate page.

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this paragraph shall prevent a single line from being shown on a separate page whenever it would normally occur.

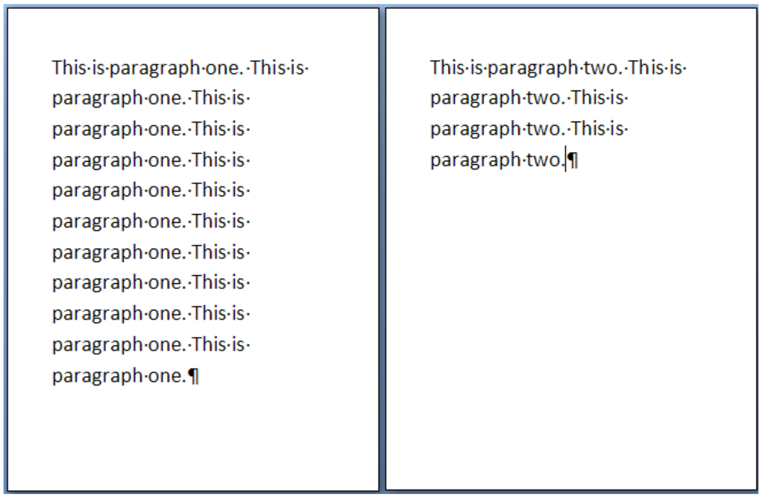
[*Example*: Consider a document with a paragraph which must be shown on four lines at display time. If this paragraph would normally be laid out with its first line at the bottom of one page, and its following lines on the next page, as follows:



This·is·paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.¶
This·is·paragraph·two.·This·is·

paragraph·two.·This·is·
paragraph·two.·This·is·
paragraph·two.¶

This property would ensure that the default behavior for each paragraph prevented this, by moving this line onto the following paragraph as follows:



This·is·paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.·This·is·
paragraph·one.¶

This·is·paragraph·two.·This·is·
paragraph·two.·This·is·
paragraph·two.·This·is·
paragraph·two.¶

However, if this default is overridden by specifying the following WordprocessingML:

```
<w:pPr>
  <w:widowControl w:val="off" />
</w:pPr>
```

The specifying of the widowControl element with value `off` means that the consumer displaying this document must not move the first line onto a separate page if it would be separated from all other lines (the first picture above). *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.1.45 wordWrap (Allow Line Breaking At Character Level)

This element specifies whether a consumer shall break text which exceeds the text extents of a line by breaking the word across two lines (breaking on the character level) or by moving the word to the following line (breaking on the word level). [*Note*: Applications are discouraged from breaking text when it would change the semantics or appearance of the content. *end note*]

If this element is omitted on a given paragraph, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then this paragraph shall break words in space-delimited languages at the word level, not the character level when it is displayed.

[*Example*: Consider a paragraph whose first line ends with the word `world`, where the text extents for that line would normally fall between the letter o and the letter r. If this element is omitted, a producer would normally move the entire word `world` to the following line, since the word does not fit within the first line's text extents. However, if this document should allow words to be broken at the character level, that constraint would be specified as follows:

```
<w:pPr>
  <w:wordWrap w:val="off" />
```

```
</w:pPr>
```

The resulting paragraph specifies that wordWrap is turned off, therefore the word "world" would be broken into two lines between the exact two characters (o and r) that match the text extents. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2 Run

The next level of the document hierarchy is the *run*, which defines a region of text with a common set of properties. A run is represented by an r element (§17.3.2.25), which allows the producer to specify a single set of formatting properties, applying the same information to all the contents of the run.

Just as a paragraph can have properties, so too can a run. All of the elements inside an r element have their properties controlled by a corresponding optional rPr run properties element (§17.7.9.1; §17.3.2.27), which shall be the first child of the r element. In turn, the rPr element is a container for a set of property elements that are applied to the rest of the children of the r element. [*Note*: The elements inside the rPr container element allow the consumer to control whether the content in the following run content is bold, underlined, or visible, for example. *end note*]

[*Example*: Consider the following run within a WordprocessingML document:

```
<w:r>
  <w:rPr>
    <w:b/>
    <w:i/>
  </w:rPr>
  <w:t>quick</w:t>
</w:r>
```

The run specifies two formatting properties in its run contents: bold and italic. These properties are therefore applied to all content within this run. *end example*]

17.3.2.1 b (Bold)

This element specifies whether the bold property shall be applied to all non-complex script characters in the contents of this run when displayed in a document.

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then bold shall not be applied to non-complex script characters.

[*Example*: Consider a run of text which must have the b property explicitly turned off for the non complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:b w:val="false"/>
</w:rPr>
```

This run explicitly declares that the b property is `false` for the non-complex script contents of this run. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.2 bCs (Complex Script Bold)

This element specifies whether the bold property shall be applied to all complex script characters in the contents of this run when displayed in a document.

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then bold shall not be applied to complex script characters.

[*Example*: Consider a run of text which must have the bCs property (bold) explicitly turned on for the complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:bCs w:val="true"/>
</w:rPr>
```

This run explicitly declares that the bCs property is `true` , so bold is turned on for the complex script contents of this run. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.3 bdo (Bidirectional Override)

This element specifies a directional override, which shall be applied as described by the Bidirectional Algorithm (cf. Unicode Technical Report #9). [*Note*: The presence of this markup is functionally equivalent to the presence of a LRO/RLO character at the location of the start element, and a corresponding PDF character at the location of the end element in a string of Unicode text. *end note*]

[*Example*: The right-to-left override can be used to force a product part number made of mixed English, digits, and Hebrew text written from right-to-left, as follows. (Assume lower-case text is English and the upper-case text is Arabic or Hebrew):

```
<w:r>
    <w:t xml:space="preserve">part number: </w:t>
</w:r>
<w:bdo w:val=”rtl”>
    <w:r>
```

```
          <w:t>ad-326D-FG</w:t>
      </w:r>
  </w:bdo>
```

This generates the following text layout:

```
  part number: GF-D623-da.
```

The bidirectional override forces all characters to be displayed right-to-left, regardless of their classification. *end example*]

Attributes	Description

val (Direction of Override)	Specifies the direction of the override being applied.
	If this attribute is omitted, the override shall be assumed to be `ltr`.
	[*Example*: For example, consider the following bidirectional override:
	`<w:bdo w:val="rtl">`
	`…`
	`</w:bdo>`
	The val attribute explicitly declares that the override is applied right to left. *end example*]
	The possible values for this attribute are defined by the ST_Direction simple type (§17.18.12).

[*Note*: The W3C XML Schema definition of this element's content model (CT_BdoContentRun) is located in §A.1. *end note*]

17.3.2.4 bdr (Text Border)

This element specifies information about the border applied to the text in the current run.

The first piece of information specified by the bdr element is that the current shall have a border when displayed. This information is specified simply by the presence of the bdr element in run's properties.

The second piece of information concerns the set of runs which share the current run border. This is determined based on the attributes on the bdr element. If the set of attribute values specifies on two adjacent runs is identical, then those two runs shall be considered to be part of the same run border group and rendered within the same set of borders in the document.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then no run border shall be applied to the text in this run.

©ISO/IEC 2012 – All rights reserved

[*Example*: Consider a document in which the following two runs are located adjacent to one another:

```
<w:r>
  <w:rPr>
    <w:bdr w:val="single" w:sz="36" w:space="0" w:color="B8CCE4"
w:themeColor="accent1" w:themeTint="66" />
  </w:rPr>
  <w:t xml:space="preserve">run one</w:t>
</w:r>
<w:r >
  <w:rPr>
    <w:b />
    <w:bdr w:val="single" w:sz="36" w:space="0" w:color="B8CCE4"
w:themeColor="accent1" w:themeTint="66" />
  </w:rPr>
  <w:t>run two</w:t>
</w:r>
```

These two runs, although each is distinct, are combined when rendering the text border because the bdr elements are identical between the two runs. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.3.2.5 caps (Display All Characters As Capital Letters)

This element specifies that any lowercase characters in this text run shall be formatted for display only as their capital letter character equivalents. This property does not affect any non-alphabetic character in this run, and does not change the Unicode character for lowercase text, only the method in which it is displayed.

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the characters are not formatted as capital letters.

This element shall not be present with the smallCaps (§17.3.2.33) property on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider the words `Hello, world`, which must be displayed in all capital letters in a document. This constraint is specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:caps w:val="true" />
  </w:rPr>
  <w:t>Hello, world</w:t>
```

```
</w:r>
```

This run displays as `HELLO, WORLD`, even though the lowercase characters are used in the run contents due to the use of the caps element. If this property is removed, the original character forms is displayed (they are not lost). *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.6 color (Run Content Color)

This element specifies the color which shall be used to display the contents of this run in the document.

This color can be explicitly specified, or set to allow the consumer to automatically choose an appropriate color based on the background color behind the run's content.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the characters are set to allow the consumer to automatically choose an appropriate color based on the background color behind the run's content.

[*Example*: Consider a run of text which should be displayed using the `accent3` theme color from the document's Theme part. This requirement would be specified as follows in the resulting WordprocessingML:

```
<w:rPr>
  <w:color w:themeColor="accent3" />
</w:rPr>
```

The color attribute specifies that the run shall use the `accent3` theme color. *end example*]

Attributes	Description

themeColor (Run Content Theme Color)	Specifies a theme color which should be applied to the current run.
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part, which allows for color information to be set centrally in the document.
	If the themeColor attribute is specified, then the val attribute is ignored for this run.
	[*Example*: Consider a run of text which should be displayed using the `accent3` theme color from the document's Theme part. This requirement would be specified as follows in the resulting WordprocessingML:
	`<w:rPr>`
	`  <w:color w:themeColor="accent3" />`
	`</w:rPr>`
	The color attribute specifies that the run must use the `accent3` theme color. *end*

Attributes	Description
	example] The possible values for this attribute are defined by the ST_ThemeColor simple type (§17.18.97).

themeShade (Run Content Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this run's contents.
	
	If the themeShade is supplied, then it is applied to the RGB value of the theme color to determine the final color applied to this run.
	
	The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border.
	
	[*Example*: Consider a shade of 40% applied to a run in a document. This shade is calculated as follows:
	
	$\begin{aligned} S_{xml} &= 0.4 * 255 \\ &= 102 \\ &= 66(hex) \end{aligned}$
	
	Te resulting themeShade value in the file format would be 66. *end example*]
	
	Given a input red, green, or blue color value C (from 0-255), an output color value of C' (from 0-255), and a shade value S (from 0-100), the shade is applied as follows:
	
	$C' = \left(1 - \frac{S}{100}\right) \mathrm{C}$
	
	[*Example*: Consider a document with a run using the `accent6` theme color, whose RGB value (in RRGGBB hex format) is F79646.
	
	The hex value for the green component is 96 - 150 in decimal. Applying the shade formula with shade of 50%, the output decimal value of the green component is 75, or a hex value of 4B. This transformed value can be seen in the resulting run color WordprocessingML's val attribute:
	
	`<w:color w:val="7B4B23" w:themeColor="accent6"`
	`w:themeShade="80" />`
	
	end example]
	
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

themeTint (Run Content Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this run's contents. If the themeTint is supplied, then it is applied to the RGB value of the theme color to

Attributes	Description
	determine the final color applied to this run. The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a run in a document. This tint is calculated as follows: $\begin{aligned} T_{xml} &= 0.6 * 255 \\ &= 153 \\ &= 99(hex) \end{aligned}$ The resulting themeTint value in the file format would be 99. *end example*] Given a input red, green, or blue color value C (from 0-255), an output color value of C' (from 0-255), and a tint value T (from 0-100), the tint is applied as follows: $C' = \left(1 - \frac{T}{100}\right)(255 - \mathrm{C}) + \mathrm{C}$ [*Example*: Consider a document with a run using the accent1 theme color, whose RGB value (in RRGGBB hex format) is `C0504D`. The hex value for the green component is `50` - 80 in decimal. Applying the tint formula with tint of 60%, the output decimal value of the green component is 150, or a hex value of 96. This transformed value can be seen in the resulting run color's WordprocessingML val attribute: `<w:color w:val="D99694" w:themeColor="accent1" w:themeTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).
val (Run Content Color)	Specifies the color for this run. This color can either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the run color as appropriate. If the run specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value. [*Example*: Consider a run color with value `auto`, as follows: `<w:rPr>`

Attributes	Description

	`<w:color … w:val="auto" />`
	`</w:rPr>`
	This color therefore can be automatically be modified by a consumer as appropriate, for example, in order to ensure that the run contents can be distinguished against the page's background color. *end example*]
	The possible values for this attribute are defined by the ST_HexColor simple type (§17.18.38).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Color) is located in §A.1. *end note*]

17.3.2.7 cs (Use Complex Script Formatting on Run)

This element specifies whether the contents of this run shall be treated as complex script text regardless of their Unicode character values when determining the formatting for this run.

This means that a consumer shall use the complex script formatting applied to the run [*Example*: The bCs value (§17.3.2.2), not the b value (§17.3.2.1). *end example*] when determining the resulting formatting properties.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the run contents are set to complex script based on the Unicode character positions of the content.

[*Example*: Consider the following run of English text in a WordprocessingML document:

```
<w:r>
  <w:rPr>
    <w:bCs/>
    <w:i/>
    <w:cs/>
  </w:rPr>
  <w:t>some English text</w:t>
</w:r>
```

This run has bold applied to complex script characters, and italics applied to non-complex script characters. However, since the cs property is set, the text in this run must be treated as complex script text when determining the resulting formatting. Therefore, the run has bold formatting, but no italic formatting when displayed. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.8 dir (Bidirectional Embedding Level)

This element specifies that the embedding level of its contents shall be increased, as described by the Bidirectional Algorithm (cf. Unicode Technical Report #9). Unlike the rtl element (§17.3.2.30), this element can be nested and causes the embedding level to increase appropriately. Within this element, the semantics for the rtl element shall continue apply, specifying a high-level override of the character classification applied at the appropriate embedding level. [*Note*: The presence of this markup is functionally equivalent to the presence of a LRE/RLE character at the location of the start element, and a corresponding PDF character at the location of the end element in a string of Unicode text. *end note*]

[*Example*: Consider a left-to-right paragraph that has a quote in a right-to-left language, but the quote itself includes left-to-right text. In this case, the text layout within the quote should be read from right-to-left. Adding this element around the quote increases the embedding level and result in the desired layout. (Assume the lower-case text is English and the upper-case text is Arabic or Hebrew):

```
<w:r>
    <w:t>he said: “</w:t>
</w:r>
<w:dir w:val=”rtl”>
    <w:r>
        <w:rPr>
            <w:rtl />
        </w:rPr>
        <w:t>I LEAVE FOR </w:t>
    </w:r>
    <w:r>
        <w:t>united states</w:t>
    </w:r>
    <w:r>
        <w:rPr>
            <w:rtl />
        </w:rPr>
        <w:t xml:space="preserve"> TOMORROW</w:t>
    </w:r>
</w:dir>
<w:r>
    <w:t>”.</w:t>
</w:r>
```

The alternating RTL and LTR elements help in resolving the classification of characters within this run as follows:

```
he said: “I LEAVE FOR united states TOMORROW”.
LLLLLLLLLLRRRRRRRRRRRRLLLLLLLLLLLLLRRRRRRRRRLL
```

But the fact that the quote is enclosed in an RTL embedding; the following embedding levels are generated:

```
    he said: “I LEAVE FOR united states TOMORROW”.
    00000000001111111111112222222222222211111111100
```

This generates the following text layout:

```
    he said: “WORROMOT united states ROF EVAEL I”.
```

end example]

Attributes	Description

val (Direction of Embedding)	Specifies the direction of the embedding being applied.
	If this attribute is omitted, the embedding shall be assumed to be `ltr`.
	[*Example*: For example, consider the following bidirectional embedding:
	`<w:dir w:val="rtl">`
	`  …`
	`</w:dir>`
	The val attribute explicitly declares that the embedding is applied right to left. *end example*]
	The possible values for this attribute are defined by the ST_Direction simple type (§17.18.12).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_DirContentRun) is located in §A.1. *end note*]

17.3.2.9 dstrike (Double Strikethrough)

This element specifies that the contents of this run shall be displayed with two horizontal lines through each character displayed on the line.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then double strikethrough shall not be applied to the contents of this run.

This element shall not be present with the strike (§17.3.2.37) property on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which must have the dstrike property explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:dstrike w:val="true"/>
</w:rPr>
```

This run explicitly declares that the dstrike property is `true`, so the contents of this run has two horizontal strikethrough lines. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.10 eastAsianLayout (East Asian Typography Settings)

This element specifies any East Asian typography settings which shall be applied to the contents of the run.

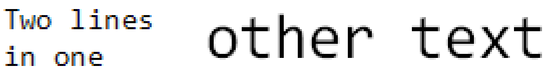
The specific typography settings represented by this element include the *two lines in one* and *horizontal in vertical* text options.

The *two lines in one* setting specifies that the characters in this run should be written out on a single line in the document by creating two sub-lines within the regular line, and laying out this text equally between those sub lines.

[*Example*: Consider a paragraph with the text `two lines in one`, which must be displayed within a single logical line in the document. This constraint would be specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:eastAsianLayout w:id="1" w:combine="on" />
  </w:rPr>
  <w:t>two lines in one</w:t>
</w:r>
```

The resulting text would be displayed on two sub lines within the other text on this line, like this:



Two lines
in one
other text

end example]

The *horizontal in vertical* setting specifies that characters in this run should be rendered with a 90 degree rotation to the left from all other contents of the line when displayed in the document, while keeping the text on the same line as all other text in the paragraph.

[*Example*: Consider a paragraph with the text `this word is vertical`, of which the word `vertical` must be displayed vertically within the document. This constraint would be specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:eastAsianLayout w:id="2" w:vert="on" />
  </w:rPr>
  <w:t>vertical</w:t>
```

©ISO/IEC 2012 – All rights reserved

```
</w:r>
```

The resulting text would be displayed with a 90 degree rotation from the other text content. *end example*]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>combine (Two Lines in One)</td><td>

Specifies whether the contents of the current run should be combined into one line using the two lines in one logic described above in the parent element.

If this attribute is omitted, then this run shall not be displayed on two sub lines.

[*Example*: Consider a paragraph with the text `two lines in one`, which must be displayed within a single logical line in the document. This constraint would be specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:eastAsianLayout w:id="1" w:combine="on" />
  </w:rPr>
  <w:t>two lines in one</w:t>
</w:r>
```

The resulting text would be displayed on two sub lines within the other text on this line. *end example*]

The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

</td></tr>
<tr><td>combineBrackets (Display Brackets Around Two Lines in One)</td><td>

Specifies that the two lines in one text should be enclosed within a pair of brackets when displayed. This attribute's values determine the bracket style to put around combined text.

If this attribute is not specified, then no brackets shall be placed around this content when displayed in the document. If the combine attribute is not specified, then this attribute is ignored.

[*Example*: Consider a paragraph with the text `two lines in one`, which must be displayed within a single logical line in the document and enclosed in curly brackets. This constraint would be specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:eastAsianLayout w:id="1" w:combine="on"
w:combineBrackets="curly"/>
  </w:rPr>
  <w:t>two lines in one</w:t>
</w:r>
```

The resulting text would be displayed on two sub lines within the other text on this line and enclosed within curly brackets when displayed. *end example*]

</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

The possible values for this attribute are defined by the ST_CombineBrackets simple type (§17.18.8).

</td></tr>
<tr><td>id (East Asian Typography Run ID)</td><td>

Specifies a unique ID which shall be used to link multiple runs containing eastAsianLayout element to each other to ensure that their contents are correctly displayed in the document.

This means that multiple runs which are broken apart due to differences in formatting can be identified as belonging to the same grouping in terms of eastAsianLayout properties, although they are separated into multiple runs of text.

[*Example*: Consider the following three runs in a document:

```
<w:r>
  <w:rPr>
    <w:eastAsianLayout w:id="-1552701694" w:combine="lines"
w:combineBrackets="curly" />
  </w:rPr>
  <w:t>two</w:t>
</w:r>
<w:r>
  <w:rPr>
    <w:u w:val="single" w:color="4F81BD" w:themeColor="accent1"
/>
    <w:eastAsianLayout w:id="-1552701694" w:combine="lines"
w:combineBrackets="curly" />
  </w:rPr>
  <w:t>lines in</w:t>
</w:r>
<w:r>
  <w:rPr>
  <w:eastAsianLayout w:id="-1552701694" w:combine="lines"
w:combineBrackets="curly" />
  </w:rPr>
  <w:t>one</w:t>
</w:r>
```

Although there are three runs of content, all three regions must be combined into a single two lines in one region based on the identical value used in the id attribute for all three runs. *end example*]

The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

</td></tr>
<tr><td>vert (Horizontal in Vertical (Rotate Text))</td><td>

Specifies that characters in this run should be rendered with a 270 degree rotation to the left from all other contents of the line when displayed in the document as described above.

</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<p>If this attribute is omitted, then the contents of this run shall not be rotated with respect to the normal text flow.</p>
<p>[<em>Example</em>: Consider a paragraph with the text <code>this word is vertical</code>, of which the word <code>vertical</code> must be displayed vertically within the document. This constraint would be specified as follows in the WordprocessingML:</p>
<pre>
<w:r>
  <w:rPr>
    <w:eastAsianLayout w:id="2" w:vert="on" />
  </w:rPr>
  <w:t>vertical</w:t>
</w:r>
</pre>
<p>The resulting text would be displayed with a 270 degree rotation from the other text content. <em>end example</em>]</p>
<p>The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).</p>
</td></tr>
<tr><td>vertCompress (Compress Rotated Text to Line Height)</td><td>
<p>Specifies whether the rotated text shall be compressed at display time in order to ensure that it fits into the existing line height without increasing the overall height of the line.</p>
<p>If the vert attribute is not specified, then this attribute is ignored. If this attribute is omitted, then text shall not be compressed in order to fit into the existing height of the line when it is rotated.</p>
<p>[<em>Example</em>: Consider a paragraph with the text <code>this word is vertical</code>, of which the word <code>vertical</code> must be displayed vertically within the document but must not change the height of the line. This constraint would be specified as follows in the WordprocessingML:</p>
<pre>
<w:r>
  <w:rPr>
    <w:eastAsianLayout w:id="2" w:vert="true"
w:vertCompress="true" />
  </w:rPr>
  <w:t>vertical</w:t>
</w:r>
</pre>
<p>The resulting text would be compressed in order to fit the height of the line as defined by all non-compressed characters. <em>end example</em>]</p>
<p>The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).</p>
</td></tr>
</table>

[*Note*: The W3C XML Schema definition of this element's content model (CT_EastAsianLayout) is located in §A.1. *end note*]

17.3.2.11 effect (Animated Text Effect)

This element specifies an animated text effect which should be displayed when rendering the contents of this run. This effect is rendered around the extents of the text in the run in the same location as a run border with zero pixels of padding would be rendered (if such a run border was present).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then no text effect shall be applied to the contents of this run.

[*Example*: Consider a run of text which must have an animated text effect consisting of multiple colored flashing lights (see possible attribute values for descriptions of each effect). This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:effect w:val="lights"/>
</w:rPr>
```

This run explicitly declares that the effect property is `lights`, so the contents of this run has an animated lights text effect. *end example*]

Attributes	Description

val (Animated Text Effect Type)	Specifies the type of animated text effect which shall be applied to this text run.
	[*Example*: Consider a run of text which must have an animated text effect consisting of multiple colored flashing lights. This constraint is specified using the following WordprocessingML:
	`<w:rPr>`
	`  <w:effect w:val="lights"/>`
	`</w:rPr>`
	This run explicitly declares a type of text effect, using the val property, of `lights`, so the contents of this run has the animated lights text effect. *end example*]
	The possible values for this attribute are defined by the ST_TextEffect simple type (§17.18.94).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextEffect) is located in §A.1. *end note*]

©ISO/IEC 2012 – All rights reserved

17.3.2.12 em (Emphasis Mark)

This element specifies the emphasis mark which shall be displayed for each non-space character in this run. An *emphasis mark* is an additional character that is rendered above or below the main character glyph as specified by the contents of the val attribute.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then no emphasis mark shall be added to each character in the contents of this run.

[*Example*: Consider a run of text which must have a dot underneath each character as an emphasis mark. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:em w:val="dot"/>
</w:rPr>
```

This run explicitly declares that the emphasis mark type is dot, so the contents of this run has a dot emphasis mark above each character. *end example*]

Attributes	Description

val (Emphasis Mark Type)	Specifies the emphasis mark type used for each character in this run.
	[*Example*: Consider a run of text which must have a dot underneath each character as an emphasis mark. This constraint is specified using the following WordprocessingML:
	`<w:rPr>`
	`  <w:em w:val="dot"/>`
	`</w:rPr>`
	This run explicitly declares that the em type is dot, so the contents of this run have a dot emphasis mark beneath each character. *end example*]
	The possible values for this attribute are defined by the ST_Em simple type (§17.18.24).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Em) is located in §A.1. *end note*]

17.3.2.13 emboss (Embossing)

This element specifies that the contents of this run should be displayed as if embossed, which makes text appear as if it is raised off the page in relief.

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then embossing shall not be applied to the contents of this run.

This element shall not be present with either the imprint (§17.3.2.18) or outline (§17.3.2.23) properties on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which must have the emboss property explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:emboss w:val="true"/>
</w:rPr>
```

This run explicitly declares that the emboss property is `true`, so the contents of this run appear embossed. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.14 fitText (Manual Run Width)

This element specifies that the contents of this run shall not be automatically displayed based on the width of its contents, rather its contents shall be resized to fit the width specified by the val attribute. This expansion/contraction shall be performed by equally increasing/decreasing the size of each character in this run's contents when displayed.

If this element is omitted, then the contents of this run shall be displayed based on the size of its contents.

[*Example*: Consider a document with a run which must be displayed in exactly one-half inch of space, regardless of its contents. This constraint would be specified using the following WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:fitText w:id="50" w:val="720" />
  </w:rPr>
  <w:t>This text must be displayed in one-half of an inch.</w:t>
</w:r>
```

The resulting run contents must be displayed in exactly `720` twentieths of a point (one half of an inch) when displayed in a document. *end example*]

Attributes	Description
id (Fit Text Run ID)	Specifies a unique ID which shall be used to link multiple contiguous runs containing fitText elements to each other to ensure that their contents are correctly merged into the specified width in the document. This means that multiple runs which are broken apart due to differences in formatting can be identified as belonging to the same grouping in terms of fitText properties, although they are multiple runs of text in the WordprocessingML. If the runs are not contiguous, then the id attribute is ignored, and the runs are not

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>linked.

If this attribute is omitted, then this run has no id and shall not be linked with any other run in the parent paragraph.

[<i>Example</i>: Consider the following three runs in a document, which should be fit into exactly one inch at display time:
<pre>
<w:r>
  <w:rPr>
    <w:fitText w:id="99" w:val="1440" />
  </w:rPr>
  <w:t>fit this into</w:t>
</w:r>
<w:r>
  <w:rPr>
    <w:b/>
    <w:fitText w:id="99" w:val="1440" />
   </w:rPr>
  <w:t>one</w:t>
</w:r>
<w:r>
  <w:rPr>
    <w:fitText w:id="99" w:val="1440" />
   </w:rPr>
  <w:t>inch</w:t>
</w:r>
</pre>
Although there are three runs of content, all three regions must be combined into a single fit text region (e.g. they all fit into one inch, rather than one inch each) based on the identical value used in the id attribute for all three runs. <i>end example</i>]

The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).</td></tr>
<tr><td>val (Value)</td><td>This attribute specifies the exact width of space which this run shall be fit into when displayed in the document.

[<i>Example</i>: Consider a document with a run which must be displayed in exactly one-half inch of space, regardless of its contents. This constraint would be specified using the following WordprocessingML:
<pre>
<w:r>
  <w:rPr>
    <w:fitText w:id="50" w:val="720" />
  </w:rPr>
  <w:t>This text must be displayed in one-half of an
inch.</w:t>
</pre></td></tr>
</table>

Attributes	Description
	`</w:r>` The resulting run contents must be displayed in exactly `720` twentieths of a point (one half of an inch) when displayed in a document. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FitText) is located in §A.1. *end note*]

17.3.2.15 highlight (Text Highlighting)

This element specifies a highlighting color which is applied as a background behind the contents of this run.

If this run has any background shading specified using the shd element (§17.3.2.32), then the background shading shall be superseded by the highlighting color when the contents of this run are displayed.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then text highlighting shall not be applied to the contents of this run.

[*Example*: Consider a run within a paragraph which has run shading applied as well as yellow text highlighting using the highlight element. This formatting is specified using the following WordprocessingML:

```
<w:rPr>
  <w:highlight w:val="yellow" />
  <w:shd w:themeFill="accent2" w:themeFillTint="66" />
</w:rPr>
```

The resulting run would have `yellow` highlighting visible over its contents, as the highlighting supersedes the shading for the contents of the run. *end example*]

Attributes	Description

val (Highlighting Color)	Specifies the color of the text highlighting which shall be applied to the contents of this run.
	[*Example*: Consider a text run which must be displayed with colored text highlighting. This highlighting would be specified using the following WordprocessingML:
	`<w:rPr>`
	`  <w:highlight w:val="red" />`
	`</w:rPr>`
	The resulting text highlighting would be `red`, as this is the color specified by the val

Attributes	Description
	attribute. *end example*] The possible values for this attribute are defined by the ST_HighlightColor simple type (§17.18.40).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Highlight) is located in §A.1. *end note*]

17.3.2.16 i (Italics)

This element specifies whether the italic property should be applied to all non-complex script characters in the contents of this run when displayed in a document.

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy* .If this element is never applied in the style hierarchy, then italics shall not be applied to non-complex script characters.

[*Example*: Consider a run of text which must have the i property explicitly turned on for the non-complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:i />
</w:rPr>
```

This run explicitly declares that the i property is `true` for the non-complex script contents of this run. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.17 iCs (Complex Script Italics)

This element specifies whether the italic property should be applied to all complex script characters in the contents of this run when displayed in a document.

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then italics shall not be applied to complex script characters.

[*Example*: Consider a run of text which must have the iCs property explicitly turned on for the complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:iCs w:val="true"/>
</w:rPr>
```

This run explicitly declares that the iCs property is `true`, so italics are turned on for the complex script contents of this run. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.3.2.18 imprint (Imprinting)

This element specifies that the contents of this run should be displayed as if imprinted, which makes text appear to be imprinted or pressed into page (also referred to as 'engrave').

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then imprinting shall not be applied to the contents of this run.

This element shall not be present with either the emboss (§17.3.2.13) or outline (§17.3.2.23) properties on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which must have the imprint property explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:imprint w:val="true"/>
</w:rPr>
```

This run explicitly declares that the imprint property is `true`, so the contents of this run appear imprinted. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.3.2.19 kern (Font Kerning)

This element specifies whether font kerning shall be applied to the contents of this run. If it is specified, then kerning shall be automatically adjusted when displaying characters in this run as needed.

The val attribute specifies the smallest font size which shall have its kerning automatically adjusted if this setting is specified. If the font size in the sz element (§17.3.2.38) is smaller than this value, then no font kerning shall be performed.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then font kerning shall not be applied to the contents of this run.

[*Example*: Consider the following WordprocessingML run which has font kerning properties specified:

```
<w:r>
  <w:rPr>
    <w:kern w:val="28" />
    <w:sz w:val="22" />
  </w:rPr>
</w:r>
```

Even though font kerning is turned on via the kern element, the contents of this run must not be kerned because that settings only applied to font sizes of 14 points (28 half-points) or larger. If the kern element's val attribute was less than or equal to the sz element's val attribute, then kerning would be applied:

```
<w:r>
  <w:rPr>
    <w:kern w:val="22" />
    <w:sz w:val="22" />
  </w:rPr>
</w:r>
```

end example]

Attributes	Description
val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch). The contents of this attribute value are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment: `<… w:val="30" />` The value in the val attribute is 30, which is equivalent to 15 points (30 half-points). This value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_HpsMeasure simple type (§17.18.42).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_HpsMeasure) is located in §A.1. *end note*]

17.3.2.20 lang (Languages for Run Content)

This element specifies the languages which shall be used to check spelling and grammar (if requested) when processing the contents of this run.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the languages for the contents of this run shall be automatically determined based on their contents using any method desired.

[*Example*: Consider a run which contains both Latin and complex script characters in its contents. If those contents should be interpreted as French (Canada) and Hebrew, respectively, that requirement would be specified as follows in the resulting WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:lang w:val="fr-CA" w:bidi="he-IL" />
  </w:rPr>
</w:r>
```

The resulting run specifies that any complex script contents must be spell and grammar checked as if they were Hebrew, and any Latin character contents must be spell and grammar checked as if they were French (Canada). *end example*]

Attributes	Description

bidi (Complex Script Language)	Specifies the language which shall be used when processing the contents of this run which use complex script characters, as determined by the Unicode character values of the run content.
	If this attribute is omitted, then the languages for the contents of this run using complex script characters shall be automatically determined based on their contents using any appropriate method.
	[*Example*: Consider a run which contains complex script characters in its contents. If those contents should be interpreted as Hebrew, that requirement would be specified as follows in the resulting WordprocessingML:
	`<w:r>`
	`  <w:rPr>`
	`    <w:lang w:bidi="he-IL" />`
	`  </w:rPr>`
	`</w:r>`
	The resulting run specifies that any complex script contents must be spell and grammar checked using a Hebrew dictionary and grammar engine, if one is available. *end example*]
	The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).

eastAsia (East Asian Language)	Specifies the language which shall be used when processing the contents of this run which use East Asian characters, as determined by the Unicode character values of the run content. If this attribute is omitted, then the languages for the contents of this run using East

Attributes	Description

	Asian characters shall be automatically determined based on their contents using any appropriate method.
	
	[*Example*: Consider a run which contains East Asian characters in its contents. If those contents should be interpreted as Korean, that requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`    <w:lang w:eastAsia="ko-KR" />`
	`  </w:rPr>`
	`</w:r>`
	
	The resulting run specifies that any complex script contents must be spell and grammar checked using a Korean dictionary and grammar engine, if one is available. *end example*]
	
	The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).

val (Latin Language)	Specifies the language which shall be used to check spelling and grammar (if requested) when processing the contents of this run which use Latin characters, as determined by the Unicode character values of the run content.
	
	If this attribute is omitted, then the languages for the contents of this run using Latin characters shall be automatically determined based on their contents using any appropriate method.
	
	[*Example*: Consider a run which contains Latin characters in its contents. If those contents should be interpreted as English (Canada), that requirement would be specified as follows in the resulting WordprocessingML:
	
	`<w:r>`
	`  <w:rPr>`
	`    <w:lang w:val="en-CA" />`
	`  </w:rPr>`
	`</w:r>`
	
	The resulting run specifies that any complex script contents must be spell and grammar checked using a English (Canada) dictionary and grammar engine, if one is available. *end example*]
	
	The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Language) is located in §A.1. *end note*]

17.3.2.21 noProof (Do Not Check Spelling or Grammar)

This element specifies that the contents of this run shall not report any errors when the document is scanned for spelling and grammar. [*Note*: It is entirely at the consumer's/producer's discretion whether this is done by not checking the region for spelling and grammar, or simply by suppressing the results. *end note*]

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then spelling and grammar error shall not be suppressed on the contents of this run.

[*Example*: Consider a run of text which must not ever have spelling or grammar errors reported for the contents of the run, for example, the XML fragments included in ISO/IEC 29500. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:noProof w:val="true"/>
</w:rPr>
```

This run explicitly declares that the noProof property is true, so the contents of this run never report spelling or grammar errors. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.22 oMath (Office Open XML Math)

This element specifies that this run contains WordprocessingML which shall be handled as though it was Office Open XML Math.

[*Rationale*: Like other run properties can be applied to the glyph representing the paragraph mark, it is possible to create an Office Open XML Math equation on an empty paragraph as well. Since that paragraph mark must be defined by WordprocessingML, it is not possible to store the paragraph using the Office Open XML Math markup. Instead, this run property is stored on the paragraph mark's run properties to indicate that the paragraph mark is part of an Office Open XML Math equation. For example, the first paragraph below is stored as Office Open XML Math:

The paragraph must be a p (§17.3.1.22) element, but that would mean the data loss of the Math markup when saving as a WordprocessingML package. In order to prevent that data loss, this property stores the Math property as a run property. *end rationale*]

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then this run shall not be treated as Office Open XML Math.

This property can be applied to any run, but that should only introduce the semantic that the run is math in the user interface, and shall not change the appearance of the text.

[*Example*: Consider a paragraph in WordprocessingML where the paragraph mark glyph (the pilcrow mark - ¶) has been formatted as Math. Since this mark is not an actual run, it cannot be written out in the Office Open XML Math syntax, and must be written out as a property on the actual run as follows:

```
<w:pPr>
  <w:rPr>
    <w:oMath />
  </w:rPr>
</w:pPr>
```

This property is therefore used to roundtrip the math setting on this paragraph mark character. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.23 outline (Display Character Outline)

This element specifies that the contents of this run should be displayed as if they have an outline, by drawing a one pixel wide border around the inside and outside borders of each character glyph in the run.

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then outline shall not be applied to the contents of this run.

This element shall not be present with either the emboss (§17.3.2.13) or imprint (§17.3.2.18) properties on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which must have the outline property explicitly turned off for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:outline w:val="false"/>
</w:rPr>
```

This run explicitly declares that the outline property is `false`, so the contents of this run do not appear as if they have an exterior outline around them. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.24 position (Vertically Raised or Lowered Text)

This element specifies the amount by which text shall be raised or lowered for this run in relation to the default baseline of the surrounding non-positioned text. This allows the text to be repositioned without altering the font size of the contents.

If the val attribute is positive, then the parent run shall be raised above the baseline of the surrounding text by the specified number of half-points. If the val attribute is negative, then the parent run shall be lowered below the baseline of the surrounding text by the specified number of half-points.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the text shall not be raised or lowered relative to the default baseline location for the contents of this run.

[*Example*: Consider a run which must be positioned 12 points above the default baseline location when displaying its contents. This requirement would be specified using the following WordprocessingML:

```
<w:rPr>
  <w:position w:val="24" />
</w:rPr>
```

The resulting run is positioned 24 half-points above the default baseline location because the contents of the val attribute are positive. *end example*]

Attributes	Description

val (Signed Half-Point Measurement)	Specifies a positive or negative measurement in half-points (1/144 of an inch).
	The contents of this attribute value are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:rPr>`
	`  <w:position w:val="-12" />`
	`</w:rPr>`
	In this case, the value in the val attribute is amount by which the specified run must be raised or lowered compared to the baseline of the surrounding text.
	In all cases, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_SignedHpsMeasure simple type (§17.18.80).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SignedHpsMeasure) is located in §A.1. *end note*]

©ISO/IEC 2012 – All rights reserved

17.3.2.25 r (Text Run)

This element specifies a run of content in the parent field, hyperlink, custom XML element, structured document tag, smart tag, or paragraph.

The contents of a run in a WordprocessingML document shall consist of any combination of run content.

[*Example*: Consider a basic WordprocessingML paragraph with a pair of runs. This run would be expressed as follows:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Text</w:t>
      </w:r>
      <w:fldSimple w:instr="AUTHOR">
        <w:r>
          <w:t>Author Name</w:t>
        </w:r>
      </w:fldSimple>
    </w:p>
  </w:body>
</w:document>
```

The r element is the container for all of the content in the run, which in this example includes both a run in the paragraph and a run within a simple field. *end example*]

Attributes	Description
rsidDel (Revision Identifier for Run Deletion)	Specifies a unique identifier used to track the editing session when the run was deleted from the main document. All rsid* attributes throughout this document with the same value, if present, shall indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidR (Revision Identifier for Run)	Specifies a unique identifier used to track the editing session when the run was added to the main document. All rsid* attributes throughout this document with the same value, if present, shall

Attributes	Description
	indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidRPr (Revision Identifier for Run Properties)	Specifies a unique identifier used to track the editing session when the run properties were last modified in the main document. All rsid* attributes throughout this document with the same value, if present, shall indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_R) is located in §A.1. *end note*]

17.3.2.26 rFonts (Run Fonts)

This element specifies the fonts which shall be used to display the text contents of this run. Within a single run, there can be up to four types of content present which shall each be allowed to use a unique font:

- ASCII (i.e., the first 128 Unicode code points)
- High ANSI
- Complex Script
- East Asian

The use of each of these fonts shall be determined by the Unicode character values of the run content, unless manually overridden via use of the cs element (§17.3.2.7).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the text shall be displayed in any default font which supports that set of characters.

[*Example*: Consider a single text run with both Arabic and English text, as follows:

Englishالعربية

This content can be expressed in a single WordprocessingML run:

```
<w:r>
  <w:t>English العربية</w:t>
</w:r>
```

Although it is in the same run, the contents are in different font faces by specifying a different font for ASCII and CS characters in the run:

```
<w:r>
  <w:rPr>
    <w:rFonts w:ascii="Courier New" w:cs="Times New Roman" />
  </w:rPr>
  <w:t>English العربية</w:t>
</w:r>
```

This text run must therefore use the `Courier New` font for all characters in the range U+0000 to U+007F, and must use the `Times New Roman` font for all characters in the Complex Script range. *end example*]

Attributes	Description

ascii (ASCII Font)	Specifies a font which shall be used to format all characters in the Unicode range (U+0000–U+007F) within the parent run.
	If the asciiTheme attribute is also specified, then this attribute shall be ignored and that value shall be used instead.
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in any default font which supports these characters.
	[*Example*: Consider a run of text consisting of characters in this range, which must be displayed using the `Courier New` font. This requirement would be specified as follows in the resulting WordprocessingML:
	`<w:rPr>`
	`  <w:rFonts w:ascii="Courier New" />`
	`</w:rPr>`
	The ascii attribute specifies that the run must use the `Courier New` font for all text in thisrange. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

asciiTheme (ASCII Theme Font)	Specifies a theme font which shall be used to format all characters in the Unicode range (U+0000–U+007F) within the parent run. This theme font is a reference to one of the

Attributes	Description

	predefined theme fonts, located in the document's Theme part,which allows for font information to be set centrally in the document.
	If the ascii attribute is also specified, then that attribute shall be ignored and this value shall be used instead.
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in the font specified by the ascii attribute.
	[*Example*: Consider a run of ASCII text which must be displayed using the `majorASCII` theme font. This requirement would be specified as follows in the resulting WordprocessingML:
	`<w:rPr>`
	`  <w:rFonts w:asciiTheme="majorAscii" />`
	`</w:rPr>`
	The ascii attribute specifies that the run must use the `majorAscii` theme font as defined in the document's themes part for all text in this range. *end example*]
	The possible values for this attribute are defined by the ST_Theme simple type (§17.18.96).

cs (Complex Script Font)	Specifies a font which shall be used to format all characters in a complex script Unicode range within the parent run.
	If the cstheme attribute is also specified, then this attribute shall be ignored and that value shall be used instead.
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in any default font which supports complex script content.
	[*Example*: Consider a run of Arabic text which must be displayed using the `Arial Unicode MS` font. This requirement would be specified as follows in the resulting WordprocessingML:
	`<w:rPr>`
	`  <w:rFonts w:cs="Arial Unicode MS" />`
	`</w:rPr>`
	The cs attribute specifies that the run must use the `Arial Unicode MS` font for all text in a complex script range. *end example*]
	The possible values for this attribute are defined by the ST_String simple type

Attributes	Description
	(§22.9.2.13).

cstheme (Complex Script Theme Font)	Specifies a theme font which shall be used to format all characters in a complex script Unicode range within the parent run. This theme font is a reference to one of the predefined theme fonts, located in the document's Theme part,which allows for font information to be set centrally in the document.
	If the cs attribute is also specified, then that attribute shall be ignored and this value shall be used instead.
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in the font specified by the cs attribute.
	[*Example*: Consider a run of Arabic text that must be displayed using the `majorBidi` theme font. This requirement would be specified as follows in the resulting WordprocessingML:
	`<w:rPr>`
	`  <w:rFonts w:cstheme="majorBidi" />`
	`</w:rPr>`
	The cstheme attribute specifies that the run must use the `majorBidi` theme font as defined in the document's themes part for all text in a complex script range. *end example*]
	The possible values for this attribute are defined by the ST_Theme simple type (§17.18.96).

eastAsia (East Asian Font)	Specifies a font which shall be used to format all characters in an East Asian Unicode range within the parent run.
	If the eastAsiaTheme attribute is also specified, then this attribute shall be ignored and that value shall be used instead.
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in any default font which supports East Asian content.
	[*Example*: Consider a run of Japanese text which must be displayed using the `MS Mincho` font. This requirement would be specified as follows in the resulting WordprocessingML:
	`<w:rPr>`
	`  <w:rFonts w:eastAsia="MS Mincho" />`
	`</w:rPr>`
	The eastAsia attribute specifies that the run must use the `MS Mincho` font for all text in

Attributes	Description
	an East Asian range. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

eastAsiaTheme (East Asian Theme Font)	Specifies a theme font which shall be used to format all characters in an East Asian Unicode range within the parent run. This theme font is a reference to one of the predefined theme fonts, located in the document's Theme part,which allows for font information to be set centrally in the document.
	If the eastAsia attribute is also specified, then that attribute shall be ignored and this value shall be used instead.
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in the font specified by the eastAsia attribute.
	[*Example*: Consider a run of Japanese text which must be displayed using the `minorEastAsia` theme font. This requirement would be specified as follows in the resulting WordprocessingML:
	`<w:rPr>`
	`  <w:rFonts w:eastAsiaTheme="minorEastAsia" />`
	`</w:rPr>`
	The eastAsiaTheme attribute specifies that the run must use the `minorEastAsia` theme font as defined in the document's themes part for all text in an East Asian range. *end example*]
	The possible values for this attribute are defined by the ST_Theme simple type (§17.18.96).

hAnsi (High ANSI Font)	Specifies a font which shall be used to format all characters in a Unicode range within the parent run which does not fall into one of the three categories defined above, which is called the *high ANSI* range in WordprocessingML. If the hAnsiTheme attribute is also specified, then this attribute shall be ignored and that value shall be used instead. If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in any default font which supports high ANSI content. [*Example*: Consider a run of text which falls into a high ANSI range, and must be displayed using the `Bauhaus 93` font. This requirement would be specified as follows in the resulting WordprocessingML:

Attributes	Description

	`<w:rPr>`
	`  <w:rFonts w:hAnsi="Bauhaus 93" />`
	`</w:rPr>`
	The hAnsi attribute specifies that the run must use the `Bauhaus 93` font for all text in a high ANSI range. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

hAnsiTheme (High ANSI Theme Font)	Specifies a theme font which shall be used to format all characters in a Unicode range within the parent run which does not fall into one of the three categories defined above, which is called the *high ANSI* range in WordprocessingML. This theme font is a reference to one of the predefined theme fonts, located in the document's Theme part,which allows for font information to be set centrally in the document.
	If the hAnsi attribute is also specified, then that attribute shall be ignored and this value shall be used instead.
	If this attribute is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this attribute is never applied in the style hierarchy, then the text shall be displayed in the font specified by the hAnsi attribute.
	[*Example*: Consider a run of text which falls into a high ANSI range, and must be displayed using the `minorHAnsi` theme font. This requirement would be specified as follows in the resulting WordprocessingML:
	`<w:rPr>`
	`  <w:rFonts w:hAnsiTheme="minorHAnsi" />`
	`</w:rPr>`
	The hAnsiTheme attribute specifies that the run must use the `minorHAnsi` theme font as defined in the document's themes part for all text in a high ANSI range. *end example*]
	The possible values for this attribute are defined by the ST_Theme simple type (§17.18.96).

hint (Font Content Type)	Specifies the font type which shall be used to format any ambiguous characters in the current run. There are certain characters which are not explicitly stored in the document, and can be mapped into multiple categories of the four mentioned above. This attribute shall be used to arbitrate that conflict, and determine how ambiguities in this run shall be handled. [*Note*: This is primarily used to handle the formatting on the paragraph mark glyph, and other characters that are not stored as text in the WordprocessingML document. *end note*]

Attributes	Description

	If this attribute is omitted, then this ambiguity can be resolved by any means available.
	[*Example*: Consider the run representing the paragraph mark glyph, which is not stored as a physical character. Since this could therefore be formatted with any of the fonts specified for the run, this ambiguity is resolved using the following WordprocessingML:
	`<w:pPr>`
	`  <w:rPr>`
	`    <w:rFonts w:hint="eastAsia" />`
	`  </w:rPr>`
	`</w:pPr>`
	The hint attribute specifies that the run must use the `eastAsia` font (theme or not, whichever is in use for East Asian text) as defined for this range. *end example*]
	The possible values for this attribute are defined by the ST_Hint simple type (§17.18.41).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Fonts) is located in §A.1. *end note*]

17.3.2.27 rPr (Previous Run Properties)

This element specifies a set of run properties which shall be attributed to a revision by a particular author and at a particular time. This element contains the set of properties which have been tracked as a specific set of revisions by one author.

[*Example*: Consider a run which has a set of run formatting properties that were added with revision tracking turned on. This set of revised properties is specified in the run properties as follows:

```
<w:r>
  <w:rPr>
    <w:b />
    <w:imprint />
    <w:lang w:val="en-CA" />
    <w:rPrChange … w:author="user1">
      <w:rPr>
        <w:i />
        <w:dstrike w:val="false" />
      </w:rPr>
    </w:rPrChange>
  </w:rPr>
</w:r>
```

The rPr element under rPrChange specifies the properties which were applied to the current run before revision tracking was turned on - in this case, italics using the i element (§17.3.2.16), and that any double strikethrough which was applied based on the style hierarchy must be turned off using the dstrike element (§17.3.2.9). *end example*]

The W3C XML Schema definition of this element’s content model (CT_RPrOriginal) is located in §A.1. Each child element from the above table shall not occur more than once. [*Note*: This restriction is not reflected in the element's content model due to limitations of W3C XML Schema language. *end note*]

17.3.2.28 rPr (Run Properties)

This element specifies a set of run properties which shall be applied to the contents of the parent run after all style formatting has been applied to the text. These properties are defined as *direct formatting*, since they are directly applied to the run and supersede any formatting from styles.

This formatting is applied at the following location in the *style hierarchy*:

- Document defaults
- Table styles
- Numbering styles
- Paragraph styles
- Character styles
- Direct formatting (this element)

[*Example*: Consider a run which should have a set of run formatting properties. This set of properties is specified in the run properties as follows:

```
<w:r>
  <w:rPr>
    <w:b />
    <w:imprint />
    <w:lang w:val="en-CA" />
  </w:rPr>
</w:r>
```

The rPr element specifies the properties which are applied to the current run - in this case, bold formatting on the run contents using the b element (§17.3.2.1), an imprinted (engraved) text effect using the imprint element (§17.3.2.18), and that this text should be interpreted as English (Canada) when spell or grammar checking the run text using the lang element (§17.3.2.20). *end example*]

The W3C XML Schema definition of this element’s content model (CT_RPr) is located in §A.1. Each child element from the above table shall not occur more than once. [*Note*: This restriction is not reflected in the element's content model due to limitations of W3C XML Schema language. *end note*]

17.3.2.29 rStyle (Referenced Character Style)

This element specifies the style ID of the character style which shall be used to format the contents of this paragraph.

This formatting is applied at the following location in the *style hierarchy*:

- Document defaults
- Table styles
- Numbering styles
- Paragraph styles
- Character styles (this element)
- Direct Formatting

This means that all properties specified in the style element (§17.7.4.17) with a styleId which corresponds to the value in this element's val attribute are applied to the run at the appropriate level in the hierarchy.

If this element is omitted, or it references a style which does not exist, then no character style shall be applied to the current paragraph. As well, this property is ignored if the run properties are part of a character style.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:rPr>
  <w:pStyle w:val="TestCharacterStyle" />
  <w:b />
  <w:i />
</w:rPr>
```

This run specifies that it inherits all of the run properties specified by the paragraph style with a styleId of `TestCharacterStyle`, which then have any bold or italics settings overridden and set to be applied to the run. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`

Attributes	Description
	`</w:sdtPr>` In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.3.2.30 rtl (Right To Left Text)

This element specifies whether the contents of this run shall have right-to-left characteristics. Specifically, the following behaviors are applied when this element's val attribute is `true` (or an equivalent):

- Formatting – When the contents of this run are displayed, all characters shall be treated as complex script characters. This means that the values of the bCs element (§17.3.2.2) and the iCs element (§17.3.2.17) shall be use to determine bold and italic formatting, that the cs/cstheme attributes on the rFonts element (§17.3.2.26) shall be used to determine the font face, and the szCs element (§17.3.2.39) shall be used to determine the font size.
- Character Directionality Override – When the contents of this run are displayed, this property acts as a right-to-left override for characters which are classified as follows (using the Unicode Character Database):
  - Weak types except European Number, European Number Terminator, Common Number Separator, Arabic Number and (for Hebrew text) European Number Separator when constituting part of a number
  - Neutral types
- [*Rationale*: This override allows applications to store and utilize higher-level information beyond that implicitly derived from the Unicode Bidirectional algorithm. For example, if the string "first second" appears in a right-to-left paragraph inside a document, the Unicode algorithm would always result in "first second" at display time (since the neutral character is surrounded by strongly classified characters). However, if the whitespace was entered using a right-to-left input method (e.g. a Hebrew keyboard), then that character could be classified as RTL using this property, allowing the display of "second first" in a right-to-left paragraph, since the user explicitly asked for the space in a right-to-left context. *end rationale*]

  This element provides information used to resolve the (Unicode) classifications of individual characters as either L, R, AN or EN. Once this is determined, the line should be displayed subject to the recommendation of the Unicode BiDi algorithm in reordering resolved levels.

This property shall not be used with strongly left-to-right text. Any behavior under that condition is unspecified. . This property, when off, shall not be used with strong right-to-left text. Any behavior under that condition is unspecified.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then right to left characteristics shall not be applied to the contents of this run.

[*Example*: Consider the following WordprocessingML visual content: “`first second,` أول ثاني”. This content might appear as follows within its parent paragraph:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">first second, </w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rtl/>
    </w:rPr>
    <w:t>أول ثاني
</w:t>
  </w:r>
</w:p>
```

The presence of the rtl element on the second run specifies that:

- The formatting on that run is specified using the complex-script property variants.
- The whitespace character is treated as right-to-left.

end example]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.3.2.31 shadow (Shadow)

This element specifies that the contents of this run shall be displayed as if each character has a shadow. For left-to-right text, the shadow is beneath the text and to its right; for right-to-left text, the shadow is beneath the text and to its left.

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then shadowing shall not be applied to the contents of this run.

This element shall not be present with either the emboss (§17.3.2.13) or imprint (§17.3.2.18) properties on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which must have the shadow property explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:shadow w:val="true"/>
</w:rPr>
```

This run explicitly declares that the shadow property is `true`, so the contents of this run appear with a shadow. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.32 shd (Run Shading)

Like paragraph shading, this element specifies the shading applied to the contents of the run.

This shading consists of three components:

- Background Color
- (optional) Pattern
- (optional) Pattern Color

The resulting shading is applied by setting the background color behind the paragraph, then applying the pattern color using the mask supplied by the pattern over that background.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then run shading shall not be applied to the contents of this run.

[*Example*: Consider a run which must have a background consisting of a theme color `accent6` with a theme color `text2` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:

```
<w:pPr>
  <w:shd w:val="pct20" w:themeColor="text2" w:themeFill="accent6" />
</w:pPr>
```

The resulting run uses the background color `accent6` under the foreground pattern color `text2` as specified by the `pct20` pattern mask. *end example*]

This element's content model is defined by the common shading properties definition in §17.3.5.

17.3.2.33 smallCaps (Small Caps)

This element specifies that all small letter characters in this text run shall be formatted for display only as their capital letter character equivalents in a font size two points smaller than the actual font size specified for this text. This property does not affect any non-alphabetic character in this run, and does not change the Unicode character for lowercase text, only the method in which it is displayed. If this font cannot be made two point smaller than the current size, then it shall be displayed as the smallest possible font size in capital letters.

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the characters are not formatted as capital letters.

This element shall not be present with the caps (§17.3.2.5) property on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider the words `Hello, world`, which must be displayed in small capital letters in a document. This constraint is specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:smallCaps w:val="true" />
    <w:sz w:val="24" />
  </w:rPr>
  <w:t>Hello, world</w:t>
</w:r>
```

This run displays using a 12 point capital letter for the capital letter H and W, and a 10 point capital letter for the lowercase letters in the run, even though the lowercase characters are used in actual run contents. If this property is removed, the original character forms is displayed (they are not lost). *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.34 snapToGrid (Use Document Grid Settings For Inter-Character Spacing)

This element specifies whether the current run should use the document grid characters per line settings defined in the docGrid element (§17.6.5) when laying out the contents in this run. This setting determines whether the additional character pitch specified in the document grid shall be added to each character in this run as specified by the document grid.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the run shall use the document grid setting to lay out text when a document grid is defined for the parent section.

[*Example*: Consider two runs in a section with a document grid set to allow `20` characters per line. This document grid would effectively specifies that an additional character pitch must be added to each line in order to ensure that the resulting line contains only 20 East Asian characters.

If this property is set on the first run, but turned off on the second run, as follows:

```
<w:r>
  <w:t>Run One</w:t>
</w:r>
<w:r>
  <w:rPr>
    <w:snapToGrid w:val="off" />
  </w:rPr>
  <w:t>Run Two</w:t>
</w:r>
```

The resulting document must have the required additional character pitch added to each character in run one, but zero additional character pitch added to each character in run two, since the snapToGrid property is turned off. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.35 spacing (Character Spacing Adjustment)

This element specifies the amount of character pitch which shall be added or removed after each character in this run before the following character is rendered in the document. This property has an effect equivalent to the additional character pitched added by a document grid applied to the contents of a run.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the run shall not have any additional character pitch applied to any character in its contents.

[*Example*: Consider a run of text which must have ten points of additional character spacing explicitly added to each character within the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:spacing w:val="200"/>
</w:rPr>
```

This run explicitly declares that the spacing value is 200, so the contents of this run appear as if they have 10 additional points of spacing added between them. *end example*]

Attributes	Description
val (Positive or Negative Value in Twentieths of a Point)	Specifies a positive or negative measurement in twentieths of a point (equivalent to 1/1440th of an inch). The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of -720 whose type is ST_SignedTwipsMeasure. This attribute value specifies a value of negative one-half of an inch or -36 points (-

Attributes	Description
	720 twentieths of a point = -36 points = -0.5 inches). *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SignedTwipsMeasure) is located in §A.1. *end note*]

17.3.2.36 specVanish (Paragraph Mark Is Always Hidden)

This element specifies that the given run shall always behave as if it is hidden, even when hidden text is being displayed in the current document.

This property shall only be used to specify that a paragraph mark shall never be used to break the end of a paragraph for display, even if it is being shown on the document, as would be the case if a regularly hidden paragraph was not being displayed in the document. [*Note*: This property was typically used to ensure that a paragraph style can be applied to a part of a paragraph, and still appear as in the Table of Contents (which in previous word processors would ignore the use of the style if it were being used as a character style. *end note*] If this element is applied to any other run, it can be ignored.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the run properties for the paragraph mark shall not always be treated as if hidden.

[*Example*: Consider a paragraph mark which never be used to break the end of the paragraph in the document. This constraint is specified using the following WordprocessingML:

```
<w:pPr>
  <w:rPr>
    <w:specVanish />
  </w:rPr>
</w:pPr>
```

The presence of the specVanish element means that this paragraph mark must always be treated as hidden (must never be used to end the paragraph for display), but can be used to mark the end of use of a paragraph style. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.37 strike (Single Strikethrough)

This element specifies that the contents of this run shall be displayed with a single horizontal line through the center of the line.

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then strikethrough shall not be applied to the contents of this run.

This element shall not be present with the dstrike (§17.3.2.9) property on the same run, since they are mutually exclusive in terms of appearance.

[*Example*: Consider a run of text which must have the strike property explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:strike w:val="true"/>
</w:rPr>
```

This run explicitly declares that the strike property is `true`, so the contents of this run has a single horizontal strikethrough line. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.38 sz (Non-Complex Script Font Size)

This element specifies the font size which shall be applied to all non complex script characters in the contents of this run when displayed. The font sizes specified by this element's val attribute are expressed as half-point values.

If this element is not present, the default is to leave the font size at the value applied at the previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then any appropriate font size can be used for non complex script characters.

[*Example*: Consider a run of text which must have an explicit font size of 13.5 points for the non complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:sz w:val="27"/>
</w:rPr>
```

This run explicitly declares that the sz property is 27 half-point for the non-complex script contents of this run, so the text is displayed in 13.5 point font size. *end example*]

Attributes	Description
val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch). The contents of this attribute value are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment:

Attributes	Description
	`<… w:val="30" />` The value in the val attribute is 30, which is equivalent to 15 points (30 half-points). This value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_HpsMeasure simple type (§17.18.42).

[*Note*: The W3C XML Schema definition of this element's content model (CT_HpsMeasure) is located in §A.1. *end note*]

17.3.2.39 szCs (Complex Script Font Size)

This element specifies the font size which shall be applied to all complex script characters in the contents of this run when displayed. The font sizes specified by this element's val attribute are expressed as half-point values.

If this element is not present, the default is to leave the font size at the value applied at the previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then any appropriate font size can be used for complex script characters.

[*Example*: Consider a run of text which must have an explicit font size of 10 points for the complex script contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:szCs w:val="20"/>
</w:rPr>
```

This run explicitly declares that the sz property is 20 half-point for the non-complex script contents of this run, so the text is displayed in 10 point font size. *end example*]

Attributes	Description
val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch). The contents of this attribute value are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment: `<… w:val="30" />` The value in the val attribute is 30, which is equivalent to 15 points (30 half-points). This value is interpreted in the context of the parent element. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_HpsMeasure simple type (§17.18.42).

[*Note*: The W3C XML Schema definition of this element's content model (CT_HpsMeasure) is located in §A.1. *end note*]

17.3.2.40 u (Underline)

This element specifies that the contents of this run should be displayed along with an underline appearing directly below the character height (less all spacing above and below the characters on the line).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then an underline shall not be applied to the contents of this run.

[*Example*: Consider a run of text which must have a double underline explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:u w:val="double"/>
</w:rPr>
```

This run explicitly declares an underline using the u property. The val of that underline is double, so the style of the underline on this run must be a double line. *end example*]

Attributes	Description

color (Underline Color)	Specifies the color for the underlining on this run.
	This color can either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the underline color as appropriate.
	If the underline specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value.
	[*Example*: Consider a run color with value auto, as follows:
	`<w:rPr>`
	`  <w:u … w:color="auto" />`
	`</w:rPr>`
	This color therefore can be automatically be modified by a consumer as appropriate, for example, in order to ensure that the underline can be distinguished against the page's background color. *end example*]
	The possible values for this attribute are defined by the ST_HexColor simple type

Attributes	Description
	(§17.18.38).

themeColor (Underline Theme Color)	Specifies a theme color which should be applied to the current underline.
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part, which allows for color information to be set centrally in the document.
	If the themeColor attribute is specified, then the color attribute is ignored for this underline.
	[*Example*: Consider an underlined run of text whose underline should be displayed using the `accent3` theme color from the document's Theme part. This requirement would be specified as follows in the resulting WordprocessingML:
	`<w:rPr>`
	`  <w:u … w:themeColor="accent3" />`
	`</w:rPr>`
	The themeColor attribute specifies that the underline must use the `accent3` theme color. *end example*]
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§17.18.97).

themeShade (Underline Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this underline. If the themeShade is supplied, then it is applied to the RGB value of the theme color to determine the final color applied to this underline. The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a underline in a document. This shade is calculated as follows: $\begin{aligned} S_{xml} &= 0.4 * 255 \\ &= 102 \\ &= 66(hex) \end{aligned}$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows:

Attributes	Description
	$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$ - Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48,0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L' = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48,0.39698\right)$and converting back to RGB, we get `943634`. This transformed value can be seen in the resulting underline's color attribute: `<w:u w:color="943634" w:themeColor="accent2" w:themeShade="BF" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).
themeTint (Underline Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this underline's contents. If the themeTint is supplied, then it is applied to the RGB value of the theme color to determine the final color applied to this run. The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to an underline in a document. This tint is calculated as follows: $T_{xml} = 0.6 * 255$ $= 153$ $= 99(hex)$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied

Attributes	Description
	as follows: - Convert the color to the HSL color format (values from 0 to 1) - Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \ \mathrm{Tint}_{\mathrm{pct}}\right)$$ - Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45{,}0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get: $$\begin{aligned} L' &= 0.53 * 0.6 + (1 - .6) \\ &= 0.71 \end{aligned}$$ Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45{,}0.71\right)$and converting back to RGB, we get `95B3D7`. This transformed value can be seen in the resulting underline formatting's WordprocessingML color attribute: `<w:u … w:color="95B3D7" w:themeColor="accent2" w:themeTint="99" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

val (Underline Style)	Specifies the pattern which shall be used to create the underline applied beneath the text in this run.
	
	Each of these possible patterns are shown in the simple type referenced below.
	
	[*Example*: Consider a run of text which must have a double underline explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:
	
	`<w:rPr>`
	`  <w:u w:val="double"/>`
	`</w:rPr>`
	
	The val of the underline on this run is `double`, so the style of the underline on this run

Attributes	Description
	must be a double line. *end example*] The possible values for this attribute are defined by the ST_Underline simple type (§17.18.99).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Underline) is located in §A.1. *end note*]

17.3.2.41 vanish (Hidden Text)

This element specifies whether the contents of this run shall be hidden from display at display time in a document. [*Note*: The setting should affect the normal display of text, but an application can have settings to force hidden text to be displayed. *end note*]

This formatting property is a *toggle property* (§17.7.3).

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy* .If this element is never applied in the style hierarchy, then this text shall not be hidden when displayed in a document.

[*Example*: Consider a run of text which must have the hidden text property turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:vanish />
</w:rPr>
```

This run declares that the vanish property is set for the contents of this run, so the contents of this run is hidden when the document contents are displayed. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.2.42 vertAlign (Subscript/Superscript Text)

This element specifies the alignment which shall be applied to the contents of this run in relation to the default appearance of the run's text. This allows the text to be repositioned as subscript or superscript without altering the font size of the run properties.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the text shall not be subscript or superscript relative to the default baseline location for the contents of this run.

[*Example*: Consider a run which must be positioning as superscript when displaying its contents. This requirement would be specified using the following WordprocessingML:

```
<w:rPr>
  <w:vertAlign w:val="superscript" />
</w:rPr>
```

The resulting run is positioned as `superscript`, therefore it is rendered in a smaller size above the default baseline location for the contents of the run. *end example*]

Attributes	Description

val (Subscript/Superscript Value)	Specifies the type of vertical alignment applied to the contents of the current run.
	[*Example*: Consider a run which must be positioning as superscript when displaying its contents. This requirement would be specified using the following WordprocessingML:
	`<w:rPr>`
	`  <w:vertAlign w:val="superscript" />`
	`</w:rPr>`
	The value of the `val` attribute is `superscript`, therefore the run's contents are rendered in a smaller size above the default baseline location for the contents of the run. *end example*]
	The possible values for this attribute are defined by the ST_VerticalAlignRun simple type (§22.9.2.17).

[*Note*: The W3C XML Schema definition of this element's content model (CT_VerticalAlignRun) is located in §A.1. *end note*]

17.3.2.43 w (Expanded/Compressed Text)

This element specifies the amount by which each character shall be expanded or when the character is rendered in the document. This property has an of stretching or compressing each character in the run, as opposed to the spacing element (§17.3.2.35) which expands/compresses the text by adding additional character pitch but not changing the width of the actual characters displayed on the line.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy*. If this element is never applied in the style hierarchy, then the run shall be displayed at 100% of its normal width.

[*Example*: Consider a run of text which must be expanded to 200% of its normal width when displaying each character within the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:w w:val="200%"/>
</w:rPr>
```

This run explicitly declares that the w value is 200%, so the contents of this run appear at 200% of their normal character width by stretching the width of each character. *end example*]

Attributes	Description

val (Text Expansion/Compression Value)	Specifies that the percentage by which the contents of this run shall be expanded or compressed with respect to its normal (100%) character width.
	If this attribute is omitted, then the contents of this run shall be displayed at 100% of its normal size.
	[*Example*: Consider a run of text which must be compressed to 200% when displaying each character within the contents of the run. This constraint is specified using the following WordprocessingML:
	`<w:rPr>`
	`  <w:w w:val="50%"/>`
	`</w:rPr>`
	This run explicitly declares that the w value is 50%, so the contents of this run appear at 50% of their normal character width by compressing the width of each character. *end example*]
	The possible values for this attribute are defined by the ST_TextScale simple type (§17.18.95).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextScale) is located in §A.1. *end note*]

17.3.2.44 webHidden (Web Hidden Text)

This element specifies whether the contents of this run shall be hidden from display at display time in a document when the document is being displayed in a web page view. [*Note*: The setting should affect the normal display of text in a web page view, but an application can have settings to force hidden text to be displayed. *end note*] As well, this setting should not affect a normal paginated view of the document.

If this element is not present, the default value is to leave the formatting applied at previous level in the *style hierarchy* .If this element is never applied in the style hierarchy, then this text shall not be hidden when displayed in a document in a web page view.

[*Example*: Consider a run of text which must have the hidden text property turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:webHidden />
</w:rPr>
```

This run declares that the webHidden property is set for the contents of this run, so the contents of this run is hidden when the document contents are displayed in a web page view. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.3 Run Content

The final level of the document hierarchy is *run content*, which is defined as the set of elements which can be contained as the contents of a particular run in a document.

[*Note*: Types of run content in WordprocessingML include:

- Text
- Field Codes
- DrawingML objects
- Fields

end note]

[*Example*: Consider the following run within a WordprocessingML document:

```
<w:r>
  <w:rPr>
    <w:b/>
    <w:i/>
  </w:rPr>
  <w:t>quick</w:t>
</w:r>
```

The run content consists of a single string of run text inside the t element, which reads `quick`. *end example*]

17.3.3.1 br (Break)

This element specifies that a break shall be placed at the current location in the run content. A *break* is a special character which is used to override the normal line breaking that would be performed based on the normal layout of the document's contents. [*Example*: Normal breaking for English would occur only after a breaking space or optional hyphen character. *end example*]

The behavior of this break character (the location where text shall be restarted after this break) shall be determined by its type and clear attribute values, described below.

[*Example*: Consider the following sentence in a WordprocessingML document:

```
This is a simple sentence.
```

Normally, just as shown above, this sentence would be displayed on a single line as it is not long enough to require line breaking (given the width of the current page). However, if a text wrapping break character (a typical line break) were inserted after the word `is`, as follows:

```
<w:r>
  <w:t>This is</w:t>
  <w:br/>
  <w:t xml:space="preserve"> a simple sentence.</w:t>
</w:r>
```

This would imply that this break must be treated as a simple line break, and break the line after that word:

```
This is
 a simple sentence.
```

The break character forced the following text to be restarted on the next available line in the document. *end example*]

Attributes	Description
clear (Restart Location For Text Wrapping Break)	Specifies the location which shall be used as the next available line when the break's type attribute has a value of `textWrapping`. This property only affects the restart location when the current run is being displayed on a line which does not span the full text extents due to the presence of a floating object (see possible values for details). If this break is not of style `textWrapping`, then this attribute shall be ignored. If this attribute is omitted, then its value shall be assumed to be none if needed. [*Example*: Consider a text wrapping break character which should force the restart location to the next line which spans the full width of the text extents of the page (there are no floating objects which interrupt the line). This line break is of style `textWrapping`, since it must only advance to the next line, but the clear value must specify that this restart location must ignore all lines which are not of the full line width by specifying a value of `all`, as follows: `<w:br w:type="textWrapping" w:clear="all" />` This break must therefore not use the next available line, but rather the next available line ignoring all lines which do not span the full text width. *end example*] The possible values for this attribute are defined by the ST_BrClear simple type (§17.18.3).
type (Break Type)	Specifies the break type of the current break. The break type determines the next location where text shall be placed after this manual break is applied to the text contents (see possible values for details). If this attribute is omitted, then it shall be assumed to be of style `textWrapping`. [*Example*: Consider a manual break which must advance the text to the next text column in the document, rather than just the next available line. This break would therefore be specified as follows:

Attributes	Description
	`<w:br w:type="column"/>` The type attribute specifies a value of `column`, which means that the break must force the next character in the document to be restarted on the next line in a new text column in the document. *end example*] The possible values for this attribute are defined by the ST_BrType simple type (§17.18.4).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Br) is located in §A.1. *end note*]

17.3.3.2 contentPart (Content Part)

This element specifies a reference to XML content in a format not defined by ISO/IEC 29500. [*Note*: This part allows the native use of other commonly used interchange formats, such as:

- MathML (http://www.w3.org/TR/MathML2/)
- SMIL (http://www.w3.org/TR/REC-smil/)
- SVG (http://www.w3.org/TR/SVG11/)

end note]

The relationship type of the explicit relationship specified by this element shall be http://purl.oclc.org/ooxml/officeDocument/relationships/customXml and have a TargetMode attribute value of `Internal`. If an application cannot process content of the content type specified by the targeted part, then it should continue to process the file. If possible, it should also provide some indication that unknown content was not imported.

[*Example*: Consider a WordprocessingML document which includes the following MathML markup in a part named math1.xml:

```
<mml:math xmlns:mml="http://www.w3.org/1998/Math/MathML">
  <mml:mi>r</mml:mi>
  <mml:mo>=</mml:mo>
  <mml:msup>
    <mml:mrow>
      <mml:mfenced separators="|">
        <mml:mrow>
          <mml:mi>V</mml:mi>
          <mml:mfrac>
            <mml:mrow>
              <mml:mn>3</mml:mn>
            </mml:mrow>
```

```
              <mml:mrow>
                <mml:mn>4</mml:mn>
                <mml:mi>π</mml:mi>
              </mml:mrow>
            </mml:mfrac>
          </mml:mrow>
        </mml:mfenced>
      </mml:mrow>
      <mml:mrow>
        <mml:mfrac>
          <mml:mrow>
            <mml:mn>1</mml:mn>
          </mml:mrow>
          <mml:mrow>
            <mml:mn>3</mml:mn>
          </mml:mrow>
        </mml:mfrac>
      </mml:mrow>
    </mml:msup>
  </mml:math>
```

The Main Document Part would reference this content as follows:

```
  <w:body>
    …
    <w:p>
      <w:r>
        <w:contentPart r:id="rId8" />
      </w:r>
    </w:p>
    …
  </w:body>
```

The contentPart element specifies that the content targeted by the relationship with an ID of `rId8` is part of the WordprocessingML document. Examining the contents of the corresponding relationship part item, we can see the targets for that relationship:

```
  <Relationships … >
    …
    <Relationship Id="rId8" TargetMode="Internal"
  Type="http://purl.oclc.org/ooxml/officeDocument/relationships/customXml"
  Target="math1.xml" />
    …
  </Relationships>
```

The corresponding relationship part item shows that the file to be imported is located next to the main document and is named `math1.xml`. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rel) is located in §A.1. *end note*]

17.3.3.3 control (Embedded Control)

This element specifies that the parent embedded object is a representation of an embedded control. This element shall be used to associate the appropriate embedded control settings and properties when the document is displayed.

If the embedded control is not present, cannot be loaded due to application settings, or is not supported, then a suitable placeholder image shall be used to provide a representation of the presence of an embedded control at the appropriate location in the document.

[*Example*: Consider a run which consists of an embedded control. That run would be specified using the following WordprocessingML:

```
<w:r>
  <w:object>
    …
    <w:control r:id="rId99" w:shapeid="10" … />
  </w:object>
</w:r>
```

The control element indicates that the parent embedded object is an embedded control, whose settings and properties are stored on this element and the (optional) target of the relationship specified using the id attribute. *end example*]

Attributes	Description
id (Embedded Control Properties Relationship Reference) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID for the relationship which contains the properties for this embedded control. This property bag is contained in a separate part within the Office Open XML package. The relationship explicitly targeted by this attribute shall be of type http://purl.oclc.org/ooxml/officeDocument/relationships/control or the document shall be conisdered non-conformant. If this attribute is omitted, then the embedded control shall be given no property bag when instantiated. [*Example*: Consider the following WordprocessingML markup for an embedded control in a document: `<w:control r:id="rId5" w:name="CheckBox1" w:shapeid="_x0000_s1027" />` The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rId5` must contain the property data for this embedded control. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
name (Unique Name for Embedded Control)	Specifies a unique name for this embedded control. This name shall be unique across all controls in this document. [*Example*: Consider the following WordprocessingML markup for an embedded control in a document: `<w:control r:id="rId5" w:name="CheckBox1" w:shapeid="_x0000_s1027" />` The name attribute specifies that the unique name for this control must be `CheckBox1`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
shapeid (Shape Reference)	Specifies the shape ID for a shape which shall be used to define the presentation and location of this embedded control within the document if the control is floating using the DrawingML syntax. [*Note*: This positioning data is sufficient to display the control in any case where: • The embedded control is not on the current machine • Embedded controls are disabled • Embedded controls of this control type are not supported *end note*] This shape ID reference is resolved by looking for a DrawingML object whose id attribute matches the value specified within this attribute. If no such shape exists, then the control shall be rendered inline in the document content at the current run content location. If this attribute is omitted, then this embedded control shall be displayed inline in the current location in the parent run. [*Example*: Consider the following WordprocessingML markup for an embedded control in a document: `<w:control r:id="rId5" w:name="CheckBox1" w:shapeid="10" />` The shapeid attribute specifies that the DrawingML object with an id attribute value of `10` must contain the positioning data for this embedded control. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Control) is located in §A.1. *end note*]

17.3.3.4 cr (Carriage Return)

This element specifies that a carriage return shall be placed at the current location in the run content. A *carriage return* is the equivalent of Unicode character 000D, and is used to end the current line of text in WordprocessingML.

The behavior of a carriage return in run content shall be identical to a break character with null type and clear attributes, which shall end the current line and find the next available line on which to continue.

[*Example*: Consider the following sentence in a WordprocessingML document:

```
This is another simple sentence.
```

Normally, just as shown above, this sentence would be displayed on a single line as it is not long enough to require line breaking (given the width of the current page). However, if a carriage return were inserted after the word another, as follows:

```
<w:r>
  <w:t>This is another</w:t>
  <w:cr/>
  <w:t xml:space="preserve"> simple sentence.</w:t>
</w:r>
```

This would imply that this carriage return character must force a line break, and break the line after that word:

```
This is another
 simple sentence.
```

The carriage return character forced the following text to be restarted on the next available line in the document. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.3.3.5 dayLong (Date Block - Long Day Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs) . [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run's lang property (§17.3.2.20).

The long day format date block shall use a date picture of DDDD, retrieving the long day format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a long date: </w:t>
  <w:dayLong />
</w:r>
```

This run specifies that a long day format date block must be placed after the text string literal This is a long date: in the document. Assuming that the host application's primary editing language is French (Canada) and today's date is 2006-04-12, this run would be displayed as follows:

```
This is a long date: mercredi
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.3.3.6 dayShort (Date Block - Short Day Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs) . [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run's lang property (§17.3.2.20).

The short day format date block shall use a date picture of DD, retrieving the short day format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a short date: </w:t>
  <w:dayShort />
</w:r>
```

This run specifies that a short day format date block must be placed after the text string literal `This is a short date:` in the document. Assuming that the host application's primary editing language is English (Canada) and today's date is 2006-04-12, this run would be displayed as follows:

```
This is a short date: 12
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.3.3.7 delText (Deleted Text)

This element specifies that this run contains literal text which shall be displayed in the document. The `delText` element shall be used for all text runs which are part of a region of text that is contained in a deleted region using the del element (§17.13.5.14).

[*Example*: Consider a paragraph of WordprocessingML content which reads `This is deleted text`, where the words `deleted text` are part of a deleted region of the document. This paragraph would therefore be represented as follows:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is </w:t>
  </w:r>
  <w:del w:author="Cooper W.">
    <w:r>
      <w:delText>deleted text</w:delText>
    </w:r>
  </w:del>
</w:p>
```

The deleted text is contained in a delText node, while the regular text is contained in a t node. *end example*]

Attributes	Description

xml:space (Content Contains Significant Whitespace)
 	
Namespace: http://www.w3.org/XML/1998/namespace
	Specifies how white space should be handled for the contents of this element using the W3C space preservation rules.
	
	[*Example*: Consider the following run contained within a WordprocessingML document:
	
	`<w:r>`
	`  <w:t>   significant whitespace   </w:t>`
	`</w:r>`
	
	Although there are three spaces on each side of the text content in the run, that whitespace has not been specifically marked as significant, therefore it is subject to the space preservation rules currently specified in that run's scope. *end example*]
	
	The possible values for this attribute are defined by §2.10 of the XML 1.0 specification.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Text) is located in §A.1. *end note*]

17.3.3.8 dirty (Invalidated Field Cache)

This element specifies that the field has been changed and the results shall be updated on open in a conforming consumer.

This element's content model is defined by the common boolean property definition in §17.17.4.

17.3.3.9 drawing (DrawingML Object)

This element specifies that a DrawingML object is located at this position in the run's contents. The layout properties of this DrawingML object are specified using the WordprocessingML Drawing syntax (§20.4).

[*Example*: Consider a run which consists of a picture which is in line with the text in that paragraph (i.e. on the line and affects the line height). That run would be specified using the following WordprocessingML:

```
<w:r>
  <w:drawing>
    <wp:inline>
      …
    </wp:inline>
  </w:drawing>
</w:r>
```

The drawing element indicates that a DrawingML object and its WordprocessingML Drawing positioning data are located at the current position in the run (e.g. a picture or a chart). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Drawing) is located in §A.1. *end note*]

17.3.3.10 hps (Phonetic Guide Text Font Size)

This element specifies the font size which shall be applied to the phonetic guide text in the contents of this run when displayed.

If this element disagrees with the run properties on the phonetic guide text rt element (§17.3.3.24), then those properties shall be ignored and this element shall determine the size of the phonetic guide text.

[*Example*: Consider a run of phonetic guide text which must have an explicit font size of 13.5 points. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:hps w:val="27"/>
  …
</w:rubyPr>
```

The hps property is 27 half-points for the ruby text in this run, so the phonetic guide text is displayed in 13.5 point font size. *end example*]

Attributes	Description
val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch). The contents of this attribute value are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment: `<… w:val="30" />` The value in the val attribute is 30, which is equivalent to 15 points (30 half-points). This value is interpreted in the context of the parent element. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_HpsMeasure simple type (§17.18.42).

[*Note*: The W3C XML Schema definition of this element's content model (CT_HpsMeasure) is located in §A.1. *end note*]

17.3.3.11 hpsBaseText (Phonetic Guide Base Text Font Size)

This element specifies the font size which shall be applied to the base text of this phonetic guide text when displayed. If this element disagrees with the run properties on the phonetic guide base text rubyBase element (§17.3.3.27), then this property shall be ignored and the sz element (§17.3.2.38) in that run shall determine the size of the phonetic guide base text.

[*Example*: Consider a run of phonetic guide base text which shall have an explicit font size of 30 points. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:hpsBaseText w:val="60"/>
  …
</w:rubyPr>
```

The hpsBaseText property is 60 half-points for the base text in this phonetic guide, so the phonetic guide base text is displayed in 30 point font size. *end example*]

Attributes	Description
val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch). The contents of this attribute value are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment: `<… w:val="30" />` The value in the val attribute is 30, which is equivalent to 15 points (30 half-points). This value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_HpsMeasure simple type (§17.18.42).

[*Note*: The W3C XML Schema definition of this element's content model (CT_HpsMeasure) is located in §A.1. *end note*]

17.3.3.12 hpsRaise (Distance Between Phonetic Guide Text and Phonetic Guide Base Text)

This element specifies the distance which shall be left between the phonetic guide base text and the phonetic guide text when this phonetic guide text is displayed.

[*Example*: Consider a run of phonetic guide text which must have 10 points between the phonetic guide base text and the phonetic guide text. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:hpsRaise w:val="20"/>
  …
</w:rubyPr>
```

The hpsRaise property is 20 half-points for the phonetic guide, so the phonetic guide text is displayed 10 points above the phonetic guide base text. *end example*]

Attributes	Description
val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch). The contents of this attribute value are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment: `<… w:val="30" />` The value in the val attribute is 30, which is equivalent to 15 points (30 half-points). This value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_HpsMeasure simple type (§17.18.42).

[*Note*: The W3C XML Schema definition of this element's content model (CT_HpsMeasure) is located in §A.1. *end note*]

17.3.3.13 lastRenderedPageBreak (Position of Last Calculated Page Break)

This element specifies that this position delimited the end of a page when this document was last saved by an application which paginates its content.

[*Guidance*: This element must be used by applications to specify the locations of page breaks within a document when it is saved as WordprocessingML, in order to allow other applications (e.g. assistive software) to utilize this information when reading the document. *end guidance*]

[*Example*: Consider a run which consists of the text `This is the end of the page`, where the word end was the last word on a page. If the application saving this file had paginated this content, that information can be saved with the file as follows:

```
<w:r>
  <w:t>This is the end</w:t>
  <w:lastRenderedPageBreak/>
  <w:t xml:space="preserve"> of the page</w:t>
</w:r>
```

The lastRenderedPageBreak element indicates that there was a page break resulting from pagination of this content, which occurred between the word end and the word `of`. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.3.3.14 lid (Language ID for Phonetic Guide)

This element specifies the language which shall be used for this phonetic guide.

[*Example*: Consider a run of phonetic guide text which is using Japanese as it language. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:lid w:val="ja-JP"/>
  …
</w:rubyPr>
```

The lid property is `ja-JP` for the phonetic guide, so the phonetic guide is specified to be Japanese. *end example*]

Attributes	Description
val (Language Code)	Specifies an identifier for a specific language. This code is interpreted in the context of the parent XML element. [*Example*: Consider an object which must specify the English(Canada) language. That object would use an identifier of `en-CA` to specify this language. *end example*] The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Lang) is located in §A.1. *end note*]

17.3.3.15 monthLong (Date Block - Long Month Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs) . [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run's lang property (§17.3.2.20).

The long month format date block shall use a date picture of MMMM, retrieving the long month format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a long date: </w:t>
  <w:monthLong />
</w:r>
```

This run specifies that a long month format date block shall be placed after the text string literal `This is a long date:` in the document. Assuming that the host application's primary editing language is French (Canada) and today's date is 2006-04-12, this run would be displayed as follows:

```
This is a long date: avril
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.3.3.16 monthShort (Date Block - Short Month Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs). [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run's lang property (§17.3.2.20).

The short month format date block shall use a date picture of MM, retrieving the short month format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a short date: </w:t>
  <w:monthShort />
</w:r>
```

This run specifies that a short month format date block must be placed after the text string literal `This is a short date:` in the document. Assuming that the host application's primary editing language is English (Canada) and today's date is 2006-04-12, this run would be displayed as follows:

```
This is a short date: 04
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.3.3.17 movie (Embedded Video)

This element specifies a location within a document where the specified parent image shall be treated as a static placeholder for an embedded movie. [*Note*: A list of suggested video types in provided in §15.2.17. *end note*]The specified movie file's contents should be displayed when requested at this location in the document. The location of the embedded movie to be displayed when supported shall be specified by the relationship whose Id attribute matches the id attribute on this element.

If the relationship type of the relationship specified by this element is not http://purl.oclc.org/ooxml/officeDocument/relationships/movie, or is not present, then the document shall be considered non-conformant. If an application cannot process external content of the content type specified by the targeted part, then it can be ignored.

[*Example*: Consider a WordprocessingML document which contains a DrawingML shape holding the static image for a movie:

```
<w:object>
  <w:drawing>
    …
  </w:drawing>
  <w:movie r:id="rIdMovie" />
</w:object>
```

The movie element specifies that the part targeted by the relationship with an ID of `rIdMovie` must be imported at the beginning of the document. Examining the contents of the corresponding relationship part item, we can see the targets for that relationship:

```
<Relationships … >
  …
  <Relationship Id="rIdMovie" TargetMode="Internal"
Type="http://purl.oclc.org/ooxml/officeDocument/movie" Target="movie.mov" />
  …
</Relationships>
```

The corresponding relationship part item shows that the movie file is located next to the main document and is named `movie.mov`. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rel) is located in §A.1. *end note*]

17.3.3.18 noBreakHyphen (Non Breaking Hyphen Character)

This element specifies that a non breaking hyphen character shall be placed at the current location in the run content.

The behavior of a non-breaking hyphen in run content shall be to display using the same glyph as the hyphen-minus character (U+002D), however, without that hyphen being a line breaking position (unlike the hyphen-minus character, which does allow line breaking).

[*Example*: Consider the following sentence in a WordprocessingML document: ‘Each citizen has a unique Social Security Number of the form “999-99-9999”, where each 9 represents a decimal digit.’ The fragment of this sentence involving the string literal might be represented in WordprocessingML, as follows:

```
<w:r>
  <w:t>Number of the form “999-99-9999”, where</w:t>
</w:r>
```

However, consider the case in which, on rendering, the right margin was such that the quoted string is broken across multiple lines with the hyphens being used as possible line breaking points; for example:.

```
Each citizen has a unique Social Security Number of the form “999-99-
9999”, where …
```

If such line breaks are undesirable, those hyphens can be marked as non-breaking, as follows:

```
<w:r>
  <w:t>Number of the form “999</w:t>
</w:r>
<w:r>
  <w:noBreakHyphen />
  <w:t>99</w:t>
</w:r>
<w:r>
  <w:noBreakHyphen />
  <w:t>9999”, where</w:t>
</w:r>
```

in which case, for the same margin settings, the rendered result might be like the following:

```
Each citizen has a unique Social Security Number of the form
“999-99-9999”, where …
```

end example]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Empty) is located in §A.1. *end note*]

17.3.3.19 object (Embedded Object)

This element specifies that an embedded object is located at this position in the run’s contents. The layout properties of this embedded object, as well as an optional static representation, are specified using the drawing element (§17.3.3.9).

[*Example*: Consider a run which consists of an embedded object which is in line with the text in that paragraph (i.e. on the line and affects the line height). That run would be specified using the following WordprocessingML:

```
<w:r>
  <w:object>
    <w:drawing>
      …
    </w:drawing>
  </w:object>
</w:r>
```

The object element indicates that an embedded object and its positioning data are located at the current position in the run (e.g. an embedded object). *end example*]

Attributes	Description

dxaOrig (Original Image Width)	Specifies the original (natural) width of the image representation of the current control within the document. Some vector image formats do not store a native size within their format, and this attribute shall only be used in those cases to store this information, so that the image can be appropriately restored as needed.
	If this element is excluded, then the natural size of the image as stored in its format shall be used.
	[*Example*: Consider the following WordprocessingML for an embedded object:
	`<w:object w:dxaOrig="3360" w:dyaOrig="2520">`
	`  …`
	`</w:object>`
	The dxaOrig attribute has a value of `3360`, which specifies that the image used for the embedded object doesn't store its native width, but that width should be 3360 twentieths of a point. *end example*]
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

dyaOrig (Original Image Height)	Specifies the original (natural) height of the image representation of the current control within the document. Some vector image formats do not store a native size within their format, and this attribute shall only be used in those cases to store this information, so that the image can be appropriately restored as needed. If this element is excluded, then the natural size of the image as stored in its format shall be used. [*Example*: Consider the following WordprocessingML for an embedded object: `<w:object w:dxaOrig="3360" w:dyaOrig="2520">`

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	`…`
	`</w:object>`
	
	The dyaOrig attribute has a value of `2520`, which specifies that the image used for the embedded object doesn't store its native height, but that height should be 2520 twentieths of a point. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Object) is located in §A.1. *end note*]

17.3.3.20 objectEmbed (Embedded Object Properties)

This element specifies the visual properties and associated server application of an embedded object.

[*Example*: The following demonstrates a video file embedded in a WordprocessingML document:

```
<w:object … >
  <w:drawing> … </w:drawing>
  <w:objectEmbed drawAspect="content" r:id="rId3" progId="AVIFile"
shapeId="10"/>
</w:object>
```

end example]

Attributes	Description
drawAspect (Object Representation)	Specifies how the object is represented visually in the application. [*Example*: `<w:objectEmbed … drawAspect="content"/>` *end example*] The possible values for this attribute are defined by the ST_ObjectDrawAspect simple type (§17.18.60).
fieldCodes (Field Switches)	This element specifies the WordprocessingML field switches which shall be stored with an embedded object, using the set of field switches defined by the LINK field, as specified in §17.16.5.32. This element shall specify the exact field switches for the field which represents the object. [*Rationale*: Legacy word processors used fields to represent embedded objects - this element stores the field switches not explicitly defined for embeddings so as not to use the fidelity of their contents. *end rationale*]

Attributes	Description
	[*Example*: `<w:objectEmbed … fieldCodes="\f 0"/>` This embedded object specifies additional LINK field code values of \f 0, which specifies that the embedded object must retain its source formatting (as defined in §17.16.5.32). *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
id (Relationship to Embedded Object Data) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID for the relationship which targets the Embedded Object Part containing the embedded object data. The specified relationship shall be of type http://purl.oclc.org/ooxml/officeDocument/relationships/oleObject or the document shall be considered non-conformant. [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId1" />` The markup specifies the associated relationship part with relationship ID `rId1` targets the part containing the corresponding embedded object information. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
progId (Object Application)	Specifies the application associated with the object. [*Example*: `<w:objectEmbed … progId="AVIFile"/>` *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

shapeId (Object Shape)	Specifies the shape with which the object is associated. A shape provides the visual placeholder for an object and this attribute is set to the ID of the placeholder shape.
	[*Example*:
	`<w:objectEmbed … shapeId="10"/>`
	end example]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ObjectEmbed) is located in §A.1. *end note*]

17.3.3.21 objectLink (Linked Object Properties)

This element specifies the visual properties, associated server application and refresh mode of an embedded linked object.

[*Example*: The following demonstrates a video file embedded in a WordprocessingML document:

```
<w:object … >
  <w:drawing> … </w:drawing>
  <w:objectLink drawAspect="icon" r:id="rId3" progId="AVIFile" shapeId="10"
updateMode="user"/>
</w:object>
```

end example]

Attributes	Description
drawAspect (Object Representation)	Specifies how the object is represented visually in the application. [*Example*: `<w:objectEmbed … drawAspect="content"/>` *end example*] The possible values for this attribute are defined by the ST_ObjectDrawAspect simple type (§17.18.60).
fieldCodes (Field Switches)	This element specifies the WordprocessingML field switches which shall be stored with an embedded object, using the set of field switches defined by the LINK field, as specified in §17.16.5.32. This element shall specify the exact field switches for the field which represents the object. [*Rationale*: Legacy word processors used fields to represent embedded objects - this element stores the field switches not explicitly defined for embeddings so as not to use the fidelity of their contents. *end rationale*] [*Example*:

Attributes	Description
	`<w:objectEmbed … fieldCodes="\f 0"/>` This embedded object specifies additional LINK field code values of \f 0, which specifies that the embedded object must retain its source formatting (as defined in §17.16.5.32). *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
id (Relationship to Embedded Object Data) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID for the relationship which targets the Embedded Object Part containing the embedded object data. The specified relationship shall be of type http://purl.oclc.org/ooxml/officeDocument/relationships/oleObject or the document shall be considered non-conformant. [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId1" />` The markup specifies the associated relationship part with relationship ID `rId1` targets the part containing the corresponding embedded object information. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
lockedField (Object Refresh Flag)	Specifies whether the object's appearance is locked. If it is locked, the object's current representation shall be locked to prevent any user interaction or automatic application behavior from modifying its contents. [*Example*: `<w:objectLink … lockedField="true"/>` *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
progId (Object Application)	Specifies the application associated with the object. [*Example*: `<w:objectEmbed … progId="AVIFile"/>` *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

Attributes	Description
shapeId (Object Shape)	Specifies the shape with which the object is associated. A shape provides the visual placeholder for an object and this attribute is set to the ID of the placeholder shape. [*Example*: `<w:objectEmbed … shapeId="10"/>` *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
updateMode (Object Update Mode)	Specifies how the object is updated with new data - automatically or on-demand by the user. [*Example*: `<w:objectLink … updateMode="always"/>` *end example*] The possible values for this attribute are defined by the ST_ObjectUpdateMode simple type (§17.18.61).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ObjectLink) is located in §A.1. *end note*]

17.3.3.22 pgNum (Page Number Block)

This element specifies the presence of a page number block at the current location in the run content. A *page number block* is a non-editable region of text which shall display the current page using ascending decimal numbers. [*Note*: The page number block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the PAGENUM field is used in its place. *end note*]

A page number block shall be displayed using ascending decimal numbers, regardless of the languages specified in the parent run's lang property (§17.3.2.20).

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is the current page: </w:t>
  <w:pgNum />
</w:r>
```

This run specifies that a page number block must be placed after the text string literal `This is the current page:` in the document. Assuming that this content is on the first page, this run would be displayed as follows:

```
This is the current page: 1
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.3.3.23 ptab (Absolute Position Tab Character)

This element specifies that an absolute position tab character shall be placed at the current location in the run content. An *absolute position tab* is a character which is used to advance the position on the current line of text when displaying this WordprocessingML content, using the following logic:

Regardless of any number of custom tab stops defined using the tabs element (§17.3.1.38) , the absolute position tab character shall advance to the position specified by its alignment and relativeTo attributes. The resulting end position of the tab character shall not be affected by the addition of any custom tab stops or changes to the value of the defaultTabStop element (§17.15.1.25).

If the alignment location specified by the positional tab cannot be found on the current line, because the starting location is past that point, then the tab character shall advance to that location on the next available line in the document.

[*Example*: Consider a paragraph which contains two custom tab stops at 1.5" and 3.5", respectively. These two tab stops would be contained within a tabs element defining the set of tab stops of the paragraph as follows:

```
<w:pPr>
  <w:tabs>
    <w:tab w:val="start" w:pos="2160" />
    <w:tab w:val="start" w:pos="5040" />
  </w:tabs>
</w:pPr>
```

If a positional tab character was added to a run in this paragraph starting at 1" inside the margin and was defined as follows:

```
<w:ptab … w:alignment="center" w:relativeTo="margin" />
```

This positional tab would then ignore the next custom tab stop and the indents on the current paragraph defined using the ind element (§17.3.1.12) and would advance to the center of the line with respect to the text margins, moving to a new line if needed. *end example*]

Attributes	Description
alignment (Positional Tab Stop Alignment)	Specifies the location of the positional tab stop on the line, as well as the alignment which shall be applied to text subsequent to the current positional tab stop.

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	[*Example*: Consider a positional tab stop in a WordprocessingML document who must move to the left edge of the text margins and whose subsequent text should be left aligned. This positional tab stop would be defined as follows: `<w:ptab w:alignment="left" w:relativeTo="margin" … />` The alignment attribute specifies that this absolute position tab stop must align on the left edge of the line relative to the margin. *end example*] The possible values for this attribute are defined by the ST_PTabAlignment simple type (§17.18.71).
leader (Tab Leader Character)	Specifies the character which shall be used to fill in the space created by a positional tab. This character shall be repeated as required to completely fill the tab spacing generated by the positional tab character. [*Example*: Consider a positional tab stop which should be preceded by a sequence of underscore characters, as follows: _______________Text at the positional tab stop This tab stop would have a leader attribute value of `underscore`, indicating that the tab stop must be preceded by underscore characters as needed to fill the tab spacing. *end example*] The possible values for this attribute are defined by the ST_PTabLeader simple type (§17.18.72).
relativeTo (Positional Tab Base)	Specifies the extents which shall be used to calculate the absolute positioning of this positional tab character. [*Example*: Consider a positional tab stop in a WordprocessingML document that should have a resulting position that is centered on the text margins, ignoring both any custom tab stops and any text indents on the paragraph. This positional tab stop would be defined as follows: `<w:ptab w:relativeTo="margin" … />` The relativeTo attribute specifies that this absolute position tab stop must be relative to the margin. *end example*] The possible values for this attribute are defined by the ST_PTabRelativeTo simple type (§17.18.73).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PTab) is located in §A.1. *end note*]

17.3.3.24 rt (Phonetic Guide Text)

This element specifies the presence of the guide text within a phonetic guide at the current location in the document.

The contents of the guide text run are specified in the child r element (§17.3.2.25).

[*Example*: Consider the following two runs, each containing a phonetic guide:

tōkyō
東京

The guide text run would be specified using the following WordprocessingML:

```
<w:rt>
  …
  <w:r>
    <w:t>tō</w:t>
  </w:r>
</w:rt>
```

The guide text is contained in a run within the rt element. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_RubyContent) is located in §A.1. *end note*]

17.3.3.25 ruby (Phonetic Guide)

This element specifies the presence of a phonetic guide at the current location in the document. A *phonetic guide* (often called ruby text) is a run of content with base text which appears at the normal baseline location for text in this run, with phonetic guide text displayed above it in the document. The resulting construct is called a phonetic guide as it is typically used to map words in one language to another phonetically.

The base text is stored in the rubyBase element (§17.3.3.27) and the guide text is stored in the rt element (§17.3.3.24).

[*Example*: Consider the following two runs, each containing a phonetic guide:

tōkyō
東京

The first run would be specified using the following WordprocessingML:

```
<w:r>
  <w:ruby>
    <w:rubyPr>
      …
    </w:rubyPr>
    <w:rt>
      …
      <w:r>
        <w:t>tō</w:t>
      </w:r>
    </w:rt>
    <w:rubyBase>
      …
      <w:r>
        <w:t>東</w:t>
      </w:r>
    </w:rubyBase>
  </w:ruby>
</w:r>
```

The base text is contained in a run within the rubyBase element, and the guide text is contained in a run within the rt element. *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Ruby) is located in §A.1. *end note*]

17.3.3.26 rubyAlign (Phonetic Guide Text Alignment)

This element specifies the alignment setting which shall be used to determine the placement of phonetic guide text with respect to the base text when this phonetic guide is displayed.

[*Example*: Consider a run of phonetic guide text which must have the ruby text positioned to the far left of the base text. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:rubyAlign w:val="left"/>
  …
</w:rubyPr>
```

The rubyAlign property is `left` for the phonetic guide, so the ruby text is displayed on the left side of the base text. *end example*]

Attributes	Description
val (Phonetic Guide Text Alignment	Specifies the type of alignment to be applied to the phonetic guide text.

Attributes	Description

Value)	[*Example*: Consider a run of phonetic guide text which must have the ruby text positioned to the far right of the base text. This constraint is specified using the following WordprocessingML:
	`<w:rubyPr>`
	`  …`
	`  <w:rubyAlign w:val="left"/>`
	`  …`
	`</w:rubyPr>`
	The value of the val attribute is `right` for the phonetic guide, so the ruby text is displayed on the right side of the base text. *end example*]
	The possible values for this attribute are defined by the ST_RubyAlign simple type (§17.18.75).

[*Note*: The W3C XML Schema definition of this element's content model (CT_RubyAlign) is located in §A.1. *end note*]

17.3.3.27 rubyBase (Phonetic Guide Base Text)

This element specifies the presence of the base text within a phonetic guide at the current location in the document.

The contents of the base text run are specified in the child r element (§17.3.2.25).

[*Example*: Consider the following two runs, each containing a phonetic guide:

t ōkyō
東京

The base text run would be specified using the following WordprocessingML:

```
<w:rubyBase>
  …
  <w:r>
    <w:t>東/w:t>
  </w:r>
</w:rubyBase>
```

The base text is contained in a run within the rubyBase element. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_RubyContent) is located in §A.1. *end note*]

©ISO/IEC 2012 – All rights reserved

17.3.3.28 rubyPr (Phonetic Guide Properties)

This element specifies a set of properties which determine the behavior and appearance of a phonetic guide within the document.

[*Example*: Consider the following two runs, each containing a phonetic guide:

tōkyō
東京

The properties for both of these phonetic guides are as follows:

```
<w:r>
  <w:ruby>
    <w:rubyPr>
      <w:rubyAlign w:val="distributeSpace" />
      <w:hps w:val="16" />
      <w:hpsRaise w:val="20" />
      <w:hpsBaseText w:val="22" />
      <w:lid w:val="ja-JP" />
    </w:rubyPr>
    …
  </w:ruby>
</w:r>
```

The phonetic guide properties specify that the guide text must be:

- Distributed across the top (using the rubyAlign element)
- 8 point font face (using the hps element)
- 10 points above the base text (using the hpaRaise element)
- Japanese (using the lid element)

As well, the phonetic guide properties specify that the base text must be:

- 11 point font face (using the hpsBaseText element)

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_RubyPr) is located in §A.1. *end note*]

17.3.3.29 softHyphen (Optional Hyphen Character)

This element specifies that an optional hyphen character shall be placed at the current location in the run content. An *optional hyphen* is a character which can be used as a line breaking character for the current line of text when displaying this WordprocessingML content, using the following logic:

- When this character is not the character which is used to break the line, then it shall not change the normal display of text (it shall have zero width)
- When this character is the character used to break the line, it shall display using the hyphen-minus character within the display of text

[*Note*: This character is typically used to mark locations where a word can optionally be hyphenated without causing the hyphen character to be displayed unnecessarily. *end note*]

[*Example*: Consider the following sentence in a WordprocessingML document:

```
This sentence needs to be long enough to cause some kind of line breaking.
```

Normally, just as shown above, this sentence not would be displayed on a single line as it is long enough to require line breaking (given the width of the current page). However, if an optional hyphen were inserted after the letter r in breaking, as follows:

```
<w:r>
  <w:t>This sentence needs to be long enough to cause some kind of line br</w:t>
  <w:softHyphen/>
  <w:t>eaking.</w:t>
</w:r>
```

This would allow a break at that position, and when that location is the point of the line break, would insert a hyphen-minus in the word after that character:

```
This sentence needs to be long enough to cause some kind of line breaking.
```

If this was not the point of the line break, then no character would be displayed at that location:

```
This sentence should not be long enough to cause line breaking.
```

The sentence now does not break at that location, so no hyphen appears in the word breaking. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.3.3.30 sym (Symbol Character)

This element specifies the presence of a symbol character at the current location in the run's content. A *symbol character* is a special character within a run's content which does not use any of the run fonts specified in the rFonts element (§17.3.2.26) (or by the style hierarchy).

Instead, this character shall be determined by pulling the character with the hexadecimal value specified in the char attribute from the font specified in the font attribute.

[*Example*: Consider a run containing the following run content:

```
This is a symbol character:💻
```

The last character in that run is a symbol character from the Wingdings font, and the run is specified as follows:

```
<w:r>
  <w:rPr>
    <w:rFonts w:ascii="Courier New" w:hAnsi="Courier New" />
  </w:rPr>
  <w:t>This is a symbol character:</w:t>
  <w:sym w:font="Wingdings" w:char="F03A" />
</w:r>
```

The resulting symbol is the specified using the sym element, and consists of character code `003A` formatted as `Wingdings`, even though the run properties specify the `Courier New` font. *end example*]

Attributes	Description

char (Symbol Character Code)	Specifies the hexadecimal code for the Unicode character value of the symbol.
	When this value is stored in the char attribute, it can be stored in either of the following two formats:
	Directly in its Unicode character value from the font glyph
	In a Unicode character value created by adding F000 to the actual character value, shifting the character value of this character into the Unicode private use area.
	[*Note*: The use of the latter syntax allows for interoperability with legacy word processing formats, as they used this technique to store the fact that a particular character or set of characters came from a font which was not Unicode compliant, and therefore any font matching performed on this range (if the specified font was not present) would be undesirable, as the resulting glyphs and their appearance could not be predicted. *end note*]
	[*Example*: Consider a run with a single symbol character defined as follows:
	`<w:r>`
	`  <w:rPr>`
	`    <w:rFonts w:ascii="Arial Black" w:hAnsi="Arial Black" />`
	`  </w:rPr>`
	`  <w:sym w:font="Wingdings" w:char="F045" />`
	`</w:r>`
	The symbol character must use the font defined in its font attribute and hence use the `Wingdings` font. The character value for the character to be used from this font is obtained by removing the F000 value from the value in the char attribute, and therefore is the character at hexadecimal position 0045 in that font. *end example*]
	The possible values for this attribute are defined by the ST_ShortHexNumber simple type (§17.18.79).

font (Symbol	Specifies a font which shall be used to format this symbol character.

Attributes	Description

Character Font)	[*Example*: Consider a run with a single symbol character defined as follows:
	
	`<w:r>`
	`  <w:rPr>`
	`    <w:rFonts w:ascii="Arial Black" w:hAnsi="Arial Black" />`
	`  </w:rPr>`
	`  <w:sym w:font="Wingdings" w:char="F045" />`
	`</w:r>`
	
	Although the run specifies that its contents must use the `Arial Black` font, the symbol character must use the font defined in its font attribute and hence use the `Wingdings` font. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Sym) is located in §A.1. *end note*]

17.3.3.31 t (Text)

This element specifies that this run contains literal text which shall be displayed in the document. The `t` element shall be used for all text runs which are not:

- Part of a region of text that is contained in a deleted region using the del element (§17.13.5.14)
- Part of a region of text that is contained within a field code

[*Example*: Consider a paragraph of WordprocessingML content which reads `This is text`. This paragraph would therefore be represented as follows:

```
<w:p>
  <w:r>
    <w:t>This is text</w:t>
  </w:r>
</w:p>
```

The text is contained in a t node. *end example*]

Attributes	Description
xml:space (Content Contains Significant Whitespace) Namespace: http://www.w3.or	Specifies how white space should be handled for the contents of this element using the W3C space preservation rules. [*Example*: Consider the following run contained within a WordprocessingML document: `<w:r>`

Attributes	Description

g/XML/1998/namespace	`<w:t>   significant whitespace   </w:t>`
	`</w:r>`
	Although there are three spaces on each side of the text content in the run, that whitespace has not been specifically marked as significant, therefore it is subject to the space preservation rules currently specified in that run's scope. *end example*]
	The possible values for this attribute are defined by §2.10 of the XML 1.0 specification.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Text) is located in §A.1. *end note*]

17.3.3.32 tab (Tab Character)

This element specifies that a tab character shall be placed at the current location in the run content. An *tab* is a character which is used to advance the position on the current line of text when displaying this WordprocessingML content, using the following logic:

- When there are one or more custom tab stops defined using the tabs element (§17.3.1.38) , then the tab character shall advance to the next custom tab stop location which is further along than the starting location of the tab
- When there are no custom tab stops which are further than the current position in the line, the tab character shall advance to the nearest multiple of the defaultTabStop element (§17.15.1.25) width value.

[*Example*: Consider a paragraph which contains two custom tab stops at 1.5" and 3.5", respectively. These two tab stops would be contained within a tabs element defining the set of tab stops of the paragraph as follows:

```
<w:pPr>
  <w:tabs>
    <w:tab w:val="start" w:pos="2160" />
    <w:tab w:val="start" w:pos="5040" />
  </w:tabs>
</w:pPr>
```

If a tab character was added to a run in this paragraph and appeared 1.4” along the line after all preceding content was laid out, then this tab would move the position to 1.5”. If the tab character appeared 1.6” along the line after all preceding content was laid out, then this tab would move the position to 3.5”. In both cases, the tab advanced to the next custom tab stop. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.3.3.33 yearLong (Date Block - Long Year Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs). [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run's lang property (§17.3.2.20).

The long year format date block shall use a date picture of YYYY, retrieving the long year format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a long date: </w:t>
  <w:yearLong />
</w:r>
```

This run specifies that a long year format date block must be placed after the text string literal `This is a long date:` in the document. Assuming that the host application's primary editing language is English (Canada) and today's date is 2006-04-12, this run would be displayed as follows:

```
This is a long date: 2006
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.3.3.34 yearShort (Date Block - Short Year Format)

This element specifies the presence of a date block at the current location in the run content. A *date block* is a non-editable region of text which shall display the current date filtered through the specified date picture (see following paragraphs). [*Note*: The date block is a legacy construct used for compatibility with older word processors, and should not be produced unless it was consumed while reading a document – it is recommended that the DATE field is used in its place. *end note*]

A date block shall be displayed using the primary editing language of the host application, regardless of the languages specified in the parent run's lang property (§17.3.2.20).

The short year format date block shall use a date picture of YY, retrieving the short year format for the primary editing language.

[*Example*: Consider a WordprocessingML run with the following run content:

```
<w:r>
  <w:t xml:space="preserve">This is a short date: </w:t>
  <w:yearShort />
</w:r>
```

©ISO/IEC 2012 – All rights reserved

This run specifies that a short year format date block shall be placed after the text string literal `This is a short date:` in the document. Assuming that the host application's primary editing language is French (Canada) and today's date is 2006-04-12, this run would be displayed as follows:

```
This is a short date: 06
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.3.4 Border Properties (CT_Border)

This common complex type specifies the set of attributes used to define a object's border.

[*Example*: Consider the following run border:

```
<w:r>
  <w:rPr>
    <w:bdr w:val="single" w:sz="36" w:space="0" w:themeColor="accent1"
w:themeTint="66" />
  </w:rPr>
  <w:t xml:space="preserve">run one</w:t>
</w:r>
```

The bdr element specifies a single line border with a width of 3.5 points using the document's `accent1` theme color. *end example*]

Attributes	Description
color (Border Color)	Specifies the color for this border. This value can be defined as either: • A color value using the RGB color model whose red, green, and blue values are written as numbers in the range 0 to 255, hex encoded, and concatenated. [*Example*: Full intensity red would be 255 red, 0 green, 0 blue, encoded to FF, 00, 00, and concatenated to `FF0000`. *end example*] . RGB colors are specified in the sRGB color space. • `auto` to allow a consumer to automatically determine the border color in order to make the document's text readable. [*Example*: A document with white text and a background color of `auto` might result in the use of a black background, in order to ensure legibility of the content. *end example*] [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore can be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's

Attributes	Description
	background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§17.18.38).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which must appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" … />` This frame's val is `true`, indicating that the border frame effect must be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which must appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" … />` This frame's val is `true`, indicating that the shadow effect must be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§17.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.

Attributes	Description

	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§17.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24" />`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value provides the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset must be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§17.18.68).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range can be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range can be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§17.18.23).

themeColor (Border Theme	Specifies the base theme color used to generate the border color. The border color is the RGB value associated with themeColor as further transformed by themeTint or

Attributes	Description

Color)	themeShade (if one is present), else the background color is the RGB value associated with themeColor.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part (§14.2.7 and §20.1.6.9), which allows color information to be set centrally in the document.
	
	To determine the color to display, the following actions are performed:
	• Using the mapping specified in the ST_ThemeColor simple type (§17.18.97), the appropriate attribute on the clrSchemeMapping element (§17.15.1.20) is read.
	• Using that value and the mapping specified in the ST_ColorSchemeIndex simple type (§17.18.103), the appropriate element in the document's Theme part is read to get the base theme color.
	• The specified color is modified based on the presence of the themeTint or themeShade attribute.
	
	[*Example*: Consider a set of borders configured to use the `accent2` theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:themeColor="accent2" w:themeTint="99" />`
	`<w:bottom … w:themeColor="accent2" w:themeTint="99" />`
	`<w:left … w:themeColor="accent2" w:themeTint="99" />`
	`<w:right … w:themeColor="accent2" w:themeTint="99" />`
	
	If the Settings part contained the following markup:
	
	`<w:clrSchemeMapping … w:accent2="accent2"/>`
	
	and the Theme part contained the following XML markup:
	
	`<a:accent2>`
	`  <a:srgbClr val="4F81BD"/>`
	`</a:accent2>`
	
	the resulting border color would be `95B3D7` (the result of a 60% tint applied to the original theme color; see the calculations in themeTint below for details). *end example*]
	
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§17.18.97).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeColor attribute is not present, then this attribute shall not be used. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255)

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

applied to the current border.

[*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows:

$$\begin{aligned} S_{xml} &= 0.4 * 255 \\ &= 102 \\ &= 66(hex) \end{aligned}$$

The resulting themeShade value in the file format would be `66`. *end example*]

Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows:

- Convert the color to the HSL color format (values from 0 to 1)
- Modify the luminance factor as follows:

$$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$

- Convert the resultant HSL color to RGB

[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`.

The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$.

Applying the shade formula with a shade percentage of 75% to the luminance, we get:

$$\begin{aligned} L' &= 0.53 * 0.75 \\ &= 0.39698 \end{aligned}$$

Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get `943634`.

This transformed value can be seen in the resulting background's color attribute:

```
<w:top w:val="single" w:sz="4" w:space="24"
  w:color="943634" w:themeColor="accent2"
  w:themeShade="BF"/>
```

end example]

The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

</td></tr>
<tr><td>themeTint (Border Theme Color Tint)</td><td>Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeColor attribute is not present, then this attribute shall not be used.</td></tr>
</table>

Attributes	Description

	If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border.
	The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border.
	[*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows:
	$\begin{aligned} T_{xml} &= 0.6 * 255 \\ &= 153 \\ &= 99(hex) \end{aligned}$
	The resulting themeTint value in the file format would be 99. *end example*]
	Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows:
	• Convert the color to the HSL color format (values from 0 to 1)
	• Modify the luminance factor as follows:
	$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$
	• Convert the resultant HSL color to RGB
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`.
	The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$.
	Applying the tint formula with a tint percentage of 60% to the luminance, we get:
	$\begin{aligned} L' &= 0.53 * 0.6 + (1 - .6) \\ &= 0.71 \end{aligned}$
	Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`.
	This transformed value can be seen in the resulting background's color attribute:
	`<w:top w:val="single" w:sz="4" w:space="24"`
	`  w:color="95B3D7" w:themeColor="accent2"`
	`  w:themeTint="99"/>`
	end example]

Attributes	Description
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - shall only be used for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§17.18.2).

[*Note*: The W3C XML Schema definition of this complex type's content model (CT_Border) is located in §A.1. *end note*]

17.3.5 Shading Properties (CT_Shd)

This common complex type specifies the set of attributes used to define a object's shading.

[*Example*: Consider the following paragraph shading:

```
<w:pPr>
  <w:shd w:val="pct20" w:themeColor="accent6" w:themeFill="accent3" />
</w:pPr>
```

The resulting paragraph uses the background color `accent3` under the foreground pattern color `accent6` as specified by the `pct20` pattern mask. *end example*]

Attributes	Description
color (Shading Pattern Color)	Specifies the color used for any foreground pattern specified for this shading using the val attribute. This color can either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the foreground shading color as appropriate. If the shading style (the val attribute) specifies the use of no shading format or is omitted, then this property has no effect. Also, if the shading specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value.

Attributes	Description
	If this attribute is omitted, then its value shall be assumed to be auto. [*Example*: Consider a shading of style pct20 with a foreground color value of auto, as follows: `<w:shd w:val="pct20"… w:color="auto"/>` The foreground color for this shading pattern therefore can be automatically be modified by a consumer as appropriate, for example, in order to ensure that the shading color can be distinguished against the page's background color. *end example*] The possible values for this attribute are defined by the ST_HexColor simple type (§17.18.38).
fill (Shading Background Color)	Specifies the color used for the background for this shading. This color can either be presented as a hex value (in RRGGBB format), or auto to allow a consumer to automatically determine the background shading color as appropriate. If this attribute is omitted, then its value shall be assumed to be auto. [*Example*: Consider a shading using a background color of hex value C3D69B, using the following WordprocessingML: `<w:shd w:val="pct15" w:fill="C3D69B" />` The background color for this shading therefore is a color with a hex value of C3D69B. *end example*] If the shading specifies the use of a theme color via the themeFill attribute, then this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§17.18.38).
themeColor (Shading Pattern Theme Color)	Specifies a theme color which should be applied to any foreground pattern specified for this shading using the val attribute. The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document. If this element is omitted, then no theme color is applied, and the color attribute shall be used to determine the shading pattern color. [*Example*: Consider a paragraph which must have a background consisting of a theme

Attributes	Description

	color `accent3` with a theme color `accent6` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:
	`<w:pPr>`
	`  <w:shd w:val="pct20" w:themeColor="accent6"`
	`w:themeFill="accent3" />`
	`</w:pPr>`
	The resulting paragraph uses the foreground pattern color `accent6` in the region specified by the `pct20` pattern mask. *end example*]
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§17.18.97).

themeFill (Shading Background Theme Color)	Specifies a theme color which should be applied to the background for this shading.
	The specified theme color is a reference to one of the predefined theme colors, located in the document's themes part, which allows for color information to be set centrally in the document.
	If this element is omitted, then no theme color is applied, and the color attribute shall be used to determine the shading background color.
	[*Example*: Consider a paragraph which must have a background consisting of a theme color `accent3` with a theme color `accent6` overlaid using a 20% fill pattern. This requirement is specified using the following WordprocessingML:
	`<w:shd w:val="pct20" w:themeColor="accent6"`
	`w:themeFill="accent3" />`
	The resulting shading uses the background color specified by the `accent3` theme color. *end example*]
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§17.18.97).

themeFillShade (Shading Background Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this shading color. If the themeFillShade is supplied, then it is applied to the RGB value of the themeFill color (from the theme part) to determine the final color applied to this border. The themeFillShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border. [*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows:

Attributes	Description

	$\begin{aligned} S_{xml} &= 0.4 * 255 \\ &= 102 \\ &= 66(hex) \end{aligned}$
	
	The resulting themeFillShade value in the file format would be 66. *end example*]
	
	Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows:
	• Convert the color to the HSL color format (values from 0 to 1)
	• Modify the luminance factor as follows:
	
	$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$
	
	• Convert the resultant HSL color to RGB
	
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`.
	
	The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$.
	
	Applying the shade formula with a shade percentage of 75% to the luminance, we get:
	
	$\begin{aligned} L' &= 0.53 * 0.75 \\ &= 0.39698 \end{aligned}$
	
	Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get `943634`.
	
	This transformed value can be seen in the resulting shading's fill attribute:
	
	`<w:shd w:val="pct15" w:fill="943634" w:themeFill="accent2"`
	`  w:themeFillShade="BF" />`
	
	end example]
	
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

themeFillTint (Shading Background Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance. If the themeFillTint is supplied, then it is applied to the RGB value of the themeFill color (from the theme part) to determine the final color applied to this border. The themeFillTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border.

Attributes	Description

	[*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows:
	$T_{xml} = 0.6 * 255$
	$= 153$
	$= 99(hex)$
	The resulting themeFillTint value in the file format would be 99. *end example*]
	Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows:
	• Convert the color to the HSL color format (values from 0 to 1)
	• Modify the luminance factor as follows:
	$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$
	• Convert the resultant HSL color to RGB
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`.
	The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45,0.53\right)$.
	Applying the tint formula with a tint percentage of 60% to the luminance, we get:
	$L' = 0.53 * 0.6 + (1 - .6)$
	$= 0.71$
	Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45,0.71\right)$and converting back to RGB, we get `95B3D7`.
	This transformed value can be seen in the resulting shading's fill attribute:
	`<w:top w:val="single" w:sz="4" w:space="24"`
	`w:fill="95B3D7" w:themeFillColor="accent2"`
	`w:themeFillTint="99" />`
	end example]
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

themeShade (Shading Pattern Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this shading color.

Attributes	Description

If the themeShade is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border.

The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border.

[*Example*: Consider a shade of 40% applied to a background shading color in a document. This shade is calculated as follows:

$$\begin{aligned} S_{xml} &= 0.4 * 255 \\ &= 102 \\ &= 66(hex) \end{aligned}$$

Te resulting themeShade value in the file format would be 66. *end example*]

Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows:

- Convert the color to the HSL color format (values from 0 to 1)
- Modify the luminance factor as follows:

$$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$$

- Convert the resultant HSL color to RGB

[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`.

The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48, 0.53\right)$.

Applying the shade formula with a shade percentage of 75% to the luminance, we get:

$$\begin{aligned} L' &= 0.53 * 0.75 \\ &= 0.39698 \end{aligned}$$

Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48, 0.39698\right)$and converting back to RGB, we get `943634`.

This transformed value can be seen in the resulting background's color attribute:

```
<w:shd w:val="pct15" w:color="943634" w:themeColor="accent2"
  w:themeShade="BF" />
```

end example]

The possible values for this attribute are defined by the ST_UcharHexNumber simple

Attributes	Description
	type (§17.18.98).

themeTint (Shading Pattern Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this shading instance.
	
	If the themeTint is supplied, then it is applied to the RGB value of the themeColor color (from the theme part) to determine the final color applied to this border.
	
	The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border.
	
	[*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows:
	
	$T_{xml} = 0.6 * 255$
	$= 153$
	$= 99(hex)$
	
	The resulting themeTint value in the file format would be 99. *end example*]
	
	Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows:
	- Convert the color to the HSL color format (values from 0 to 1)
	- Modify the luminance factor as follows:
	
	$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$
	
	- Convert the resultant HSL color to RGB
	
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`.
	
	The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45,0.53\right)$.
	
	Applying the tint formula with a tint percentage of 60% to the luminance, we get:
	
	$L' = 0.53 * 0.6 + (1 - .6)$
	$= 0.71$
	
	Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45,0.71\right)$and converting back to RGB, we get `95B3D7`.
	
	This transformed value can be seen in the resulting shading's color attribute:
	
	`<w:shd w:val="pct15" w:color="95B3D7" w:themeColor="accent2"`
	`  w:themeTint="99" />`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).
val (Shading Pattern)	Specifies the pattern which shall be used to lay the pattern color over the background color for this paragraph shading. This pattern consists of a mask which is applied over the background shading color to get the locations where the pattern color should be shown. Each of these possible masks are shown in the simple type values referenced below. [*Example*: Consider a shaded paragraph which uses a 10 percent foreground fill, resulting in the following WordprocessingML: `<w:shd w:val="pct10" … />` This shading val is `pct10`, indicating that the border style is a 10 percent foreground fill mask. *end example*] The possible values for this attribute are defined by the ST_Shd simple type (§17.18.78).

[*Note*: The W3C XML Schema definition of this complex type's content model (CT_Shd) is located in §A.1. *end note*]

17.4 Tables

Another form of block-level content in WordprocessingML, a *table* is a set of paragraphs (and other block-level content) arranged in *rows* and *columns*. Tables in WordprocessingML are defined via the tbl element, which is analogous to the HTML `<table>` tag. The table element specifies the location of a table present in the document.

A tbl element (§17.4.37) has two elements that define its properties:

- tblPr (§17.4.59), which defines the set of table-wide properties (such as style and width)
- tblGrid (§17.4.48), which defines the grid layout of the table.

A tbl element can also contain an arbitrary non-zero number of rows, where each row is specified with a tr element (§17.4.78). Each tr element can contain an arbitrary non-zero number of cells, where each cell is specified with a tc element (§17.4.65).

[*Example*: Consider an empty one-cell table (i.e.; a table with one row, one column) and 1 point borders on all sides as follows:

This table is represented by the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:w="5000" w:type="pct"/>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:start w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:end w:val="single" w:sz="4" w:space="0" w:color="auto"/>
    </w:tblBorders>
  </w:tblPr>
  <w:tblGrid>
    <w:gridCol w:w="10296"/>
  </w:tblGrid>
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:tcW w:w="0" w:type="auto"/>
      </w:tcPr>
      <w:p/>
    </w:tc>
  </w:tr>
</w:tbl>
```

This table specifies table-wide properties of 100% of page width using the tblW element and the set of table borders using the tblBorders element, the table grid which defines a set of shared vertical edges within the table using the tblGrid element, and a single row using the tr element. *end example*]

17.4.1 bidiVisual (Visually Right to Left Table)

This element specifies that the cells with this table shall be visually represented in a right to left direction. This element also affects the application of all table-level properties.

When this property is specified, then the ordering of all cells (and table-level properties) in this table shall be applied to the table assuming that the table is a normal left to right table, but the table cells shall be displayed in a right to left direction. [*Example*: A left border on the first table cell must be displayed on the right side of that cell (which would be the rightmost cell) in a visually right to left table. *end example*]

If this element is omitted, then the table shall not be presented right to left.

[*Example*: Consider the following table which has the logical right to left property set:

	Three	Two	One

This property would be specified in the WordprocessingML as follows:

```
<w:tblPr>
  <w:bidiVisual/>
</w:tblPr>
```

Since the bidiVisual element specifies this is a visually right to left table, the actual table data would be stored in its logical order as follows:

```
<w:tr>
  <w:tc>
    <w:p>
      <w:r>
        <w:t>One</w:t>
      </w:r>
    </w:p>
  </w:tc>
  <w:tc>
    <w:p>
      <w:r>
        <w:t>Two</w:t>
      </w:r>
    </w:p>
  </w:tc>
  <w:tc>
    <w:p>
      <w:r>
        <w:t>Three</w:t>
      </w:r>
    </w:p>
  </w:tc>
  <w:tc>
    <w:p/>
  </w:tc>
</w:tr>
```

The first logical cell with text One is stored first in the file format, and displayed on the rightmost in this table visually. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.4.2 bottom (Table Cell Bottom Margin Exception)

This element specifies the amount of space which shall be left between the bottom extent of the cell contents and the border of a specific table cell within a table. This setting shall override the table cell bottom margin definition specified by the bottom element contained within the table properties (§17.4.5).

This value is specified in the units applied via its type attribute. Any width value of type pct or auto for this element shall be ignored.

If omitted, then this table cell shall use the bottom cell margins defined in the bottom element contained within the table properties (§17.4.5).

[*Example*: Consider a table with two cells in which the first table cell's bottom margin is specified via an exception to be ten times larger (0.2 inches vs. 0.02 inches) than the other table cell margins:

This text fills the extents of the cell.
So does this

The first cell in the table would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcMar>
      <w:bottom w:w="288" w:type="dxa" />
    </w:tcMar>
  </w:tcPr>
</w:tc>
```

The first cell in this table has an exception applied to the table cell bottom cell margin setting it to 288 twentieths of a point (0.2 inches). *end example*]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.3 bottom (Table Cell Bottom Border)

This element specifies the border which shall be displayed at the bottom of the current table cell. The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§17.4.44;§17.4.43;§17.4.45) applied to the cell is non-zero, then the cell border shall always be displayed
- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

If this element is omitted, then the bottom of this table cell shall not have a cell border, and its border can use the table's border settings as appropriate.

[*Example*: Consider a table in which the first cell in the first row specifies a bottom cell border

R1C1	R1C2
R2C1	R2C2

This bottom cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
      <w:bottom w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The bottom element specifies a three point border of type `thinThinThickSmallGap`. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.4.4 bottom (Table Bottom Border)

This element specifies the border which shall be displayed at the bottom of the current table. The appearance of this table border in the document shall be determined by the following settings:

- The display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

If this element is omitted, then the bottom of this table shall have the border specified by the associated table style. If no bottom border is specified in the style hierarchy, then this table shall not have a bottom border.

[*Example*: Consider a table in which the table properties specifies a bottom table border, as follows:

R1C1	R1C2
R2C1	R2C2

This bottom table border is specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblBorders>
      <w:bottom w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="D0D0D0" w:themeColor="accent3" w:themeTint="99"/>
    </w:tblBorders>
  </w:tblPr>
  …
</w:tbl>
```

The bottom element specifies a three point bottom table border of type `thinThinThickMediumGap`. *end example*]

This element’s content model is defined by the common border properties definition in §17.3.4.

17.4.5 bottom (Table Cell Bottom Margin Default)

This element specifies the amount of space which shall be left between the bottom extent of the cell contents and the border of all table cells within the parent table (or table row). This setting can be overridden by the table cell bottom margin definition specified by the bottom element contained within the table cell's properties (§17.4.2).

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If this element is omitted, then it shall inherit the table cell margin from the associated table style. If a bottom margin is never specified in the style hierarchy, then this table shall have no bottom cell padding by default (excepting individual cell overrides).

[*Example*: Consider a two by two table in which the default table cell bottom margin is specified to be exactly 0.25 inches, as follows (marked with an arrow in the first table cell below):

R1C1 ↕	R2C1
R2C1	R2C2

This table property is specified using the following WordprocessingML markup:

```
<w:tbl>
  <w:tblPr>
    <w:tblCellMar>
      <w:bottom w:w="360" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  …
</w:tbl>
```

Every cell in the table has a default cell margin setting it to 360 twentieths of a point. *end example*]

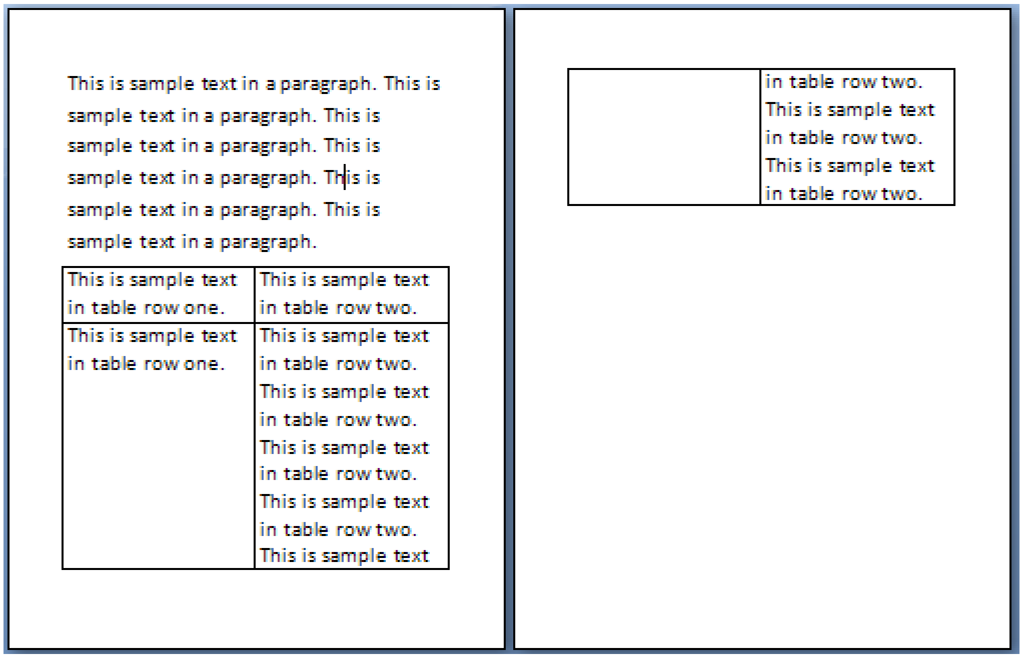
This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.6 cantSplit (Table Row Cannot Break Across Pages)

This element specifies whether the contents within the current cell shall be rendered on a single page. When displaying the contents of a table cell (such as the table cells in ISO/IEC 29500), it is possible that a page break would fall within the contents of a table cell, causing the contents of that cell to be displayed across two different pages. If this property is set, then all contents of a table row shall be rendered on the same page by moving the start of the current row to the start of a new page if necessary. If the contents of this table row cannot fit on a single page, then this row shall start on a new page and flow onto multiple pages as necessary.

If this element is not present, the default behavior is dictated by the setting in the associated table style. If this property is not specified in the style hierarchy, then this table row shall be allowed to split across multiple pages.

[*Example*: Consider the following content displayed on two different pages of a WordprocessingML document:



This is sample text in a paragraph. This is sample text in a paragraph. This is sample text in a paragraph. This is sample text in a paragraph. This is sample text in a paragraph. This is sample text in a paragraph.

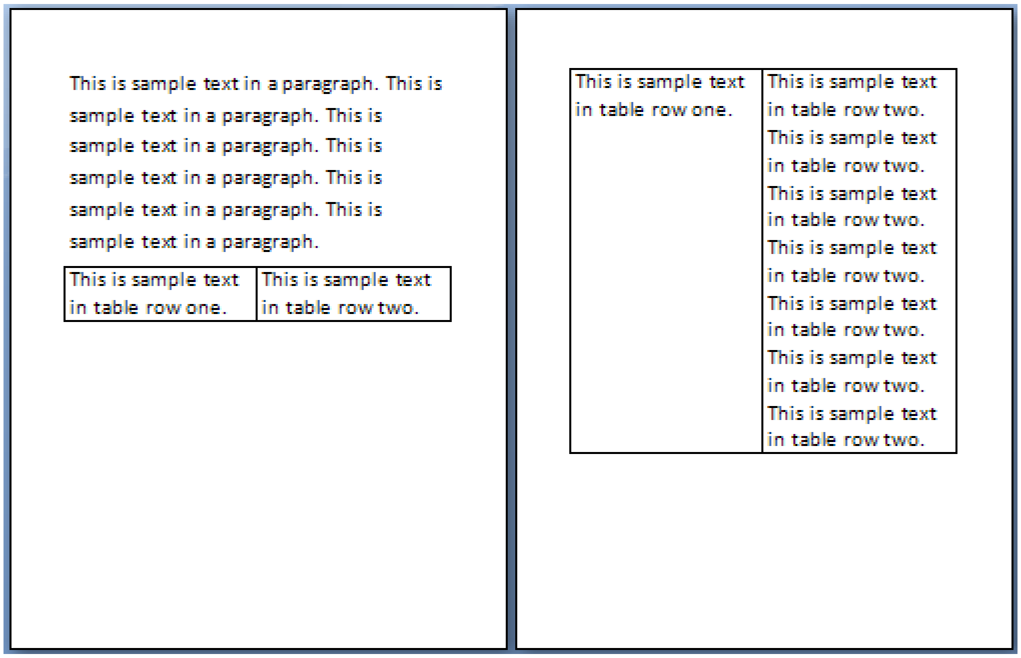
This is sample text in table row one.	This is sample text in table row two.
This is sample text in table row one.	This is sample text in table row two. This is sample text in table row two. This is sample text in table row two. This is sample text in table row two. This is sample text

	in table row two. This is sample text in table row two. This is sample text in table row two.

When this content is displayed, the contents of the 2nd table row in this document are displayed across two different pages. If the contents of this row are to be displayed on one page, then this requirement would be specified as follows:

```
<w:tr>
  <w:trPr>
    …
    <w:cantSplit />
  </w:trPr>
  …
</w:tr>
```

The presence of the cantSplit element specifies that the table row must not be broken across multiple pages, therefore the second table row starts on a new page:



This is sample text in a paragraph. This is sample text in a paragraph. This is sample text in a paragraph. This is sample text in a paragraph. This is sample text in a paragraph. This is sample text in a paragraph.

This is sample text in table row one.	This is sample text in table row two.

This is sample text in table row one.	This is sample text in table row two. This is sample text in table row two. This is sample text in table row two. This is sample text in table row two. This is sample text in table row two. This is sample text in table row two. This is sample text in table row two.

This setting therefore ensures that the content is always displayed on a single page (if it fits on one page). *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.4.7 cnfStyle (Table Row Conditional Formatting)

This element specifies the set of conditional table style formatting properties which have been applied to this table row. [*Note*: This property is an optimization which is used by consumers to determine if a given property on a table row is the result of the table style conditional formatting properties vs. direct formatting on the table cell itself. It specifies the components of the conditional formatting in the table style applied to this cell, so that the table's conditional formatting can be applied after the document is displayed without having the table style properties override the style hierarchy. *end note*]

If this element is omitted, then its value shall be assumed to be zero for all entries in the bit mask.

[*Example*: Consider a table row in the top of a table with a table style applied. This table cell would need to specify the following WordprocessingML to express that fact:

```
<w:tr>
  <w:trPr>
    <w:cnfStyle w:firstRow="true" />
    …
  </w:trPr>
  …
</w:tr>
```

This table row specifies that it has the conditional properties from the table style for the first row of the parent table by setting the appropriate attribute value. *end example*]

Attributes	Description

evenHBand (Even Numbered Horizontal Band)	Specifies that the object has inherited the conditional properties applied to the even numbered horizontal bands of the parent object.
	[*Example*: Consider a paragraph in the second row of a table with a table style applied, and where the band width is one row. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:evenHBand="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the even numbered horizontal bands of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

evenVBand (Even Numbered Vertical Band)	Specifies that the object has inherited the conditional properties applied to the even numbered vertical bands of the parent object.
	[*Example*: Consider a paragraph in the second column of a table with a table style applied, and where the band width is one column. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:evenVBand="true" />`
	`    …`

Attributes	Description

	`</w:pPr>`
	`…`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the even numbered vertical bands of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstColumn (First Column)	Specifies that the object has inherited the conditional properties applied to the first column of the parent object.
	
	[*Example*: Consider a paragraph in the first column of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`<w:pPr>`
	`<w:cnfStyle w:firstColumn="true" />`
	`…`
	`</w:pPr>`
	`…`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the first column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstRow (First Row)	Specifies that the object has inherited the conditional properties applied to the first row of the parent object.
	
	[*Example*: Consider a paragraph in the top row of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`<w:pPr>`
	`<w:cnfStyle w:firstRow="true" />`
	`…`
	`</w:pPr>`
	`…`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the first row of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstRowFirstColumn (First Row and	Specifies that the object has inherited the conditional properties applied to the cell that is in the first row and first column of the parent object.

Attributes	Description

First Column)	[*Example*: Consider a paragraph in the first row and first column of a table. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:firstRow="true" w:firstColumn="true"`
	`w:firstRowFirstColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the cell in the first row and first column of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstRowLastColumn (First Row and Last Column)	Specifies that the object has inherited the conditional properties applied to the cell that is in the first row and last column of the parent object.
	[*Example*: Consider a paragraph in the first row and last column of a table. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:firstRow="true" w:lastColumn="true"`
	`w:firstRowLastColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the cell in the first row and last column of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastColumn (Last Column)	Specifies that the object has inherited the conditional properties applied to the last column of the parent object.
	[*Example*: Consider a paragraph in the last column of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:lastColumn="true" />`
	`    …`

Attributes	Description

	`    </w:pPr>`
	`    …`
	`  </w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the last column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastRow (Last Row)	Specifies that the object has inherited the conditional properties applied to the last row of the parent object.
	
	[*Example*: Consider a paragraph in the bottom row of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	
	`  <w:p>`
	`    <w:pPr>`
	`      <w:cnfStyle w:lastRow="true" />`
	`      …`
	`    </w:pPr>`
	`    …`
	`  </w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the last row of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastRowFirstColumn (Last Row and First Column)	Specifies that the object has inherited the conditional properties applied to the cell that is in the last row and first column of the parent object.
	
	[*Example*: Consider a paragraph in the last row and first column of a table. This paragraph would need to specify the following WordprocessingML:
	
	`  <w:p>`
	`    <w:pPr>`
	`      <w:cnfStyle w:lastRow="true" w:firstColumn="true"`
	`  w:lastRowFirstColumn="true" />`
	`      …`
	`    </w:pPr>`
	`    …`
	`  </w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the cell in the last row and first column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastRowLastColum	Specifies that the object has inherited the conditional properties applied to the cell that

Attributes	Description

n (Last Row and Last Column)	is in the last row and last column of the parent object.
	[*Example*: Consider a paragraph in the last row and last column of a table. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:lastRow="true" w:lastColumn="true"`
	`w:lastRowLastColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the cell in the last row and last column of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

oddHBand (Odd Numbered Horizontal Band)	Specifies that the object has inherited the conditional properties applied to the odd numbered horizontal bands of the parent object.
	[*Example*: Consider a paragraph in the third row of a table with a table style applied, and where the band width is one column. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:oddHBand="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the odd numbered horizontal bands of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

oddVBand (Odd Numbered Vertical Band)	Specifies that the object has inherited the conditional properties applied to the odd numbered vertical bands of the parent object.
	[*Example*: Consider a paragraph in the third column of a table with a table style applied, and where the band width is one column. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`

Attributes	Description

	`<w:cnfStyle w:oddVBand="true" />`
	`…`
	`</w:pPr>`
	`…`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the odd numbered vertical bands of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Cnf) is located in §A.1. *end note*]

17.4.8 cnfStyle (Table Cell Conditional Formatting)

This element specifies the set of conditional table style formatting properties which have been applied to this table cell. [*Note*: This property is an optimization which is used by consumers to determine if a given property on a table cell is the result of the table style conditional formatting properties vs. direct formatting on the table cell itself. It specifies the components of the conditional formatting in the table style applied to this cell, so that the table's conditional formatting can be applied after the document is displayed without having the table style properties override the style hierarchy. *end note*]

If this element is omitted, then its value shall be assumed to be zero for all entries in the bit mask.

[*Example*: Consider a table cell in the top right corner of a table with a table style applied and where the table is formatted as left to right. This table cell would need to specify the following WordprocessingML to express that fact:

```
<w:tc>
  <w:tcPr>
    <w:cnfStyle w:firstRow="true" w:lastColumn="true"
w:firstRowLastColumn="true" />
    …
  </w:tcPr>
  …
</w:tc>
```

This table cell specifies that it has the conditional properties from the table style for the first column, first row, and the top right corner of the parent table by setting the appropriate attributes. *end example*]

Attributes	Description
evenHBand (Even Numbered Horizontal Band)	Specifies that the object has inherited the conditional properties applied to the even numbered horizontal bands of the parent object.

Attributes	Description

	[*Example*: Consider a paragraph in the second row of a table with a table style applied, and where the band width is one row. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:evenHBand="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the even numbered horizontal bands of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

evenVBand (Even Numbered Vertical Band)	Specifies that the object has inherited the conditional properties applied to the even numbered vertical bands of the parent object.
	
	[*Example*: Consider a paragraph in the second column of a table with a table style applied, and where the band width is one column. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:evenVBand="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the even numbered vertical bands of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstColumn (First Column)	Specifies that the object has inherited the conditional properties applied to the first column of the parent object.
	
	[*Example*: Consider a paragraph in the first column of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:firstColumn="true" />`
	`    …`
	`  </w:pPr>`

Attributes	Description

	`    …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the first column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstRow (First Row)	Specifies that the object has inherited the conditional properties applied to the first row of the parent object.
	
	[*Example*: Consider a paragraph in the top row of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:firstRow="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the first row of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstRowFirstColumn (First Row and First Column)	Specifies that the object has inherited the conditional properties applied to the cell that is in the first row and first column of the parent object.
	
	[*Example*: Consider a paragraph in the first row and first column of a table. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:firstRow="true" w:firstColumn="true"`
	`w:firstRowFirstColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the cell in the first row and first column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

firstRowLastColumn (First Row and	Specifies that the object has inherited the conditional properties applied to the cell that is in the first row and last column of the parent object.

Attributes	Description

Last Column)	[*Example*: Consider a paragraph in the first row and last column of a table. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:firstRow="true" w:lastColumn="true"`
	`w:firstRowLastColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the cell in the first row and last column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastColumn (Last Column)	Specifies that the object has inherited the conditional properties applied to the last column of the parent object.
	
	[*Example*: Consider a paragraph in the last column of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:lastColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the last column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastRow (Last Row)	Specifies that the object has inherited the conditional properties applied to the last row of the parent object.
	
	[*Example*: Consider a paragraph in the bottom row of a table with a table style applied. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:lastRow="true" />`
	`    …`
	`  </w:pPr>`

Attributes	Description

	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the last row of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastRowFirstColumn (Last Row and First Column)	Specifies that the object has inherited the conditional properties applied to the cell that is in the last row and first column of the parent object.
	
	[*Example*: Consider a paragraph in the last row and first column of a table. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:lastRow="true" w:firstColumn="true"`
	`w:lastRowFirstColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the cell in the last row and first column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

lastRowLastColumn (Last Row and Last Column)	Specifies that the object has inherited the conditional properties applied to the cell that is in the last row and last column of the parent object.
	
	[*Example*: Consider a paragraph in the last row and last column of a table. This paragraph would need to specify the following WordprocessingML:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:lastRow="true" w:lastColumn="true"`
	`w:lastRowLastColumn="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	
	This paragraph specifies that it has the conditional properties from the table style for the cell in the last row and last column of the parent table. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

oddHBand (Odd	Specifies that the object has inherited the conditional properties applied to the odd

Attributes	Description

Numbered Horizontal Band)	numbered horizontal bands of the parent object.
	[*Example*: Consider a paragraph in the third row of a table with a table style applied, and where the band width is one column. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:oddHBand="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the odd numbered horizontal bands of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

oddVBand (Odd Numbered Vertical Band)	Specifies that the object has inherited the conditional properties applied to the odd numbered vertical bands of the parent object.
	[*Example*: Consider a paragraph in the third column of a table with a table style applied, and where the band width is one column. This paragraph would need to specify the following WordprocessingML:
	`<w:p>`
	`  <w:pPr>`
	`    <w:cnfStyle w:oddVBand="true" />`
	`    …`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph specifies that it has the conditional properties from the table style for the odd numbered vertical bands of the parent table. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Cnf) is located in §A.1. *end note*]

17.4.9 divId (Associated HTML div ID)

This element specifies the HTML div information which is associated with the current table row. This information, stored in the Web Settings part, is used to associate one or more table rows with a particular HTML div element. [*Note*: This property is used when saving an HTML document into the WordprocessingML format in order to prevent a loss of all HTML div information, so that the document can later be saved back into HTML format and have the stored information replaced, since the HTML div can store formatting properties on arbitrary regions. *end note*]

In order to determine the associated HTML div properties, the value of the val attribute on this element is used to look up an associated div element (§17.15.2.8) whose id attribute matches this value.

If this table row does not have a divId element present, then this table row shall not have any associated HTML div information. If this element is present, but the val attribute specifies an id value which does not have an associated div element, then this element is ignored.

[*Example*: Consider an HTML document defined as follows:

```
<html>
  <body>
    <div style="…">
      <table>
        <tr>
          <td>R1C1</td>
           …
        </tr>
      </table>
      <p>
        …
      </p>
    </div>
    …
  </body>
</html>
```

This HTML document specifies a div spanning the table and the first paragraph. If this document is saved into WordprocessingML, then both the rows of the table and the paragraph must have a divId which points at the same div information in the web settings part:

```
<w:trPr>
  …
  <w:divId w:val="1102603671"/>
</w:trPr>
```

The val attribute then points at a div element which stores the associated div properties:

```
<w:divs>
  <w:div w:id="1102603671">
    …
  </w:div>
</w:divs>
```

This specifies that this table's rows are part of a single HTML div. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.4.10 end (Table Cell Trailing Margin Exception)

This element specifies the amount of space which shall be present between the trailing extent of the current cell's text contents and the trailing border of a specific individual table cell within a table. This setting shall override the table cell trailing margin definition specified by the end element contained within the table properties (§17.4.11).

This value is specified in the units applied via its type attribute. Any width value of type pct or auto for this element shall be ignored.

If omitted, then this table cell shall use the trailing cell margins defined in the end element contained within the table properties (§17.4.11).

[*Example*: Consider a two row, two column LTR table in which the first table cell in the second row has a trailing margin which is specified via an exception to be 0.5 inches, the region marked with an arrow in the table below:

R1C1	R1C2

R2C1 ↔	R2C2

The exception on this cell would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcMar>
      <w:end w:w="720" w:type="dxa" />
    </w:tcMar>
  </w:tcPr>
  …
</w:tc>
```

The R2C1 cell in this table has an exception applied to the table cell trailing cell margin setting it to 720 twentieths of a point (0.5 inches). *end example*]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.11 end (Table Cell Trailing Margin Default)

This element specifies the amount of space which shall be present between the trailing extent of the cell contents and the trailing border of all table cells within the parent table (or table row) . This setting can be overridden by the table cell trailing margin definition specified by the end element contained within the table cell's properties (§17.4.10).

This value is specified in the units applied via its type attribute. Any width value of type pct or auto for this element shall be ignored.

If this element is omitted, then it shall inherit the table cell margin from the associated table style. If a trailing margin is never specified in the style hierarchy, this table shall have 115 twentieths of a point (0.08 inches) left cell padding by default (excepting individual cell overrides).

[*Example*: Consider a two by two LTR table in which the default table cell trailing margin is specified to be exactly 0.25 inches, as follows (marked with an arrow in the first table cell below):

R1C1 ↔	R2C1
R2C1	R2C2

This table property is specified using the following WordprocessingML markup:

```
<w:tbl>
  <w:tblPr>
    <w:tblCellMar>
      <w:end w:w="360" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  …
</w:tbl>
```

Every cell in the table has a default trailing cell margin setting it to `360` twentieths of a point. *end example*]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.12 end (Table Cell Trailing Edge Border)

This element specifies the border which shall be displayed on the trailing edge of the current table cell (right for LTR tables, left for RTL tables). The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§17.4.44;§17.4.43;§17.4.45) applied to the cell is non-zero, then the cell border shall always be displayed
- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

If this element is omitted, then the trailing edge of this table cell shall not have a cell border, and its border can use the table's border settings as appropriate.

[*Example*: Consider an LTR table in which the second cell in the first row specifies a trailing dge cell border as follows:

R1C1	R1C2
R2C1	R2C2

This trailing edge cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
      <w:end w:val="double" w:sz="4" w:space="0" w:color="FF0000" />
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The end element specifies a ½ point border of type `double` on the trailing edge of the table cell. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.4.13 end (Table Trailing Edge Border)

This element specifies the border which shall be displayed at the trailing edge of the current table (right for LTR tables, left for RTL tables). The appearance of this table border in the document shall be determined by the following settings:

- The display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

If this element is omitted, then the trailing edge of this table shall have the border specified by the associated table style. If no trailing edge border is specified in the style hierarchy, then this table shall not have a trailing edge border.

[*Example*: Consider an LTR table in which the table properties specify a trailing edge table border, as follows:

R1C1	R1C2
R2C1	R2C2

This trailing edge table border is specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblBorders>
      <w:end w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="D0D0D0" w:themeColor="accent3" w:themeTint="99"/>
    </w:tblBorders>
  </w:tblPr>
  …
</w:tbl>
```

The end element specifies a three point trailing edge table border of type `thinThinThickMediumGap`. *end example*]

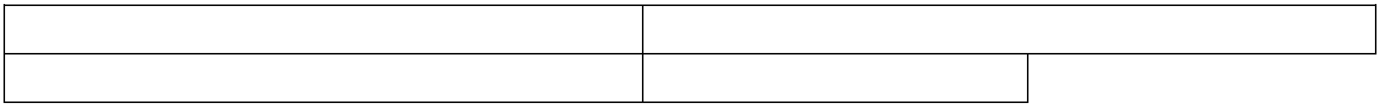
This element's content model is defined by the common border properties definition in §17.3.4.

17.4.14 gridAfter (Grid Columns After Last Cell)

This element specifies the number of grid columns in the parent table's table grid (§17.4.48; §17.4.47) which shall be left after the last cell in the table row.

If this element conflicts with the remaining size of the document grid after all table cells in this row have been added to the grid, then it shall be ignored. If this element is not specified, then its value shall be assumed to be zero grid units.

[*Example*: Consider a table whose second row ends before the first row by one grid unit:



In this table, the second row leaves one grid unit after its cell contents, which is represented using the following WordprocessingML:

```
<w:tr>
  <w:trPr>
    <w:gridAfter w:val="1" />
    …
  </w:trPr>
  …
</w:tr>
```

The gridAfter element specifies that 1 grid unit remains after the row's contents. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

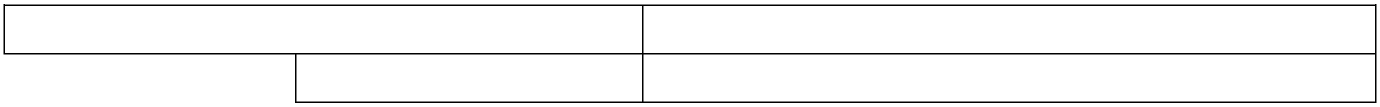
©ISO/IEC 2012 – All rights reserved

17.4.15 gridBefore (Grid Columns Before First Cell)

This element specifies the number of grid columns in the parent table's table grid (§17.4.48; §17.4.47) which must be skipped before the contents of this table row (its table cells) are added to the parent table. [*Note*: This property is used to specify tables whose leading edge (left for left-to-right tables, right for right-to-left tables) does not start at the first grid column (the same shared edge). *end note*]

If this element is omitted, then its value shall be assumed to be zero grid units. If this element's value is larger than the size of the table grid, then the value shall be ignored and the first cell in the row can span the full table grid (i.e. the second cell, if one exists, should start at the last shared edge in the table).

[*Example*: Consider a table whose second row starts after the first grid unit:



In this table, the second row skips one grid unit at the beginning which is represented by the following WordprocessingML:

```
<w:tr>
  <w:trPr>
    <w:gridBefore w:val="1" />
    …
  </w:trPr>
  …
</w:tr>
```

end example]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

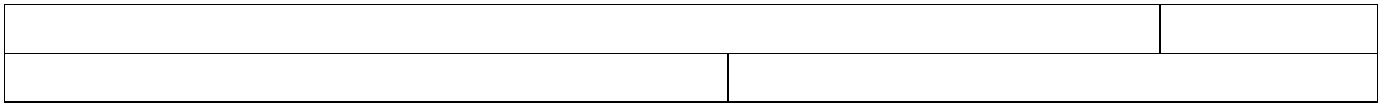
17.4.16 gridCol (Grid Column Definition)

This element specifies the presence and details about a single grid column within a table grid. A *grid column* is a logical column in a table used to specify the presence of a shared vertical edge in the table. When table cells are then added to this table, these shared edges (or grid columns, looking at the column between those shared edges) determine how table cells are placed into the table grid.

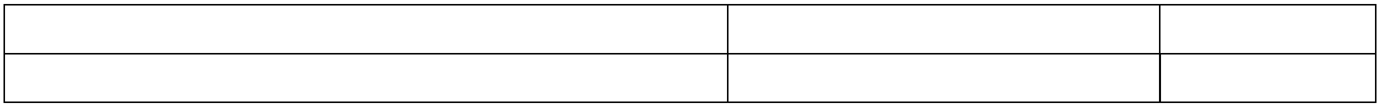
[*Example*: If a table row specifies that it is preceded by two grid columns, then it would start on the third vertical edge in the table including edges which are not shared by all columns. *end example*]

If the table grid does not match the requirements of one or more rows in the table (i.e. it does not define enough grid columns), then the grid can be redefined as needed when the table is processed.

[*Example*: Consider the following, more complex table that has two rows and two columns; as shown, the columns are not aligned:



This table is represented by laying out the cells on a table grid consisting of three table grid columns as follows, each grid column representing a logical vertical column in the table:



The dashed lines represent the virtual vertical continuations of each table grid column, and thus resulting table grid is represented as the following in WordprocessingML:

```
<w:tblGrid>
  <w:gridCol w:w="5051" />
  <w:gridCol w:w="3008" />
  <w:gridCol w:w="1531" />
</w:tblGrid>
```

end example]

Attributes	Description
w (Grid Column	Specifies the width of this grid column.

Attributes	Description

Width)	[*Note*: This value does not solely determine the actual width of the resulting grid column in the document. When the table is displayed in a document, these widths determine the initial width of each grid column, which can then be overridden by:
	• The table layout algorithm (§17.4.52;§17.4.53) applied to the current table row(s)
	• The preferred widths of specific cells which are part of that grid column as the table is displayed (which is an input to the algorithm above)
	end note]
	This value is specified in twentieths of a point.
	If this attribute is omitted, then the last saved width of the grid column is assumed to be zero.
	[*Example*: Consider the following table grid definition:
	`<w:tblGrid>`
	`  <w:gridCol w:w="6888"/>`
	`  <w:gridCol w:w="248"/>`
	`  <w:gridCol w:w="886"/>`
	`  <w:gridCol w:w="1554"/>`
	`</w:tblGrid>`
	This table grid specifies four grid columns, each of which has an initial size of 6888 twentieths of a point, 248 twentieths of a point, 886 twentieths of a point, and 1554 twentieths of a point respectively. *end example*]
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_TblGridCol) is located in §A.1. *end note*]

17.4.17 gridSpan (Grid Columns Spanned by Current Table Cell)

This element specifies the number of grid columns in the parent table's table grid which shall be spanned by the current cell. This property allows cells to have the appearance of being merged, as they span vertical boundaries of other cells in the table.

If this element is omitted, then the number of grid units spanned by this cell shall be assumed to be one. If the number of grid units specified by the val attribute exceeds the size of the table grid, then the table grid shall be augmented as needed to create the number of grid columns required.

[*Example*: Consider the following table that has two rows and two columns where the columns are not aligned:

<table>
<tr><td colspan="2"></td><td></td></tr>
<tr><td></td><td colspan="2"></td></tr>
</table>

This table is represented by laying out the cells on a table grid consisting of three table grid columns, each grid column representing a logical vertical column in the table:

<table>
<tr><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td></tr>
</table>

The first table cell in the first row spans two grid column units. The second cell in the second row also consumes two grid column units (see the grid lines represented using emphasized lines in the example above). This table is represented using the following WordprocessingML:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:tcPr>
        …
        <w:gridSpan w:val="2" />
      </w:tcPr>
      …
    </w:tc>
    …
  </w:tr>
  <w:tr>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        …
        <w:gridSpan w:val="2" />
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
</w:tbl>
```

The gridSpan element indicates the number of columns spanned by each cell with respect to the table grid (in the case of R1C1 and TR2C2, two. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.4.18 header (Header Cell Reference)

This element specifies a reference, using a unique identifier, to a table header cell that is associated with the current table cell. The identifier representing the reference shall be stored on this element's val attribute and is used to reference the unique identifier value of the *id* attribute of a header cell *tc* element of the current table. The contents of the table header cell *tc* element designated by the specific unique identifier in its *id* attribute shall be used as the table header information associated with the table cell that references that specific unique identifier.

If this element is omitted or the value of the header cell identifier cannot be resolved, no header cell shall be associated with the current table cell.

[*Example*: Consider the following 3 x 3 table with four header cells with values A, B, C, and D and four data cells with values x1, x2, y1, and y2:

	A	B
C	x1	x2
D	y1	y2

Each of the data cells is associated with two header cells and can be represented in WordprocessingML as follows:

```
<w:tbl>
  …
  <w:tr>
```

```
      …
      <w:tc >
        …
      </w:tc>
      <w:tc w:id="HeaderA">
        …
        <w:p>
          <w:r>
            <w:t>A</w:t>
          </w:r>
        </w:p>
      </w:tc>
      …
    </w:tr>
    <w:tr>
      …
      <w:tc w:id="HeaderC">
        …
        <w:p>
          <w:r>
            <w:t>C</w:t>
          </w:r>
        </w:p>
      </w:tc>
      <w:tc>
        <w:tcPr>
          …
          <w:headers>
            <w:header w:val="HeaderA" />
            <w:header w:val="HeaderC" />
          </w:headers>
          …
        </w:tcPr>
        <w:p>
          <w:r>
            <w:t>x1</w:t>
          </w:r>
        </w:p>
      </w:tc>
      …
    </w:tr>
  </w:tbl>
```

©ISO/IEC 2012 – All rights reserved

The headers element specifies the list of header cells associated with the table cell that has a value of x1. In this example the table cell with the content value of x1 is associated with headers that have an id of HeaderA and HeaderC. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's styleId.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.4.19 headers (Header Cells Associated With Table Cell)

This element specifies the list of header cells, as specified by children header elements, that provide header information associated with the current table cell. Each header cell shall specify a unique identifier, as specified by the use of the attribute id on the header cell tc element. This element is typically used to gather header information about data and sub header cells.

If this element is omitted or no children header element exists, no header cell shall be associated with the given table cell.

[*Example*: Consider the following 3 x 3 table with four header cells with values A, B, C, and D and four data cells with values x1, x2, y1, and y2:

	A	B
C	x1	x2
D	y1	y2

Each of the data cells is associated with two header cells and can be represented in WordProcessingML as follows:

```
<w:tbl>
  …
  <w:tr>
    …
    <w:tc >
      …
    </w:tc>
    <w:tc w:id="HeaderA">
      …
      <w:p>
        <w:r>
          <w:t>A</w:t>
        </w:r>
      </w:p>
    </w:tc>
    …
  </w:tr>
  <w:tr>
    …
    <w:tc w:id="HeaderC">
      …
      <w:p>
        <w:r>
          <w:t>C</w:t>
        </w:r>
      </w:p>
    </w:tc>
    <w:tc>
      <w:tcPr>
        …
        <w:headers>
          <w:header w:val="HeaderA" />
          <w:header w:val="HeaderC" />
        </w:headers>
        …
      </w:tcPr>
      <w:p>
```

©ISO/IEC 2012 – All rights reserved

```
        <w:r>
          <w:t>x1</w:t>
        </w:r>
      </w:p>
    </w:tc>
    …
  </w:tr>
</w:tbl>
```

The headers element specifies the list of header cells associated with the table cell that has a value of x1. In this example x1 is associated with headers that have an id of HeaderA and HeaderC. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Headers) is located in §A.1. *end note*]

17.4.20 hidden (Hidden Table Row Marker)

This element specifies that the glyph representing the end character of current table row shall not be displayed in the current document.

[*Note*: This setting is used to hide the end of row glyph in order to ensure that the entire table row is hidden and not displayed in the document, as if any part of the row is visible, the row is displayed. *end note*]

[*Note*: Applications can have settings which allow hidden content to be displayed, in which case this content can be visible - this property is not meant to supersede that setting. *end note*]

If this element is omitted, then this table row shall not be hidden in the document.

[*Example*: Consider a table with a table row in which the row is specified to be hidden. That requirement is specified using the following WordprocessingML:

```
<w:tbl>
  …
  <w:tr>
    <w:trPr>
      <w:hidden />
      …
    </w:trPr>
    …
  </w:tr>
</w:tbl>
```

In this example this row is not be displayed nor printed, as the hidden element is specified on in table row's properties. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.4.21 hideMark (Ignore End Of Cell Marker In Row Height Calculation)

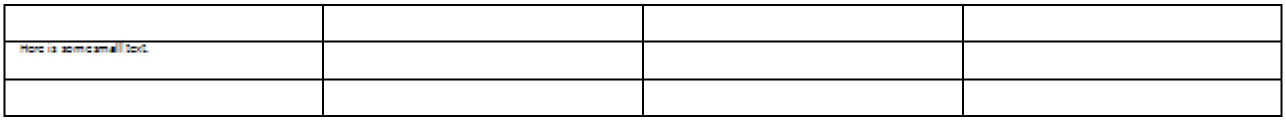
This element specifies whether the end of cell glyph shall influence the height of the given table row in the table. If it is specified, then only printing characters in this cell shall be used to determine the row height.

[*Rationale*: Typically, the height of a table row is determined by the height of all glyphs in all cells in that row, including the non-printing end of cell glyph characters. However, if these characters are not formatted, they are always created with the document default style properties. This means that the height of a table row cannot ever be reduced below the size of the end of cell marker glyph without manually formatting each paragraph in that run.

In a typical document, this behavior is desirable as it prevents table rows from 'disappearing' if they have no content. However, if a table row is being used as a border (for example, by shading its cells or putting an image in them), then this behavior makes it impossible to have a virtual border that is reasonably small without formatting each cell's content directly. This setting specifies that the end of cell glyph shall be ignored for this cell, allowing it to collapse to the height of its contents without formatting each cell's end of cell marker, which would have the side effect of formatting any text ever entered into that cell. *end rationale*]

If this element is omitted, then the end of cell marker shall be included in the determination of the height of this row.

[*Example*: Consider the following WordprocessingML table:



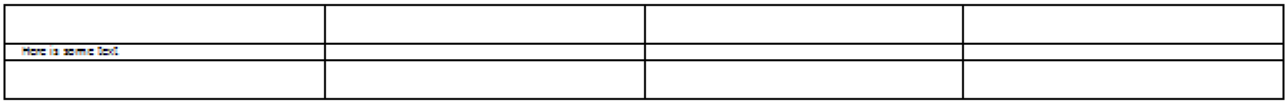
Here is some small text.			

Notice that the only printing content in this table row is displayed using 5 point font, yet the row height is influenced by the end of cell markers in the empty cells.

If each cell in the second row in this table was set to exclude the table cell from this calculation, using the following WordprocessingML:

```
<w:tcPr>
  <w:hideMark/>
</w:tcPr>
```

The resulting table must exclude the cell markers from the row height calculation:



Here is some text.			

The hideMark element specified that each cell marker was excluded, resulting in the row height being defined by the actual run contents. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.4.22 insideH (Table Inside Horizontal Edges Border)

This element specifies the border which shall be displayed on all horizontal table cell borders which are not on an outmost edge of the parent table (all horizontal borders which are not the topmost or bottommost border). The appearance of this table cell border in the document shall be determined by the following settings:

- The display of the border on interior edges is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

If this element is omitted, then the inside horizontal borders of this table shall have the border specified by the associated table style. If no inside horizontal edge border is specified in the style hierarchy, then this table shall not have an inside horizontal edge border.

[*Example*: Consider a table in which the table specifies a border on all interior horizontal and vertical edges, as follows:

R1C1	R1C2
R2C1	R2C2

This interior horizontal cell border is specified using the following WordprocessingML:

```
<w:tblPr>
  <w:tblBorders>
    <w:insideH w:val="doubleWave" w:sz="6" w:space="0" w:color="92D050"/>
    <w:insideV w:val="doubleWave" w:sz="6" w:space="0" w:color="92D050"/>
  </w:tblBorders>
  …
</w:tblPr>
```

The insideH element specifies a ¾ point border of type `doubleWave`. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.4.23 insideH (Table Cell Inside Horizontal Edges Border)

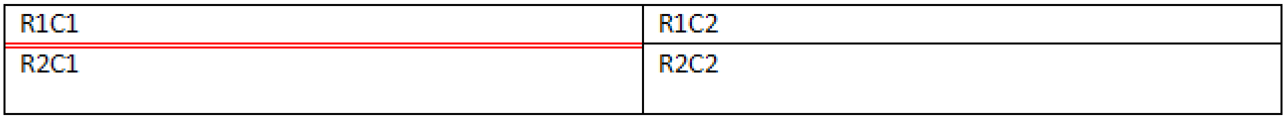
This element specifies the border which shall be displayed on all interior horizontal edges of the current group of table cells. [*Note*: Although individual table cells have no concept of an internal edge, which would render this property useless in most cases, it is used to determine the cell borders to apply to a specific group of cells as part of table conditional formatting in a table style, for example, the inside horizontal edges on the set of cells in the first column. *end note*]

The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§17.4.44;§17.4.43;§17.4.45) applied to the cell is non-zero, then the cell border shall always be displayed
- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

If this element is omitted, then the specified conditional formatting on the table shall not change the current set of internal edge borders on its set of table cells (i.e. their current setting shall remain unchanged).

[*Example*: Consider a table in which the conditional formatting on the first column specified in the associated table style specifies a double line red cell border for all internal horizontal lines as follows:



R1C1	R1C2
R2C1	R2C2

This inner horizontal edge cell border is specified using the following WordprocessingML:

```
<w:tblStylePr w:type="firstCol">
  <w:tcPr>
    <w:tcBorders>
      <w:insideH w:val="double" w:sz="4" w:space="0" w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
</w:tblStylePr>
```

The insideH element specifies a ¼ point border of type `double`. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

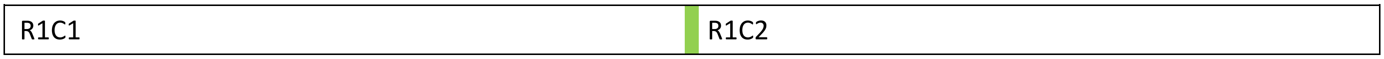
17.4.24 insideV (Table Inside Vertical Edges Border)

This element specifies the border which shall be displayed on all vertical table cell borders which are not on an outmost edge of the parent table (all horizontal borders which are not the leftmost or rightmost border). The appearance of this table cell border in the document shall be determined by the following settings:

- The display of the border on interior edges is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

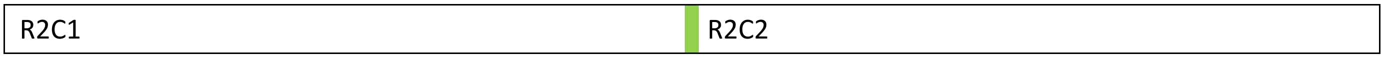
If this element is omitted, then the inside vertical borders of this table shall have the border specified by the associated table style. If no inside vertical edge border is specified in the style hierarchy, then those cells in this table shall not have an inside vertical edge border.

[*Example*: Consider a table in which the table specifies a border on all interior horizontal and vertical edges, as follows:



R1C1	R1C2

©ISO/IEC 2012 – All rights reserved



R2C1	R2C2

This interior horizontal cell border is specified using the following WordprocessingML:

```
<w:tblPr>
  <w:tblBorders>
    <w:insideH w:val="doubleWave" w:sz="6" w:space="0" w:color="92D050"/>
    <w:insideV w:val="doubleWave" w:sz="6" w:space="0" w:color="92D050"/>
  </w:tblBorders>
  …
</w:tblPr>
```

The insideV element specifies a ¾ point border of type doubleWave on all interior vertical edges. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.4.25 insideV (Table Cell Inside Vertical Edges Border)

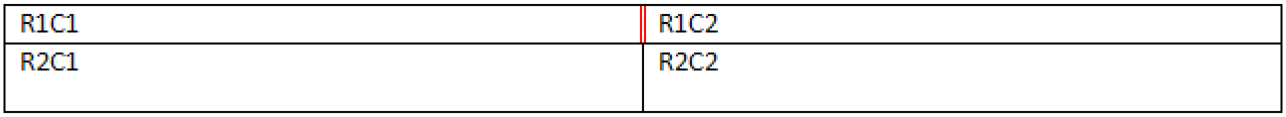
This element specifies the border which shall be displayed on all interior vertical edges of the current group of table cells. [*Note*: Although individual table cells have no concept of an internal edge, which would render this property useless in most cases, it is used to determine the cell borders to apply to a specific group of cells as part of table conditional formatting in a table style, for example, the inside vertical edges on the set of cells in the header row. *end note*]

The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§17.4.44;§17.4.43;§17.4.45) applied to the cell is non-zero, then the cell border shall always be displayed
- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

If this element is omitted, then the specified conditional formatting on the table shall not change the current set of internal edge borders on its set of table cells (i.e. their current setting shall remain unchanged).

[*Example*: Consider a table in which the conditional formatting on the header row in the associated table style specifies a double line red cell border for all internal vertical lines as follows:



R1C1	R1C2
R2C1	R2C2

This inner vertical edge cell border is specified using the following WordprocessingML:

```
<w:tblStylePr w:type="firstRow">
  <w:tcPr>
    <w:tcBorders>
      <w:insideV w:val="double" w:sz="4" w:space="0" w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
</w:tblStylePr>
```

The insideV element specifies a ¼ point border of type `double`. *end example*]

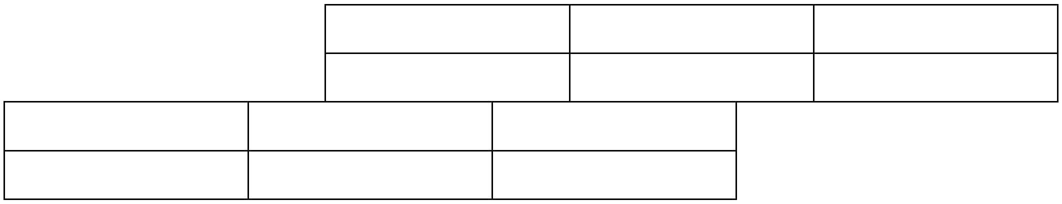
This element's content model is defined by the common border properties definition in §17.3.4.

17.4.26 jc (Table Alignment Exception)

This element specifies the alignment of the set of rows which are part of the current table properties exception list with respect to the text margins in the current section. When a table is placed in a WordprocessingML document that does not have the same width as the margins, this property is used to determine how the table is positioned with respect to those margins. The interpretation of property is reversed if the parent table is right to left using the bidiVisual element (§17.4.1).

If this property is omitted on a table, then the justification shall be determined by the default set of table properties on the parent table.

[*Example*: Consider the following WordprocessingML table, centered on the text margins with a subset of its rows justified to the left margin by a table property exception:



That exception would be specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:jc w:val="start"/>
</w:tblPrEx>
```

The jc element specifies that the rows which are part of the table properties exception table must be `left` aligned with respect to the text margins. *end example*]

Attributes	Description
val (Alignment Type)	Specifies the justification which should be applied to the parent table. [*Example*: Consider the following WordprocessingML fragment for a table in a document:

Attributes	Description

	`<w:tblPr>`
	`  <w:jc w:val="center" />`
	`</w:tblPr>`
	This table is now centered on the page. *end example*]
	The possible values for this attribute are defined by the ST_JcTable simple type (§17.18.45).

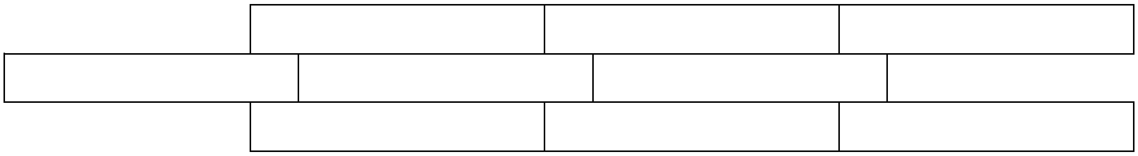
[*Note*: The W3C XML Schema definition of this element's content model (CT_JcTable) is located in §A.1. *end note*]

17.4.27 jc (Table Row Alignment)

This element specifies the alignment of a single row in the parent table with respect to the text margins in the current section. When a table is placed in a WordprocessingML document that does not have the same width as the margins, this property is used to determine how a specific row in that table is positioned with respect to those margins. The interpretation of property is reversed if the parent table is right to left using the bidiVisual element (§17.4.1).

If this property is omitted on a table, then the justification shall be determined by the default set of table properties on the parent table.

[*Example*: Consider the following WordprocessingML table, centered on the text margins with its second rows justified to the left margin by a table row level justification:



That row level setting would be specified using the following WordprocessingML:

```
<w:trPr>
  <w:jc w:val="start"/>
</w:trPr>
```

The jc element specifies that the rows which are part of the table properties exception table must be `left` aligned with respect to the text margins. *end example*]

Attributes	Description
val (Alignment	Specifies the justification which should be applied to the parent table.

Attributes	Description

Type)	[*Example*: Consider the following WordprocessingML fragment for a table in a document:
	`<w:tblPr>`
	`  <w:jc w:val="center" />`
	`</w:tblPr>`
	This table is now centered on the page. *end example*]
	The possible values for this attribute are defined by the ST_JcTable simple type (§17.18.45).

[*Note*: The W3C XML Schema definition of this element's content model (CT_JcTable) is located in §A.1. *end note*]

17.4.28 jc (Table Alignment)

This element specifies the alignment of the current table with respect to the text margins in the current section. When a table is placed in a WordprocessingML document that does not have the same width as the margins, this property is used to determine how the table is positioned with respect to those margins. The interpretation of property is reversed if the parent table is right to left using the bidiVisual element (§17.4.1).

If this property is omitted on a table, then the justification shall be determined by the associated table style. If this property is not specified in the style hierarchy, then the table shall be left justified with zero indentation from the leading margin (the left margin in a left-to-right table or the right margin in a right-to-left table).

[*Example*: Consider the following WordprocessingML table, justified to the left margin by default:

R1C1	R1C2	R1C3
R2C1	R2C2	R2C3

This table does not fill the entire width of the text margins. If the table should be right justified to the margin, as follows:

R1C1	R1C2	R1C3
R2C1	R2C2	R2C3

That requirement would be specified using the following WordprocessingML:

```
<w:tblPr>
  <w:jc w:val="end"/>
</w:tblPr>
```

The jc element specifies that the table must be `right` aligned with respect to the text margins. *end example*]

Attributes	Description

val (Alignment Type)	Specifies the justification which should be applied to the parent table.
	[*Example*: Consider the following WordprocessingML fragment for a table in a document:
	`<w:tblPr>`
	`  <w:jc w:val="center" />`
	`</w:tblPr>`
	This table is now centered on the page. *end example*]
	The possible values for this attribute are defined by the ST_JcTable simple type (§17.18.45).

[*Note*: The W3C XML Schema definition of this element's content model (CT_JcTable) is located in §A.1. *end note*]

17.4.29 noWrap (Don't Wrap Cell Content)

This element specifies how this table cell shall be laid out when the parent table is displayed in a document. This setting only affects the behavior of the cell when the tblLayout for this row (§17.4.52; §17.4.53) is set to use the `auto` algorithm.

This setting shall be interpreted in the context of the tcW element (§17.4.71) as follows:

- If the table cell width has a type attribute value of `fixed`, then this element specifies that that this table cell shall never be smaller than that fixed value when other cells on the line are not at their absolute minimum width.
- If the table cell width has a type attribute value of `pct` or `auto`, then this element specifies that when running the auto fit algorithm, the contents of that this table cell shall be treated as though they have no breaking characters (the contents should be treated as a single contiguous non-breaking string)

If this element is omitted, then cell content shall be allowed to wrap (the cell can be shrunk as needed if it is a fixed preferred width value, and the contents shall be treated as having breaking characters if it is a percentage or automatic width value).

[*Example*: Consider the following three row by three column WordprocessingML table:

In this table, each cell has a `fixed` preferred width of 2.38 inches (3427 twentieths of a point), and the tblLayout for this row (§17.4.52; §17.4.53) is set to use the `auto` algorithm. If a long non breaking string is added to the middle row, as follows, the two cells are adjusted to override their preferences and accommodate the string:

	sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss	

However, if the first table cell has the noWrap element present as follows:

```
<w:tcPr>
  <w:noWrap/>
</w:tcPr>
```

The noWrap element specifies that because it is a `fixed` width cell, that cell shall not be collapsed beyond its original size until all other cells are at their minimum size, so in this example the cell maintains its width:

	sssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssssss	

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.4.30 shd (Table Shading Exception)

This element specifies the shading which shall be applied to all cells in the current row as part of a set of table-level property exceptions. Similarly to paragraph shading, this shading shall be applied to the contents of the tab up to the table borders, regardless of the presence of text - unlike cell shading, table shading shall include any cell padding. This property shall be superseded by any cell-level shading on any cell in this row (§17.4.32).

This shading consists of three components:

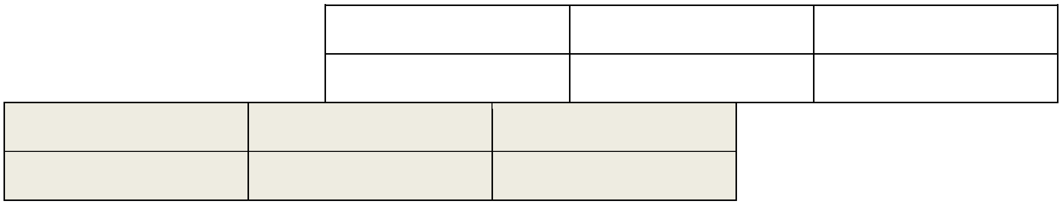
- Background Color
- (optional) Pattern
- (optional) Pattern Color

The resulting shading is applied by setting the background color behind the paragraph, then applying the pattern color using the mask supplied by the pattern over that background.

©ISO/IEC 2012 – All rights reserved

If this element is omitted, then the cell shading shall be determined by the table-level cell shading settings (§17.4.31) for the current table.

[*Example*: Consider a table in which the final two rows have a set of table-level property exceptions giving them theme color shading in the background2 theme color, as follows:



This table-level shading exception would be specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:jc w:val="start" />
  <w:shd w:val="clear" w:color="auto" w:fill="EEECE1" w:themeFill="background2"
/>
</w:tblPrEx>
```

The shd element specifies cell shading with a clear pattern using a background theme color of background2. *end example*]

This element's content model is defined by the common shading properties definition in §17.3.5.

17.4.31 shd (Table Shading)

This element specifies the shading which shall be applied to the extents the current table. Similarly to paragraph shading, this shading shall be applied to the contents of the tab up to the table borders, regardless of the presence of text - unlike cell shading, table shading shall include any cell padding. This property shall be superseded by any cell-level shading via any table-level property exceptions (§17.4.30); or on any cell in this row (§17.4.32).

This shading consists of three components:

- Background Color
- (optional) Pattern
- (optional) Pattern Color

The resulting shading is applied by setting the background color behind the paragraph, then applying the pattern color using the mask supplied by the pattern over that background.

If this element is omitted, then the cells within this table shall have the shading specified by the associated table style. If no cell shading is specified in the style hierarchy, then the cells in this table shall not have any cell shading (i.e. they shall be transparent).

[*Example*: Consider a table in which the first cell in the first row has cell-level red shading, as follows:

R1C1	R1C2
R2C1	R2C2

This table level cell shading would be specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:shd w:val="clear" w:color="auto" w:fill="FF0000"/>
    …
  </w:tblPr>
  …
</w:tbl>
```

The shd element specifies cell shading with a `clear` pattern using a background color of `FF0000` (red). *end example*]

This element's content model is defined by the common shading properties definition in §17.3.5.

17.4.32 shd (Table Cell Shading)

This element specifies the shading which shall be applied to the extents of the current table cell. Similarly to paragraph shading, this shading shall be applied to the contents of the cell up to the cell borders, regardless of the presence of text.

This shading consists of three components:

- Background Color
- (optional) Pattern
- (optional) Pattern Color

The resulting shading is applied by setting the background color behind the paragraph, then applying the pattern color using the mask supplied by the pattern over that background.

If this element is omitted, then the cell shading shall be determined by the table-level or table-level exception cell shading settings (§17.4.30;§17.4.31) for the current table.

[*Example*: Consider a table in which the first cell in the first row has cell-level red shading, as follows:

R1C1	R1C2
R2C1	R2C2

This cell shading would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:shd w:val="clear" w:color="auto" w:fill="FF0000" />
  </w:tcPr>
</w:tc>
```

The shd element specifies cell shading with a `clear` pattern using a background color of `FF0000` (red). *end example*]

This element's content model is defined by the common shading properties definition in §17.3.5.

17.4.33 start (Table Cell Leading Edge Border)

This element specifies the border which shall be displayed on the leading edge of the current table cell (left for LTR tables, right for RTL tables). The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§17.4.44;§17.4.43;§17.4.45) applied to the cell is non-zero, then the cell border shall always be displayed
- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

If this element is omitted, then the leading edge of this table cell shall not have a cell border, and its border can use the table's border settings as appropriate.

[*Example*: Consider an LTR table in which the second cell in the first row specifies a leading edge cell border:

R1C1	R1C2
R2C1	R2C2

This leading edge cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
      <w:start w:val="double" w:sz="4" w:space="0" w:color="FF0000" />
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The start element specifies a ½ point border of type `double` on the leading edge of the table cell. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.4.34 start (Table Cell Leading Margin Default)

This element specifies the amount of space which shall be left between the leading edge of the cell contents and the leading edge of all table cells within the parent table (or table row). This setting can be overridden by the table cell leading margin definition specified by the start element contained within the table cell's properties (§17.4.35).

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If this element is omitted, then it shall inherit the table cell margin from the associated table style. If a leading margin is never specified in the style hierarchy, this table shall have 115 twentieths of a point (0.08 inches) left cell padding by default (excepting individual cell overrides).

[*Example*: Consider a two by two LTR table in which the default table cell leading margin is specified to be exactly 0.25 inches, as follows (marked with an arrow in the first table cell below):

↔ R1C1	R2C1
R2C1	R2C2

This table property is specified using the following WordprocessingML markup:

```
<w:tbl>
  <w:tblPr>
    <w:tblCellMar>
      <w:start w:w="360" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  …
</w:tbl>
```

Every cell in the table has a default leading cell margin setting it to `360` twentieths of a point. *end example*]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.35 start (Table Cell Leading Margin Exception)

This element specifies the amount of space which shall be left between the leading extent of the current cell contents and the leading edge border of a specific individual table cell within a table. This setting shall override the table cell leading margin definition specified by the start element contained within the table properties (§17.4.34).

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If omitted, then this table cell shall use the leading cell margins defined in the start element contained within the table properties (§17.4.34).

[*Example*: Consider a two row, two column LTR table in which the first table cell in the second row has a leading margin which is specified via an exception to be 0.5 inches, causing the text to be position 0.5" inside the cell, as follows:

R1C1	R1C2
R2C1	R2C2

The exception on this cell would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcMar>
      <w:start w:w="720" w:type="dxa" />
    </w:tcMar>
  </w:tcPr>
</w:tc>
```

The R2C1 cell in this table has an exception applied to the table cell leading margin setting it to 720 twentieths of a point (0.5 inches). *end example*]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.36 start (Table Leading Edge Border)

This element specifies the border which shall be displayed at the leading edge of the current table (left for LTR tables, right for RTL tables). The appearance of this table border in the document shall be determined by the following settings:

- The display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

If this element is omitted, then the leading edge of this table shall have the border specified by the associated table style. If no leading edge border is specified in the style hierarchy, then this table shall not have a left border.

[*Example*: Consider an LTR table in which the table properties specify a leading edge table border, as follows:

R1C1	R1C2
R2C1	R2C2

This leading edge table border is specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblBorders>
      <w:start w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="D0D0D0" w:themeColor="accent3" w:themeTint="99"/>
    </w:tblBorders>
  </w:tblPr>
  …
</w:tbl>
```

The start element specifies a three point leading edge table border of type `thinThinThickMediumGap`. *end example*]

This element’s content model is defined by the common border properties definition in §17.3.4.

17.4.37 tbl (Table)

This element specifies the contents of a table present in the document. A *table* is a set of paragraphs (and other block-level content) arranged in *rows* and *columns*. Tables in WordprocessingML are defined via the tbl element, which is analogous to the HTML `table` tag.

When two tbl elements having the same style (§17.4.62) are present within the document content, without any intervening p elements, the corresponding tables shall be treated as a single table.

[*Example*: Consider an empty one-cell table (i.e.; a table with one row, one column) and 1 point borders on all sides:

This table is represented by the following WordprocessingML:

```
  <w:tbl>
    <w:tblPr>
      <w:tblW w:w="5000" w:type="pct"/>
      <w:tblBorders>
        <w:top w:val="single" w:sz="4" w:space="0" w:color="auto"/>
        <w:start w:val="single" w:sz="4" w:space="0" w:color="auto"/>
        <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto"/>
        <w:end w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      </w:tblBorders>
    </w:tblPr>
    <w:tblGrid>
      <w:gridCol w:w="10296"/>
    </w:tblGrid>
    <w:tr>
      <w:tc>
        <w:tcPr>
          <w:tcW w:w="0" w:type="auto"/>
        </w:tcPr>
        <w:p/>
      </w:tc>
    </w:tr>
  </w:tbl>
```

This table specifies table-wide properties of 100% of page width using the tblW element (§17.4.63); a the set of table borders using the tblBorders element (§17.4.38); the table grid which defines a set of shared vertical edges within the table using the tblGrid element (§17.4.48); and a single table row using the tr element (§17.4.78). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Tbl) is located in §A.1. *end note*]

17.4.38 tblBorders (Table Borders)

This element specifies the set of borders for the edges of the current table, using the six border types defined by its child elements.

If the cell spacing for any row is non-zero as specified using the tblCellSpacing element (§17.4.44; §17.4.43; §17.4.45), then there is no border conflict and the table border (or table-level exception border, if one is specified) shall be displayed.

If the cell spacing is zero, then there is a conflict [*Example*: Between the left border of all cells in the first column and the left border of the table. *end example*], which shall be resolved as follows:

- If there is a cell border, then the cell border shall be displayed
- If there is no cell border but there is a table-level exception border on this table row, then the table-level exception border shall be displayed

- If there is no cell or table-level exception border, then the table border shall be displayed

If this element is omitted, then this table shall have the borders specified by the associated table style. If no borders are specified in the style hierarchy, then this table shall not have any table borders.

[*Example*: Consider a table with no associated table style, which defines a set of table borders via direct formatting as follows:

These table borders are specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:w="0" w:type="auto"/>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:start w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:end w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:insideH w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
      <w:insideV w:val="single" w:sz="4" w:space="0" w:color="000000"
w:themeColor="text1"/>
    </w:tblBorders>
    …
  </w:tblPr>
  …
</w:tbl>
```

The tblBorders element specifies the set of table borders applied to the current table. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblBorders) is located in §A.1. *end note*]

©ISO/IEC 2012 – All rights reserved

17.4.39 tblBorders (Table Borders Exceptions)

This element specifies the set of borders for the edges of the parent table row via a set of table-level property exceptions, using the six border types defined by its child elements.

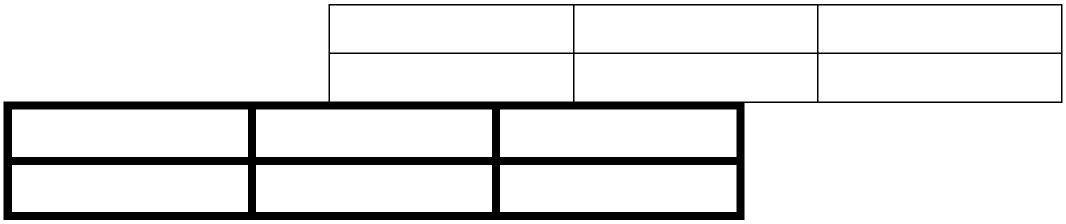
If the cell spacing for any row is non-zero as specified using the tblCellSpacing element (§17.4.44; §17.4.43; §17.4.45), then there is no border conflict and the table-level exception border shall be displayed.

If the cell spacing is zero, then there is a conflict [*Example*: Between the left border of all cells in the first column and the left border of the table-level exceptions. *end example*], which shall be resolved as follows:

- If there is a cell border, then the cell border shall be displayed
- If there is no cell border, then the table-level exception border shall be displayed

If this element is omitted, then this table shall have the borders specified by the associated table level borders (§17.4.38).

[*Example*: Consider a table in which the final two rows have a set of table-level property exceptions giving them a thicker set of table borders, as follows:



These table borders are specified via a set of table-level property exceptions using the following WordprocessingML:

```
<w:tr>
  <w:tblPrEx>
    <w:tblBorders>
      <w:top w:val="single" w:sz="24" w:space="0" w:color="000000" 
w:themeColor="text1"/>
      <w:start w:val="single" w:sz="24" w:space="0" w:color="000000" 
w:themeColor="text1"/>
      <w:bottom w:val="single" w:sz="24" w:space="0" w:color="000000" 
w:themeColor="text1"/>
      <w:end w:val="single" w:sz="24" w:space="0" w:color="000000" 
w:themeColor="text1"/>
      <w:insideH w:val="single" w:sz="24" w:space="0" w:color="000000" 
w:themeColor="text1"/>
      <w:insideV w:val="single" w:sz="24" w:space="0" w:color="000000" 
w:themeColor="text1"/>
    </w:tblBorders>
  </w:tblPrEx>
</w:tr>
```

The tblBorders element specifies the set of table borders applied to the final two rows in this table as part of the table-level property exceptions. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblBorders) is located in §A.1. *end note*]

17.4.40 tblCaption (Table Caption)

This element specifies the caption for the table.

[*Example*: Consider a table which specifies a caption. This object might contain the following XML markup:

```
<w:tbl>
    <w:tblPr>
        <w:tblCaption w:val="Here is the caption of the table" />
        …
    </w:tblPr>
</w:tbl>
```

end example]

Attributes	Description
val (String Value)	Specifies that its contents contain a string. The contents of this string are interpreted based on the context of the parent XML element.

Attributes	Description

	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

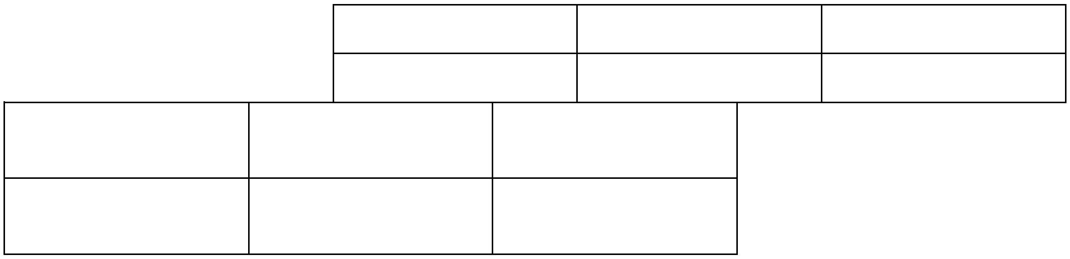
[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.4.41 tblCellMar (Table Cell Margin Exceptions)

This element specifies a set of cell margins for all cells in the parent table row via a set of table-level property exceptions. These settings can be overridden by the table cell margin definition specified by the tcMar element contained within the table cell's properties (§17.4.41).

If this element is omitted, then it shall inherit the table cell margins from the table-level cell margins (§17.4.42).

[*Example*: Consider a table whose final two rows are defined to have default cell margins of 0.1 inches for all sides via a table-level property exception, as follows:



This set of table cell margin exceptions is specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:tblCellMar>
    <w:top w:w="144" w:type="dxa"/>
    <w:start w:w="144" w:type="dxa"/>
    <w:bottom w:w="144" w:type="dxa"/>
    <w:end w:w="144" w:type="dxa"/>
  </w:tblCellMar>
  …
</w:tblPrEx>
```

The tblCellMar element as a child of tblPrEx specifies the set of default cell margins for all cells in final two rows in current table, in this case, 144 twentieths of a point on all sides. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblCellMar) is located in §A.1. *end note*]

17.4.42 tblCellMar (Table Cell Margin Defaults)

This element specifies the default cell margin settings for all cells in the current table. These setting can be overridden by the table cell margin definition specified by the tcMar element contained within the table cell's properties (§17.4.68) or by a set of table-level property exceptions (§17.4.41).

If this element is omitted, then it shall inherit the table cell margins from the associated table style. If table margins are never specified in the style hierarchy, then each margin shall use its default margin size (see child element definitions).

[*Example*: Consider a table defined to have default cells margins of 0.1 inches for all sides, as follows:

R1C1	R1C2
R2C1	R2C2

This set of default table cell margins would be specified using the following WordprocessingML:

```
<w:tblPr>
  <w:tblCellMar>
    <w:top w:w="144" w:type="dxa"/>
    <w:start w:w="144" w:type="dxa"/>
    <w:bottom w:w="144" w:type="dxa"/>
    <w:end w:w="144" w:type="dxa"/>
  </w:tblCellMar>
  …
</w:tblPr>
```

The tblCellMar element as a child of tblPr specifies the set of default cell margins for all cells in the current table, in this case, 144 twentieths of a point. *end example*]

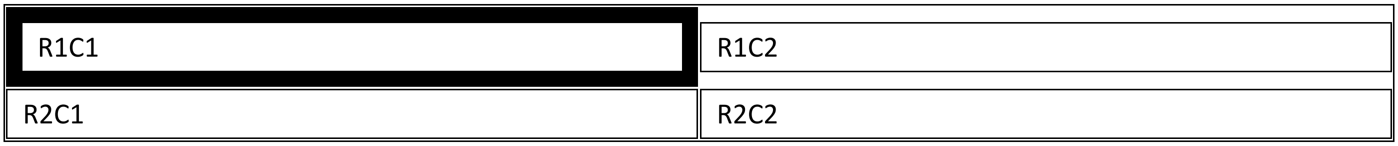
[*Note*: The W3C XML Schema definition of this element's content model (CT_TblCellMar) is located in §A.1. *end note*]

17.4.43 tblCellSpacing (Table Row Cell Spacing)

This element specifies the default table cell spacing (the spacing between adjacent cells and the edges of the table) for all cells in the parent row. If specified, this element specifies the minimum amount of space which shall be left between all cells in the table including the width of the table borders in the calculation. It is important to note that row-level cell spacing shall be added inside of the text margins, which shall be aligned with the innermost starting edge of the text extents in a cell without row-level indentation or cell spacing. Row-level cell spacing shall not increase the width of the overall table.

This value is specified in the units applied via its type attribute. Any width value of type pct or auto for this element shall be ignored.

[*Example*: Consider a table whose first cell has a six point wide table border, and a table cell spacing value of 0.01 inches. The resulting table would have 0.01 inches of space between each table cell regardless of the width of the cell border, as follows (notice that no border is covered by any other border):

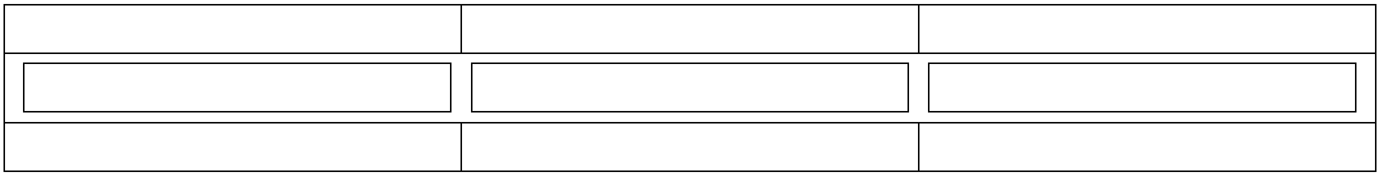


R1C1	R1C2
R2C1	R2C2

end example]

If this element is omitted, then the cells in this row shall inherit the cell spacing from the associated table level properties.

[*Example*: Consider a table where the second row has a cell spacing of 0.1 inches for all sides specified via the table row properties as follows:



This table row cell spacing is specified using the following WordprocessingML:

```
<w:trPr>
  <w:tblCellSpacing w:w="144" w:type="dxa"/>
  …
</w:trPr>
```

The tblCellSpacing element as a child of trPr specifies the default cell spacing between all cells in the current row, in this case 144 twentieths of a point. *end example*]

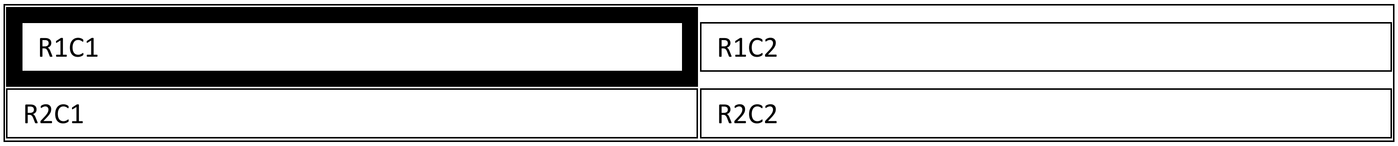
This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.44 tblCellSpacing (Table Cell Spacing Exception)

This element specifies a table cell spacing exception for all cells in the parent table row as part of a set of table-level property exceptions. If specified, this element specifies the minimum amount of space which shall be left between all cells in the parent row after including the width of the table borders in the calculation. This setting shall be superseded by the row cell spacing value (§17.4.43). It is important to note that table-level cell spacing shall be added outside of the text margins, which shall be aligned with the innermost starting edge of the text extents in a table cell.

This value is specified in the units applied via its type attribute. Any width value of type pct or auto for this element shall be ignored.

[*Example*: Consider a table whose first cell has a six point wide table border, and a table cell spacing value of 0.01 inches. The resulting table would have 0.01 inches of space between each table cell regardless of the width of the cell border, as follows (notice that no border is covered by any other border):

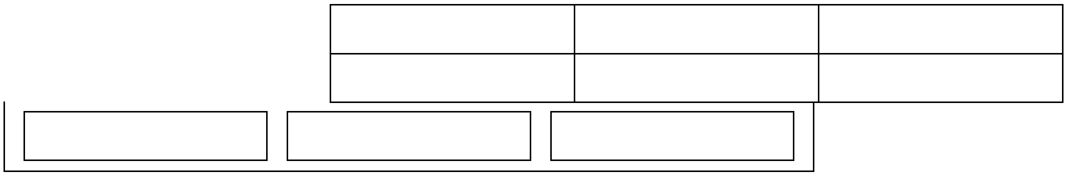


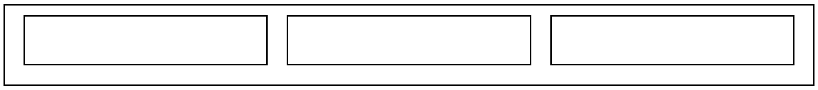
R1C1	R1C2
R2C1	R2C2

end example]

If this element is omitted, then the row shall inherit the table cell spacing from the table-level cell spacing setting (§17.4.42), excepting the case of a row level override.

[*Example*: Consider a table whose final two rows are defined to have cell spacing of 0.1 inches for all sides via a table-level property exception, as follows:





This table cell spacing exception is specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:tblCellSpacing w:w="144" w:type="dxa"/>
  …
</w:tblPrEx>
```

The tblCellSpacing element as a child of tblPrEx specifies the default cell spacing between all cells in final two rows in the current table, in this case 144 twentieths of a point. *end example*]

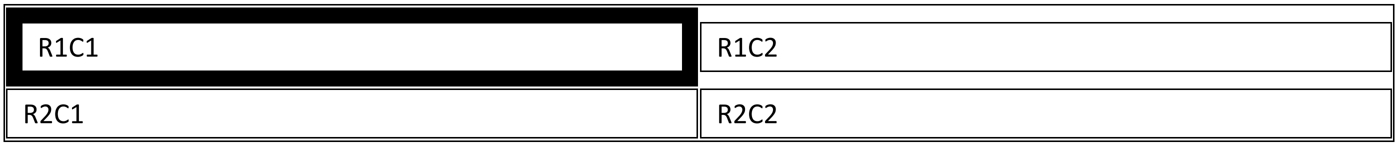
This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.45 tblCellSpacing (Table Cell Spacing Default)

This element specifies the default table cell spacing (the spacing between adjacent cells and the edges of the table) for all cells in the parent table. If specified, this element specifies the minimum amount of space which shall be left between all cells in the table including the width of the table borders in the calculation. This setting shall be superseded by a table-level exception (§17.4.44) or the row cell spacing value (§17.4.43) in that order. It is important to note that table-level cell spacing shall be added outside of the text margins, which shall be aligned with the innermost starting edge of the text extents in a table cell.

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

[*Example*: Consider a table whose first cell has a six point wide table border, and a table cell spacing value of 0.01 inches. The resulting table would have 0.01 inches of space between each table cell regardless of the width of the cell border, as follows (notice that no border is covered by any other border):

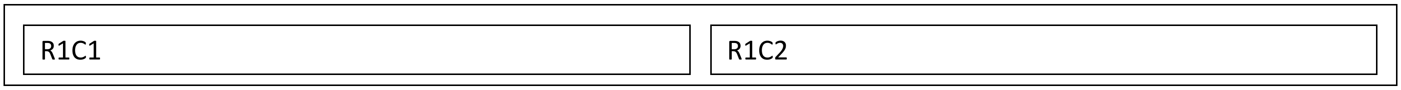


R1C1	R1C2
R2C1	R2C2

end example]

If this element is omitted, then the table shall inherit the table cell spacing from the associated table style. If table cell spacing is never specified in the style hierarchy, no cell spacing shall be added to the parent table.

[*Example*: Consider a table with a default cell spacing of 0.1 inches for all sides as follows:



R1C1	R1C2

R2C1	R2C2

This table cell spacing deafult is specified using the following WordprocessingML:

```
<w:tblPr>
  <w:tblCellSpacing w:w="144" w:type="dxa"/>
  …
</w:tblPr>
```

The tblCellSpacing element as a child of tblPr specifies the default cell spacing between all cells in the current table, in this case 144 twentieths of a point. *end example*]

This element’s content model is defined by the common table measurement definition in §17.4.87.

17.4.46 tblDescription (Table Description)

This element specifies the description for the table.

[*Example*: Consider a table which specifies a description. This object might contain the following XML markup:

```
<w:tbl>
    <w:tblPr>
        <w:tblDescription w:val="Here is the description of the table" />
        …
    </w:tblPr>
</w:tbl>
```

end example]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:

Attributes	Description

	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the nearest ancestorstructured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.4.47 tblGrid (Previous Table Grid)

This element specifies a previous table grid state, the modifications to which shall be attributed to a revision by a particular author and at a particular time. This element contains the table grid settings which were previously in place before a specific set of revisions by one author. The *table grid* is a definition of the set of grid columns which define all of the shared vertical edges of the table, as well as default widths for each of these grid columns. These grid column widths are then used to determine the size of the table based on the table layout algorithm used (§17.4.52;§17.4.53).

[*Example*: Consider the following table with four vertical edges (grid columns):

<table>
<tr><td></td><td colspan="2"></td><td></td></tr>
<tr><td colspan="2"></td><td></td><td></td></tr>
<tr><td colspan="2"></td><td></td><td></td></tr>
</table>

If we now modify this table by reducing the size of the last column without changing the overall table width, as follows:

<table>
<tr><td></td><td colspan="2"></td><td></td></tr>
<tr><td colspan="2"></td><td></td><td></td></tr>
<tr><td colspan="2"></td><td></td><td></td></tr>
</table>

This table would have a table grid consisting of four grid columns as follows:

```
<w:tblGrid>
  <w:gridCol w:w="2088"/>
  <w:gridCol w:w="1104"/>
  <w:gridCol w:w="3583"/>
  <w:gridCol w:w="2801"/>
  <w:tblGridChange w:id="1">
    <w:tblGrid>
      <w:gridCol w:w="2088"/>
      <w:gridCol w:w="1104"/>
      <w:gridCol w:w="3192"/>
      <w:gridCol w:w="3192"/>
    </w:tblGrid>
  </w:tblGridChange>
</w:tblGrid>
```

The tblGrid element as a child of tblGridChange contains the previous definition for the table grid, consisting of all for grid columns as well as the original widths for those columns. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblGridBase) is located in §A.1. *end note*]

17.4.48 tblGrid (Table Grid)

This element specifies the table grid for the current table. The *table grid* is a definition of the set of grid columns which define all of the shared vertical edges of the table, as well as default widths for each of these grid columns. These grid column widths are then used to determine the size of the table based on the table layout algorithm used (§17.4.52;§17.4.53).

If the table grid is omitted, then a new grid shall be constructed from the actual contents of the table assuming that all grid columns have a width of 0.

[*Example*: Consider the following table with four vertical edges (grid columns):

<table>
<tr><td></td><td colspan="2"></td><td></td></tr>
<tr><td colspan="2"></td><td></td><td></td></tr>
<tr><td colspan="2"></td><td></td><td></td></tr>
</table>

This table would have a table grid consisting of four grid columns as follows:

©ISO/IEC 2012 – All rights reserved

```
<w:tblGrid>
  <w:gridCol w:w="2088"/>
  <w:gridCol w:w="1104"/>
  <w:gridCol w:w="3192"/>
  <w:gridCol w:w="3192"/>
</w:tblGrid>
```

The tblGrid element contains the current definition for the table grid, consisting of all for grid columns as well as default widths for those columns. *end example*]

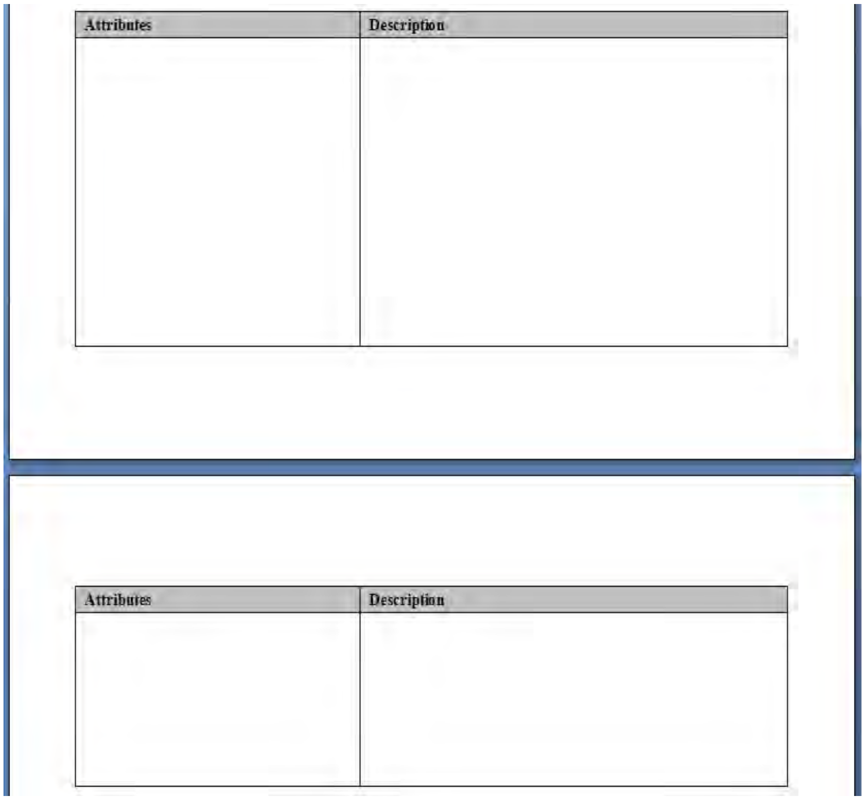
[*Note*: The W3C XML Schema definition of this element's content model (CT_TblGrid) is located in §A.1. *end note*]

17.4.49 tblHeader (Repeat Table Row on Every New Page)

This element specifies that the current table row shall be repeated at the top of each new page on which part of this table is displayed. This gives this table row the behavior of a 'header' row on each of these pages. This element can be applied to any number of rows at the top of the table structure in order to generate multi-row table headers.

If this element is omitted, this table row shall not be repeated on each new page on which the table is displayed. As well, if this row is not contiguously connected with the first row of the table (that is, if this table row is not either the first row, or all rows between this row and the first row are not marked as header rows) then this property shall be ignored.

[*Example*: Consider a table which must have its first row repeated on each new page, like the attribute listings in ISO/IEC 29500, for example:



Attributes	Description

Attributes	Description

Notice that the first row in the table is repeated on the top of the second page. This requirement would be specified as follows in the WordprocessingML for that row:

```
<w:trPr>
  <w:tblHeader />
</w:trPr>
```

The tblHeader element specifies that this table row is repeated as a header row at the top of each page. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.4.50 tblInd (Table Indent from Leading Margin)

This element specifies the indentation which shall be added before the leading edge of the current table in the document (the left edge in a left-to-right table, and the right edge in a right-to-left table). This indentation should shift the table into the text margin by the specified amount.

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If this element is omitted, then the table shall inherit the table indentation from the associated table style. If table indentation is never specified in the style hierarchy, no indentation shall be added to the parent table. If the resulting justification on any table row is not `left` after applying the value of the jc element from the three levels of this property (§17.4.26;§17.4.27;§17.4.28), then this property shall be ignored.

[*Example*: Consider a table which must be indented one inch from the left margin, as follows:

R1C1	R1C2
R2C1	R2C2

This setting would be specified using the following WordprocessingML:

```
<w:tblPr>
  <w:jc w:val="start"/>
  <w:tblInd w:w="1440" w:type="dxa"/>
</w:tblPr>
```

If the properties on this table were now modified to justify it on the right side by setting the value of the jc element to `right`, as follows:

```
<w:tblPr>
  <w:jc w:val="end"/>
  <w:tblInd w:w="1440" w:type="dxa"/>
</w:tblPr>
```

This table would now have no indent, as the justification is no longer on the leading edge (`left`):

R1C1	R1C2
R2C1	R2C2

end example]

This element's content model is defined by the common table measurement definition in §17.4.87.

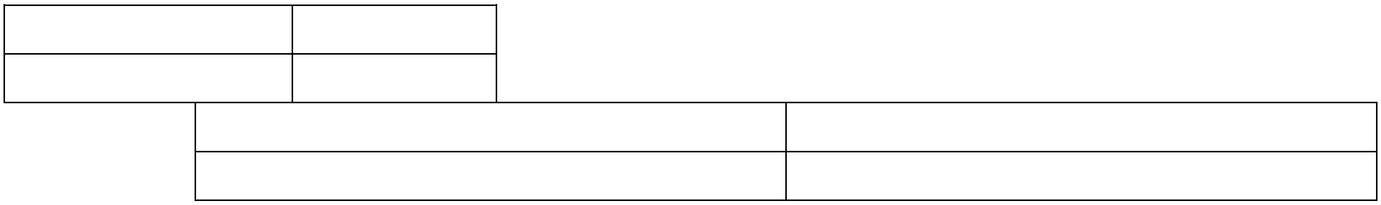
17.4.51 tblInd (Table Indent from Leading Margin Exception)

This element specifies the indentation which shall be added before the leading edge of the set of parent table rows which have this set of table-level property exceptions applied. This indentation should shift the table into the text margin by the specified amount in the document (the left edge in a left-to-right table, and the right edge in a right-to-left table).

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If this element is omitted, then the table shall inherit the table indentation from the associated table level property setting. If the resulting justification on the parent table row is not `left` after applying the value of the jc element from the three levels of this property (§17.4.26;§17.4.27;§17.4.28), then this property shall be ignored.

[*Example*: Consider a table in which the last two rows must be indented one inch from the left margin via a table-level property exception definition, as follows:



This setting would be specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:tblInd w:w="1440" w:type="dxa"/>
</w:tblPrEx>
```

The tblInd element as a child of tblPrEx specifies that the rows with the table-level property exception must be indented by `1440` twentieths of a point (one inch). *end example*]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.52 tblLayout (Table Layout)

This element specifies the algorithm which shall be used to lay out the contents of this table within the document. When a table is displayed in a document, it can either be displayed using a fixed width or autofit layout algorithm (each discussed in the simple type referenced by the val attribute).

If this element is omitted, then the value of this element shall be assumed to be `auto`.

[*Example*: Consider a table which must use the fixed width table layout algorithm. This requirement is specified using the following WordprocessingML:

```
<w:tblPr>
  <w:tblLayout w:type="fixed"/>
</w:tblPr>
```

The tblLayout element specifies that the table must use the `fixed` layout algorithm. *end example*]

Attributes	Description
type (Table Layout Setting)	Specifies the algorithm which shall be used to lay out the contents of the parent table (see simple type definition for details on each algorithm used). [*Example*: Consider a table which must use the AutoFit width table layout algorithm. This requirement is specified using the following WordprocessingML:

Attributes	Description

	`<w:tblPr>`
	`  <w:tblLayout w:type="autofit"/>`
	`</w:tblPr>`
	The tblLayout element specifies that the table must use the auto layout algorithm. *end example*]
	The possible values for this attribute are defined by the ST_TblLayoutType simple type (§17.18.87).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblLayoutType) is located in §A.1. *end note*]

17.4.53 tblLayout (Table Layout Exception)

This element specifies the algorithm which shall be used to lay out the contents of all rows with this table within the table which have the set of table-level property exceptions specified by the parent element. When a table is displayed in a document, it can either be displayed using a fixed width or autofit layout algorithm (each discussed in the simple type referenced by the val attribute).

If this element is omitted, then the value of this element shall be assumed to be auto.

[*Example*: Consider a table which must use the fixed width table layout algorithm. This requirement is specified using the following WordprocessingML:

```
<w:tblPrEx>
  <w:tblLayout w:type="fixed"/>
</w:tblPrEx>
```

The tblLayout element specifies that the table must use the fixed layout algorithm. *end example*]

Attributes	Description

type (Table Layout Setting)	Specifies the algorithm which shall be used to lay out the contents of the parent table (see simple type definition for details on each algorithm used).
	[*Example*: Consider a table which must use the AutoFit width table layout algorithm. This requirement is specified using the following WordprocessingML:
	`<w:tblPr>`
	`  <w:tblLayout w:type="autofit"/>`
	`</w:tblPr>`
	The tblLayout element specifies that the table must use the auto layout algorithm. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TblLayoutType simple type (§17.18.87).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblLayoutType) is located in §A.1. *end note*]

17.4.54 tblLook (Table Style Conditional Formatting Settings Exception)

This element specifies the components of the conditional formatting of the referenced table style (if one exists) which shall be applied to the set of table rows with the current table-level property exceptions. A table style can specify up to six different optional conditional formats [*Example*: Different formatting for first column. *end example*], which then can be applied or omitted from individual table rows in the parent table.

The default setting is to apply the row and column banding formatting, but not the first row, last row, first column, or last column formatting.

[*Example*: Consider a table which must use the following conditional formatting properties from the referenced table style:

- First row conditional formatting
- Last row conditional formatting
- No row banding formatting
- No column banding formatting

The resulting WordprocessingML would be specified as follows:

```
<w:tblPrEx>
  <w:tblLook w:firstRow="true" w:lastRow="true" w:noHBand="true"
w:noVBand="true" />
</w:tblPrEx>
```

The tblLook element specifies which components of the table style are applied to the current table. *end example*]

Attributes	Description
firstColumn (First Column)	Specifies that the first column conditional formatting shall be applied to the table. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
firstRow (First Row)	Specifies that the first row conditional formatting shall be applied to the table. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
lastColumn (Last	Specifies that the last column conditional formatting shall be applied to the table.

Attributes	Description
Column)	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
lastRow (Last Row)	Specifies that the last row conditional formatting shall be applied to the table. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
noHBand (No Horizontal Banding)	Specifies that the horizontal banding conditional formatting shall not be applied to the table. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
noVBand (No Vertical Banding)	Specifies that the vertical banding conditional formatting shall not be applied to the table. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblLook) is located in §A.1. *end note*]

17.4.55 tblLook (Table Style Conditional Formatting Settings)

This element specifies the components of the conditional formatting of the referenced table style (if one exists) which shall be applied to the current table. A table style can specify up to six different optional conditional formats [*Example*: Different formatting for first column. *end example*], which then can be applied or omitted from individual tables in the document.

The default setting is to apply the row and column banding formatting, but not the first row, last row, first column, or last column formatting.

[*Example*: Consider a table which must use the following conditional formatting properties from the referenced table style:

- First row conditional formatting
- Last row conditional formatting
- No row banding formatting
- No column banding formatting

The resulting WordprocessingML would be specified as follows:

```
<w:tblPr>
  <w:tblLook  w:firstRow="true" w:lastRow="true" w:noHBand="true"
w:noVBand="true" />
</w:tblPr>
```

The tblLook element specifies which components of the table style are applied to the current table. *end example*]

Attributes	Description
firstColumn (First Column)	Specifies that the first column conditional formatting shall be applied to the table. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
firstRow (First Row)	Specifies that the first row conditional formatting shall be applied to the table. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
lastColumn (Last Column)	Specifies that the last column conditional formatting shall be applied to the table. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
lastRow (Last Row)	Specifies that the last row conditional formatting shall be applied to the table. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
noHBand (No Horizontal Banding)	Specifies that the horizontal banding conditional formatting shall not be applied to the table. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
noVBand (No Vertical Banding)	Specifies that the vertical banding conditional formatting shall not be applied to the table. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

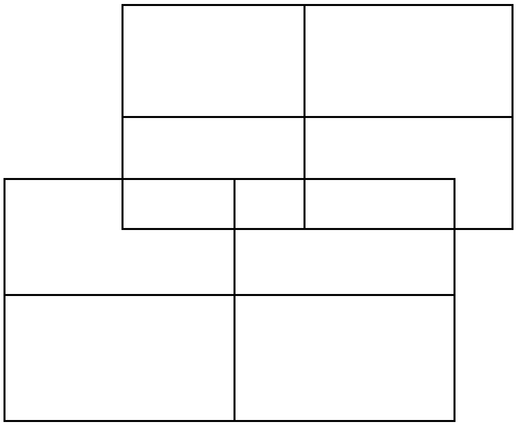
[*Note*: The W3C XML Schema definition of this element's content model (CT_TblLook) is located in §A.1. *end note*]

17.4.56 tblOverlap (Floating Table Allows Other Tables to Overlap)

This element specifies whether the current table shall allow other floating tables to overlap its extents when the tables are displayed in a document. If specified, then no adjustment shall be made to prevent tables whose properties would normally cause them to overlap from overlapping when displayed. If turned off, then the tables shall be adjusted as needed to prevent them from overlapping when displayed by adjusting the floating table properties as needed.

If this element is omitted on a given table, then this table shall allow other tables to overlap when displayed. If the parent table is not floating via the tblpPr element (§17.4.57), then this element shall be ignored.

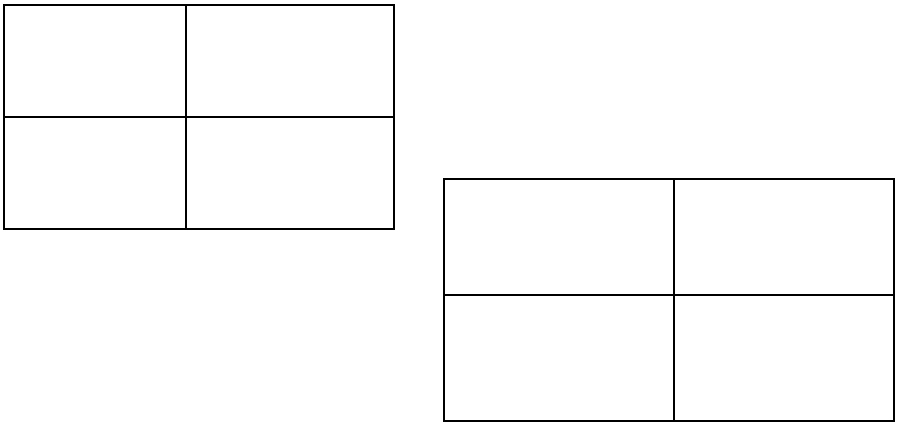
[*Example*: Consider two floating tables in a WordprocessingML document which overlap when displayed, as follows:



If either of these tables specifies that it must not allow overlapping, using the following WordprocessingML:

```
<w:tblPr>
  <w:tblOverlap w:val="never"/>
</w:tblPr>
```

The resulting tables must not overlap, and must be adjusted at display time to prevent any overlapping, for example:



The tblOverlap element with a value of never specifies that the specified table cannot overlap with other floating tables in the document. *end example*]

Attributes	Description
val (Floating Table Overlap Setting)	Specifies whether a floating table shall allow other floating tables in the document to overlap its extents when displayed.

Attributes	Description

	[*Example*: The following WordprocessingML specifies that the table is not allowed to overlap:
	`<w:tblOverlap w:val="never" />`
	end example]
	The possible values for this attribute are defined by the ST_TblOverlap simple type (§17.18.88).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblOverlap) is located in §A.1. *end note*]

17.4.57 tblpPr (Floating Table Positioning)

This element specifies information about the current table with regard to floating tables. *Floating tables* are tables in a document which are not part of the main text flow in the document, and are instead absolutely positioned with a specific size and position relative to non-frame content in the current document.

The first piece of information specified by the tblpPr element is that the current table is actually a floating table. This information is specified simply by the presence of the tblpPr element in table's properties. If the tblpPr element is omitted, the table shall not floating in the document.

The second piece of information is the positioning of the table, which is specified by the attribute values stored on the tblpPr element. In all absolute positioning cases, the positioning of the table is relative to its top-left corner position. For relative positioning (e.g. center), the positioning of the table is relative to its entire frame.

Note that the table still has a logical position in the file (its location within the block-level elements in the document). This logical location shall be used to calculate the position of the table relative to a paragraph, using the next regular (non-table, non-frame) paragraph in the document.

[*Example*: Consider a floating table which is positioned three inches from the edge of the page extents on both its top and left edges (i.e. the top-left corner occurs at 3" x 3"). This floating table would be specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblpPr w:leftFromText="144" w:rightFromText="144" w:topFromText="144"
w:bottomFromText="144" w:vertAnchor="page" w:horzAnchor="page" w:tblpX="4320"
w:tblpY="4320"/>
    …
  </w:tblPr>
  …</w:tbl>
```

The presence of the tblpPr element dictates that this table is a floating table, and its attributes specify that the floating table shall be anchored 4320 twentieths of a point (3 inches) from the top and left edges of the current page. *end example*]

Attributes	Description

bottomFromText (Distance From Bottom of Table to Text)	Specifies the minimum distance which shall be maintained between the current floating table and the top of text in the paragraph which is below this floating table.
	This distance is expressed in twentieths of a point.
	If this attribute is omitted, its value shall be assumed to be 0.
	[*Example*: Consider a floating table which should have a minimum of a one-half inch spacing from any text on its bottom side. This constraint would be specified using the following WordprocessingML:
	`<w:tblPr>`
	`  <w:tblpPr … w:bottomFromText="720" />`
	`</w:tblPr>`
	The bottomFromText attribute specifies that the spacing between text and this floating table shall be a minimum of 720 twentieths of a point. *end example*]
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

horzAnchor (Table Horizontal Anchor)	Specifies the base object from which the horizontal positioning in the tblpX and/or tblpXSpec attribute should be calculated.
	A floating table can be horizontally positioned relative to:
	• The vertical edge of the page before any runs of text (the left edge for left-to-right paragraphs, the right edge for right-to-left paragraphs)
	• The vertical edge of the text margin before any runs of text (the left edge for left-to-right paragraphs, the right edge for right-to-left paragraphs)
	• The vertical edge of the text margin for the column in which the anchor paragraph is located
	If this attribute is omitted, then its value shall be assumed to be page.
	[*Example*: Consider a floating table which should be positioned one inch to the right of its column in a left-to-right document. This floating table would be specified using the following WordprocessingML:
	`<w:tblPr>`
	`  <w:tblpPr … w:tblpX="1440" w:horzAnchor="margin" />`
	`</w:tblPr>`
	These table properties specify that they are relative to the current margin, and that

Attributes	Description
	relative to that column, the floating table should be `1440` twentieths of a point. *end example*] The possible values for this attribute are defined by the ST_HAnchor simple type (§17.18.35).

leftFromText (Distance From Left of Table to Text)	Specifies the minimum distance which shall be maintained between the current floating table and the edge of text in the paragraph which is to the left of this floating table.
	This distance is expressed in twentieths of a point.
	If this attribute is omitted, its value shall be assumed to be `0`.
	[*Example*: Consider a floating table which should have a minimum of a one-half inch spacing from any text on its left. This constraint would be specified using the following WordprocessingML:
	`<w:tblPr>`
	`  <w:tblpPr … w:leftFromText="720" />`
	`</w:tblPr>`
	The leftFromText attribute specifies that the spacing between text and this floating table must be a minimum of `720` twentieths of a point. *end example*]
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

rightFromText ((Distance From Right of Table to Text)	Specifies the minimum distance which shall be maintained between the current floating table and the edge of text in the paragraph which is to the right of this floating table.
	This distance is expressed in twentieths of a point.
	If this attribute is omitted, its value shall be assumed to be `0`.
	[*Example*: Consider a floating table which should have a minimum of a one-half inch spacing from any text on its right. This constraint would be specified using the following WordprocessingML:
	`<w:tblPr>`
	`  <w:tblpPr … w:rightFromText="720" />`
	`</w:tblPr>`
	The rightFromText attribute specifies that the spacing between text and this floating table must be a minimum of `720` twentieths of a point. *end example*]
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

Attributes	Description

tblpX (Absolute Horizontal Distance From Anchor)	Specifies an absolute horizontal position for the floating table. This absolute position is specified relative to the horizontal anchor specified by the horzAnchor attribute for this floating table.
	This value is expressed in twentieths of a point. If it is positive, then the floating table is positioned after the anchor object in the direction of horizontal text flow in this document. If it is negative, then the floating table is positioned before the anchor object in the direction of horizontal text flow in this document.
	If the tblpXSpec attribute is also specified, then this value is ignored. If this attribute is omitted, then its value shall be assumed to be `0`.
	[*Example*: Consider the following WordprocessingML fragment specifying a floating table:
	`<w:tbl>`
	`  <w:tblPr>`
	`    <w:tblpPr … w:horzAnchor="page" w:tblpX="1643"/>`
	`  </w:tblPr>`
	`  …`
	`</w:tbl>`
	This floating table specifies that it should be located exactly `1643` twentieths of a point after the vertical edge of the page (from the horizAnchor attribute). *end example*]
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

tblpXSpec (Relative Horizontal Alignment From Anchor)	Specifies a relative horizontal position for the floating table. This relative position is specified relative to the horizontal anchor specified by the horizAnchor attribute for this floating table.
	If omitted, this attribute is not specified and the value of the tblpX attribute determines the absolute horizontal position of the floating table. If specified, the position for this attribute supersedes any value which is specified in the tblpX attribute, and that value is ignored.
	[*Example*: Consider the following WordprocessingML fragment specifying a floating table:
	`<w:tbl>`
	`  <w:tblPr>`
	`    <w:tblpPr … w:horzAnchor="page" w:tblpX="1643"`
	`w:tblpXSpec="left"/>`
	`  </w:tblPr>`
	`  …`
	`</w:tbl>`
	This floating table specifies that it has a horizontal placement of exactly `1643` twentieths

Attributes	Description
	of a point relative to the page, but that exact placement is overridden by the presence of the tblpXSpec attribute to place the table on the `left` side of the page. *end example*] The possible values for this attribute are defined by the ST_XAlign simple type (§22.9.2.18).

tblpY (Absolute Vertical Distance From Anchor)	Specifies an absolute vertical position for the floating table. This absolute position is specified relative to the vertical anchor specified by the vertAnchor attribute for this floating table.
	This value is expressed in twentieths of a point. If it is positive, then the floating table is positioned after the anchor object in the direction of vertical text flow in this document. If it is negative, then the floating table is positioned before the anchor object in the direction of vertical text flow in this document.
	If the tblpYSpec attribute is also specified, then this value is ignored. If this attribute is omitted, then its value shall be assumed to be `0`.
	[*Example*: Consider the following WordprocessingML fragment specifying a floating table:
	`<w:tbl>`
	`  <w:tblPr>`
	`    <w:tblpPr … w:vertAnchor="text" w:tblpY="73" />`
	`  </w:tblPr>`
	`  …`
	`</w:tbl>`
	This floating table specifies that it should be located exactly 79 twentieths of a point below the top vertical edge of the anchor's paragraph's `text` (from the vertAnchor attribute), assuming that the vertical text direction is top to bottom. *end example*]
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

tblpYSpec (Relative Vertical Alignment from Anchor)	Specifies a relative vertical position for the floating table. This relative position is specified relative to the vertical anchor specified by the vertAnchor attribute for this floating table.
	If omitted, this attribute is not specified and the value of the tblpY attribute determines the absolute horizontal position of the floating table. If specified, the position for this attribute supersedes any value which is specified in the tblpY attribute, and that value is ignored, unless the vertAnchor is set to `text`, in which case any relative positioning is not allowed, and is itself ignored.
	[*Example*: Consider the following WordprocessingML fragment specifying a floating table:
	`<w:tbl>`
	`  <w:tblPr>`

Attributes	Description

	`<w:tblpPr … w:vertAnchor="margin" w:tblpY="73" w:tblpYSpec="center"/>`
	`</w:tblPr>`
	`…`
	`</w:tbl>`
	This floating table specifies that it has a vertical placement of exactly 73 twentieths of a point relative to the top margin, but that exact placement is overridden by the presence of the tblpYSpec attribute to place the table in the center of the margin. *end example*]
	The possible values for this attribute are defined by the ST_YAlign simple type (§22.9.2.20).

topFromText (Distance From Top of Table to Text)	Specifies the minimum distance which shall be maintained between the current floating table and the bottom edge of text in the paragraph which is above this floating table.
	This distance is expressed in twentieths of a point.
	If this attribute is omitted, its value shall be assumed to be 0.
	[*Example*: Consider a floating table which should have a minimum of a one-half inch spacing from any text above it. This constraint would be specified using the following WordprocessingML:
	`<w:tblPr>`
	`  <w:tblpPr … w:topFromText="720" />`
	`</w:tblPr>`
	The topFromText attribute specifies that the spacing between text and this floating table must be a minimum of 720 twentieths of a point. *end example*]
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

vertAnchor (Table Vertical Anchor)	Specifies the base object from which the vertical positioning in the tblpY attribute should be calculated. A floating table can be horizontally positioned relative to: • The horizontal edge of the page before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections) • The horizontal edge of the text margin before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections) • The horizontal edge of the page before any runs of text (the top edge for top-to-bottom sections, the bottom for bottom-to-top sections) If this attribute is omitted, then its value shall be assumed to be page.

Attributes	Description

	[*Example*: Consider a floating table which should be positioned two inches below the page top in a top-to-bottom document. This floating table would be specified using the following WordprocessingML:
	
	`<w:tblPr>`
	`  <w:tblpPr … w:tblpY="2880" w:vertAnchor="page" />`
	`</w:tblPr>`
	
	These floating table properties specify that they are relative to the anchor `page`, and that relative to that page, the table should be `2880` twentieths of a point in the direction of the flow of text (down, in this case). *end example*]
	
	The possible values for this attribute are defined by the ST_VAnchor simple type (§17.18.100).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblPPr) is located in §A.1. *end note*]

17.4.58 tblPr (Previous Table Properties)

This element specifies a previous set of table properties, the modifications to which shall be attributed to a revision by a particular author and at a particular time. This element contains the table property settings which were previously in place before a specific set of revisions by one author. These properties affect the appearance of all rows and cells within the parent table, but can be overridden by individual table-level exception, row, and cell level properties, as defined by each property.

[*Example*: Consider the following simple WordprocessingML table:

If the table justification is set to `center` and the table shading to set to red with revision marking on, as follows:

🟥	🟥

The revision tracked on this table would be specified as follows in the WordprocessingML:

```
  <w:tblPr>
    <w:tblStyle w:val="TableGrid"/>
    <w:tblW w:w="0" w:type="auto"/>
    <w:jc w:val="center"/>
    <w:shd w:val="clear" w:color="auto" w:fill="FF0000"/>
    <w:tblLook w:firstRow="true" w:firstColumn="true"
      w:noVBand="true" />
    <w:tblPrChange w:id="0" … >
      <w:tblPr>
        <w:tblStyle w:val="TableGrid"/>
        <w:tblW w:w="0" w:type="auto"/>
        <w:tblLook w:firstRow="true" w:firstColumn="true"
        w:noVBand="true" />
      </w:tblPr>
    </w:tblPrChange>
  </w:tblPr>
```

The tblPr element as a child of tblPrChange contains the previous definition for the table properties, consisting of the properties set before the current tracked revision. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblPrBase) is located in §A.1. *end note*]

17.4.59 tblPr (Table Properties)

This element specifies the set of table-wide properties applied to the current table. These properties affect the appearance of all rows and cells within the parent table, but can be overridden by individual table-level exception, row, and cell level properties as defined by each property.

[*Example*: Consider the following simple WordprocessingML table:

This table defines a one point single border for all border types and is set to 100% of page width - both table-wide properties. The resulting table is represented by the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:w="0" w:type="auto"/>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:start w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:end w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideH w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideV w:val="single" w:sz="4" w:space="0" w:color="auto"/>
    </w:tblBorders>
  </w:tblPr>
  …
</w:tbl>
```

In this example, the tblW element (§17.4.63) defines the total width of the table, which, in this case, is set to a type of auto, which specifies that the table should be automatically sized to fit its contents. The tblBorders element (§17.4.38) specifies each of the table's borders, and specifies a one point border on the top, left, bottom, right and inside horizontal and vertical border. *end example*]

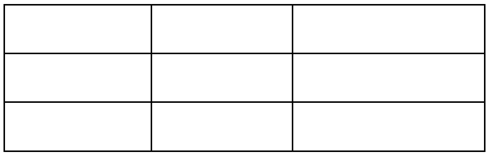
[*Note*: The W3C XML Schema definition of this element's content model (CT_TblPr) is located in §A.1. *end note*]

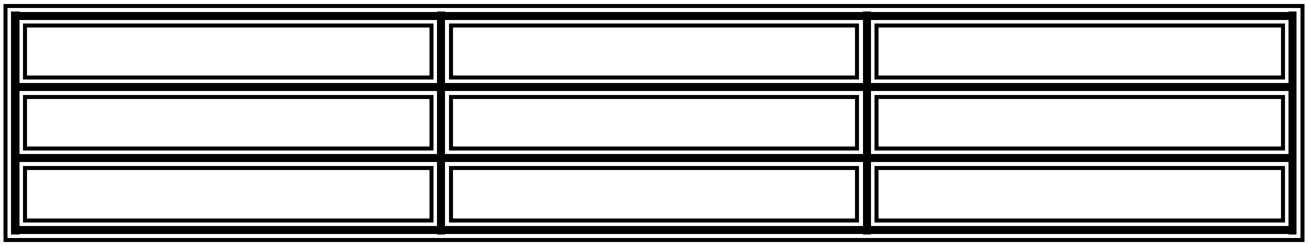
17.4.60 tblPrEx (Table-Level Property Exceptions)

This element specifies a set of table properties which shall be applied to the contents of this row in place of the table properties specified in the tblPr element.

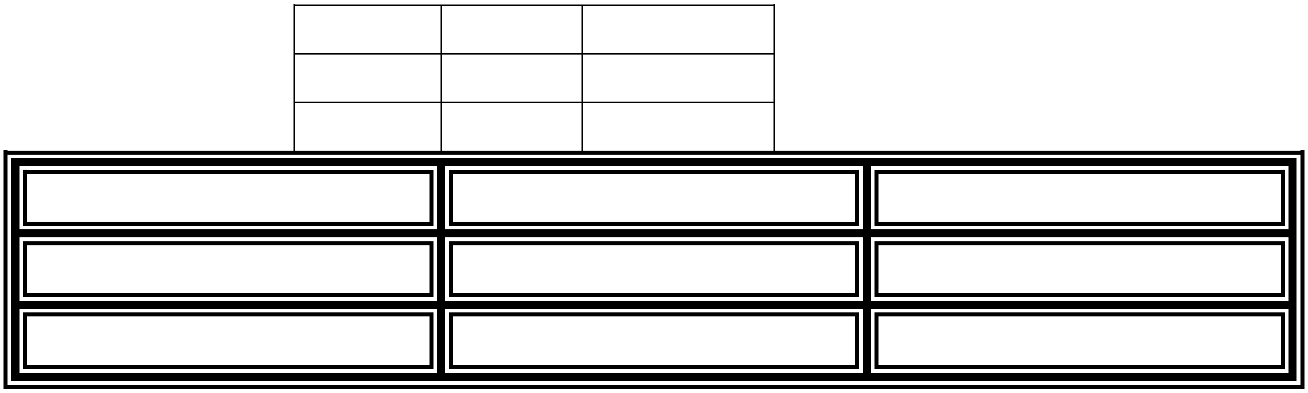
[*Note*: These properties are typically used in cases involving legacy documents, as well as cases where two existing independent tables are merged (in order to prevent the look of the second table from being superseded by the first table). *end note*]

[*Example*: Consider the following two tables in a WordprocessingML document:





These two tables each have a different set of table level borders. If the interceding paragraphs between these two tables is removed and the tables are merged together, it is obviously undesirable to have the second table lose its formatting and match the properties of the first table. Therefore, when the tables are merged as follows (note that there is now only one table):



The resulting WordprocessingML for the last three rows of the table would include the following set of table-level property exceptions:

```
<w:tr>
  <w:tblPrEx>
    <w:tblBorders>
      <w:top w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
      <w:start w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
      <w:bottom w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
      <w:end w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
      <w:insideH w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
      <w:insideV w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="auto"/>
    </w:tblBorders>
  </w:tblPrEx>
  …
</w:tr>
```

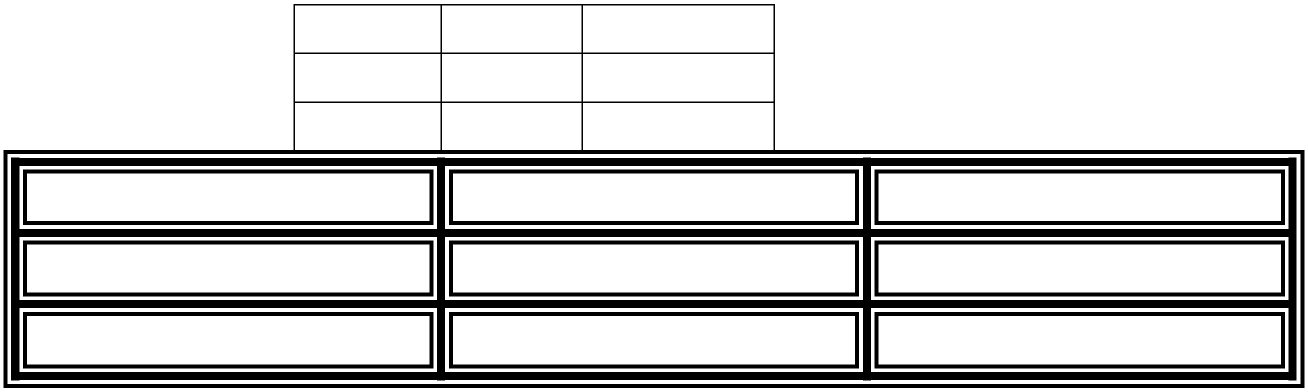
The tblPrEx element contains all table-level properties which are being overridden for the current row in the table. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblPrEx) is located in §A.1. *end note*]

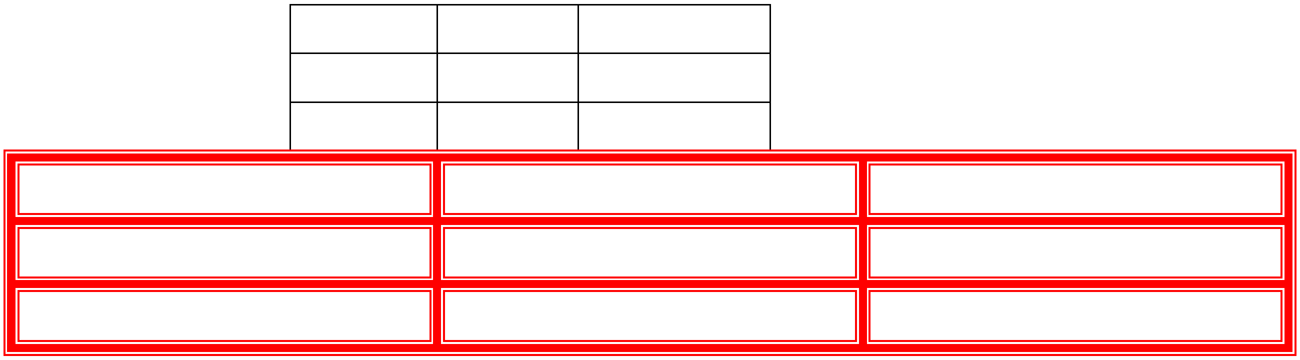
17.4.61 tblPrEx (Previous Table-Level Property Exceptions)

This element specifies a previous set of table-level property exceptions, the modifications to which shall be attributed to a revision by a particular author and at a particular time. This element contains the table-level property exceptions which were previously in place before a specific set of revisions by one author.

[*Example*: Consider the following two tables in a WordprocessingML document. If the interceding paragraphs between these two tables is removed and the tables are merged together, it is obviously undesirable to have the second table lose its formatting and match the properties of the first table. Therefore, when the tables are merged as follows (note that there is now only one table):



If the border type is changed to a red border of type `thinThickThinSmallGap` with revisions tracked, as follows:



The resulting WordprocessingML for the last three rows of the table would include the following set of table-level property exceptions with revision tracking:

```
  <w:tr>
    <w:tblPrEx>
      <w:tblBorders>
        <w:top w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
         w:color="auto"/>
        <w:start w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
         w:color="auto"/>
        <w:bottom w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
         w:color="auto"/>
        <w:end w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
         w:color="auto"/>
        <w:insideH w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
         w:color="auto"/>
        <w:insideV w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
         w:color="auto"/>
      </w:tblBorders>
      <w:tblPrExChange w:id="9" … >
        <w:tblPrEx>
          <w:tblBorders>
            <w:top w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
             w:color="FF0000"/>
            <w:start w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
             w:color="FF0000"/>
            <w:bottom w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
             w:color="FF0000"/>
            <w:end w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
             w:color="FF0000"/>
            <w:insideH w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
             w:color="FF0000"/>
            <w:insideV w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
             w:color="FF0000"/>
          </w:tblBorders>
        </w:tblPrEx>
      </w:tblPrExChange>
    </w:tblPrEx>
    …
  </w:tr>
```

The tblPrEx element as a child of tblPrExChange contains the previous definition for the table-level property exceptions, consisting of the properties set before the current tracked revision. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblPrExBase) is located in §A.1. *end note*]

17.4.62 tblStyle (Referenced Table Style)

This element specifies the style ID of the table style which shall be used to format the contents of this table.

This formatting is applied at the following location in the *style hierarchy*:

- Document defaults
- Table styles (this element)
- Numbering styles
- Paragraph styles
- Character styles
- Direct Formatting

This means that all properties specified in the style element (§17.7.4.17) with a styleId which corresponds to the value in this element's val attribute are applied to the table at the appropriate level in the hierarchy.

If this element is omitted, or it references a style which does not exist, then no table style shall be applied to the current table. As well, this property is ignored if the table properties are themselves part of a table style.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:tblPr>
  <w:tblStyle w:val="TestTableStyle" />
</w:tblPr>
```

This table specifies that it inherits all of the table properties specified by the table style with a styleId of `TestTableStyle`. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`

Attributes	Description
	`</w:sdtPr>` In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.4.63 tblW (Preferred Table Width)

This element specifies the preferred width for this table. This preferred width is used as part of the table layout algorithm specified by the tblLayout element (§17.4.52; §17.4.53) - full description of the algorithm in the ST_TblLayout simple type (§17.18.87).

All widths in a table are considered preferred because:

- The table shall satisfy the shared columns as specified by the tblGrid element (§17.4.48)
- Two or more widths can have conflicting values for the width of the same grid column
- The table layout algorithm (§17.18.87) can require a preference to be overridden

This value is specified in the units applied via its type attribute. Any width value of type `pct` for this element shall be calculated relative to the text extents of the page (page width excluding margins).

If this element is omitted, then the cell width shall be of type `auto`.

[*Example*: Consider a WordprocessingML table defined as follows:

```
<w:tbl>
  <w:tblPr>
    <w:tblW w:type="dxa" w:w="1440"/>
  </w:tblPr>
  …
</w:tbl>
```

This table specifies that it has a preferred table width of `1440` twentieths of a point (one inch). The resulting table would therefore be sized such that the table maintains that preferred width, as follows:

	Hello world	

The text `Hello world` makes the middle cell larger, and the other two cells are size to maintain the preferred widths of one inch for the overall table width:

	Hello world this is a longer string.	

However, when the middle table cell requires a larger width to accommodate non-breaking text, that preference can be overridden as needed:

	Hello worlddddddddddddddddddddddddddddd	

In this case, the middle cell's long non breaking string caused the table to be expanded to prevent breaking the string, and therefore to override the preferred width on the table. *end example*]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.64 tblW (Preferred Table Width Exception)

This element specifies the preferred width for the parent table row via a set of table-level property exceptions. This preferred width is used as part of the table layout algorithm specified by the tblLayout element (§17.4.52n; §17.4.53) - full description of the algorithm in the ST_TblLayout simple type (§17.18.87).

All widths in a table are considered preferred because:

- The table shall satisfy the shared columns as specified by the tblGrid element (§17.4.48)
- Two or more widths can have conflicting values for the width of the same grid column
- The table layout algorithm (§17.18.87) can require a preference to be overridden

This value is specified in the units applied via its type attribute. Any width value of type `pct` for this element shall be calculated relative to the text extents of the page (page width excluding margins).

If this element is omitted, then the cell width shall be of type `auto`.

[*Example*: Consider a row in a WordprocessingML table defined as follows:

```
  <w:tr>
    <w:tblPrEx>
      <w:tblW w:type="auto" w:w="1440"/>
    </w:tblPrEx>
    <w:trPr>

    </w:trPr>
    …
  </w:tr>
```

This table-level property exception specifies that it has a preferred table width of `1440` twentieths of a point (one inch). The resulting table row would therefore be sized such that the table maintains that preferred width, as follows:

	Hello world	

The text `Hello world` makes the middle cell larger, and the other two cells are size to maintain the preferred widths of one inch for the overall table width:

	Hello world this is a longer string.	

However, when the middle table cell requires a larger width to accommodate non-breaking text, that preference can be overridden as needed:

	Hello worlddddddddddddddddddddddddddddddd	

In this case, the middle cell's long non breaking string caused the table to be expanded to prevent breaking the string, and therefore to override the preferred width on the table row. *end example*]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.65 tc (Table Cell)

This element specifies a single cell in a table row, which contains the table's content. Table cells in WordprocessingML are analogous to HTML `td` elements.

A tc element has one formatting child element, tcPr (§17.4.69), which defines the properties for the cell. Each unique property on the table cell is specified by a child element of this element. As well, a table cell can contain any block-level content, which allows for the nesting of paragraphs and tables within table cells.

If a table cell does not include at least one block-level element, then this document shall be considered corrupt.

[*Example*: Consider a table consisting of a single table cell, which contains the text `Hello, world`:

Hello, world

This table cell's content is represented by the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcW w:w="0" w:type="auto"/>
  </w:tcPr>
  <w:p>
    <w:r>
      <w:t>Hello, world</w:t>
    </w:r>
  </w:p>
</w:tc>
```

The tc element contains a set of cell-level properties defined using the tcPr element, and a single block-level element - in this case, a paragraph. *end example*]

Attributes	Description

id (Table Cell Identifier)	Specifies a unique identifier for the current table cell. This identifier shall be unique within the table, and is used to identify this table cell as a header cell for other cells within the table, using the headers child element.
	If this element is omitted, this table cell has no unique identifier.
	[*Example*: Consider a table cell defined as follows:
	`<w:tc w:id="januaryeight">`
	`  …`
	`</w:tc>`
	The value in the id specifies a unique identifer of `januaryeight`. Other cells in the table are then able to reference this cell as a row or column header by referencing this ID. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Tc) is located in §A.1. *end note*]

17.4.66 tcBorders (Table Cell Borders)

This element specifies the set of borders for the edges of the current table cell, using the eight border types defined by its child elements.

If the cell spacing for any row is non-zero as specified using the tblCellSpacing element (§17.4.44; §17.4.43; §17.4.45), then there is never a border conflict (as the non-zero cell spacing is applied above and beyond each individual cell border's width) and all table, table-level exception, and table cell borders shall be displayed.

If the cell spacing is zero, then there can be a conflict between two adjacent cell borders [*Example*: Between the left border of all cells in the second column and the right border of all cells in the first column of the table. *end example*], which shall be resolved as follows:

5. If either conflicting table cell border is `nil` or `none` (no border), then the opposing border shall be displayed.
6. If a cell border conflicts with a table border, the cell border always wins.
7. Each border shall then be assigned a weight using the following formula, and the border value using this calculation shall be displayed over the alternative border:

$$W_{\text{border}} = \text{\# of lines in border} * \text{border number}$$

1. The border number shall be determined by this list:

single	1
thick	2
double	3
dotted	4
dashed	5
dotDash	6
dotDotDash	7
triple	8
thinThickSmallGap	9
thickThinSmallGap	10
thinThickThinSmallGap	11
thinThickMediumGap	12
thickThinMediumGap	13
thinThickThinMediumGap	14
thinThickLargeGap	15
thickThinLargeGap	16

thinThickThinLargeGap	17
wave	18
doubleWave	19
dashSmallGap	20
dashDotStroked	21
threeDEmboss	22
threeDEngrave	23
outset	24
inset	25

2. If the borders have an equal weight, than the higher of the two on this precedence list shall win:

   - single
   - thick
   - double
   - dotted
   - dashed
   - dotDash
   - dotDotDash
   - triple
   - thinThickSmallGap
   - thickThinSmallGap
   - thinThickThinSmallGap
   - thinThickMediumGap
   - thickThinMediumGap
   - thinThickThinMediumGap
   - thinThickLargeGap
   - thickThinLargeGap
   - thinThickThinLargeGap
   - wave
   - doubleWave
   - dashSmallGap
   - dashDotStroked
   - threeDEmboss
   - threeDEngrave
   - outset
   - inset

3. If the borders have an identical style, than each border color shall be assigned a brightness value as follows:

$$\text{Brightness} = \text{R} + \text{B} + 2 * \text{G}$$

The color with the smaller brightness value shall win.

4. If the borders have an identical brightness value above, than each border color shall be assigned a new brightness value as follows:

$$\text{Brightness} = \text{B} + 2 * \text{G}$$

The color with the smaller brightness value shall win.

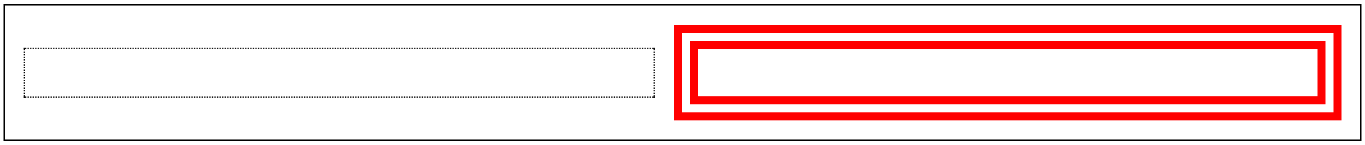
5. If the borders have an identical brightness value above, than each border color shall be assigned a brightness value as follows:

$$\text{Brightness} = \text{G}$$

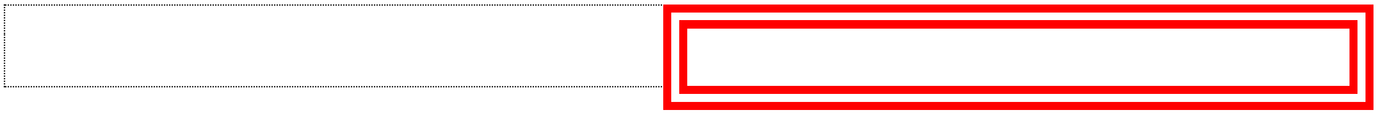
The color with the smaller brightness value shall win.

6. If the borders have an identical brightness value above, then they are functionally identical, and the first border in reading order should be displayed.

[*Example*: Consider the following two cell table (with exaggerated table cell spacing for clarity):



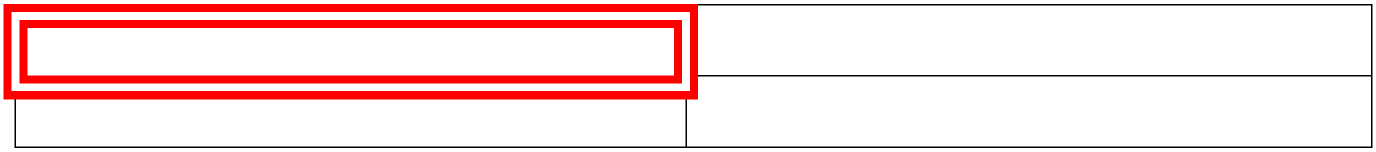
If we collapse the cell spacing, there are conflicting borders at all edges. For each cell/table border conflict, rule #2 says that the cell border must win. For the conflict in the center between two cell borders, rule #3 gives us a larger border weight for the right cell's border, resulting in the following table:



end example]

If this element is omitted, then this table shall have the borders specified by the associated table style. If no borders are specified in the style hierarchy, then this table shall not have any table borders.

[*Example*: Consider a table whose first cell specifies cell-level borders consisting of a think double red line, as follows:



These cell borders are specified using the following WordprocessingML:

```
<w:tcPr>
  <w:tcBorders>
    <w:top w:val="double" w:sz="24" w:space="0" w:color="FF0000"/>
    <w:start w:val="double" w:sz="24" w:space="0" w:color="FF0000"/>
    <w:bottom w:val="double" w:sz="24" w:space="0" w:color="FF0000"/>
    <w:end w:val="double" w:sz="24" w:space="0" w:color="FF0000"/>
  </w:tcBorders>
</w:tcPr>
```

The tcBorders element specifies the set of borders applied to the first cell as a 3 point double border. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TcBorders) is located in §A.1. *end note*]

17.4.67 tcFitText (Fit Text Within Cell)

This element specifies that the contents of the current cell shall have their inter-character spacing increased or reduced as necessary to fit the width of the text extents of the current cell. This setting shall behave identically to placing the contents of this paragraph in a run and using the fitText element (§17.3.2.14), if the width provided on that element matched the width of the current cell.

If this element is omitted, then the text in this cell shall not be fit to the current cell extents.

[*Example*: Consider a 2 row by two column table, in which the contents of the two cells in the first row have both have the fit text property set, as follows:

```
<w:tcPr>
  <w:tcFitText w:val="true"/>
</w:tcPr>
```

The resulting table cells must have their contents fit to the extents of the parent table cell, as follows:

Sample text in R1C1.	And this table cell instead contains a very very long string of sample text in R2C2.
R2C1	R2C2

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.4.68 tcMar (Single Table Cell Margins)

This element specifies a set of cell margins for a single table cell in the parent table.

This setting, if present, shall override the table cell margins from the table-level cell margins (§17.4.42).

[*Example*: Consider a table whose first cell is defined to have default cell margins of 0.5 inches for all sides rather then the table defaults, as follows:

R1C1
R2C1

This set of table cell margins is specified using the following WordprocessingML:

```
<w:tcPr>
  <w:tcMar>
    <w:top w:w="720" w:type="dxa"/>
    <w:start w:w="720" w:type="dxa"/>
    <w:bottom w:w="720" w:type="dxa"/>
    <w:end w:w="720" w:type="dxa"/>
  </w:tcMar>
  …
</w:tcPr>
```

The tcMar element as a child of tcPr specifies the set of table cell margins used for the first table cell, in this case, 720 twentieths of a point on all sides. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TcMar) is located in §A.1. *end note*]

17.4.69 tcPr (Table Cell Properties)

This element specifies the set of properties which shall be applied a specific table cell. Each unique property is specified by a child element of this element. In any instance where there is a conflict between the table level, table-level exception, or row level properties with a corresponding table cell property, these properties shall overwrite the table or row wide properties.

[*Example*: Consider a table where the cell width overwrites the table width represented in the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblCellMar>
      <w:start w:w="0" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  …
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:tcMar>
          <w:start w:w="720" w:type="dxa"/>
        </w:tcMar>
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
</w:tbl>
```

This table cell has a left cell margin of 720 twentieths of a point (one half inch) as specified in the tcMar element, which overwrites the table level setting of 0 left table cell margin. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TcPr) is located in §A.1. *end note*]

17.4.70 tcPr (Previous Table Cell Properties)

This element specifies a previous set of table cell properties, the modifications to which shall be attributed to a revision by a particular author and at a particular time. This element contains the table cell property settings which were previously in place before a specific set of revisions by one author. Each unique property is specified by a child element of this element. In any instance where there is a conflict between the table level, table-level exception, or row level properties with a corresponding table cell property, these properties shall overwrite the table or row wide properties.

[*Example*: Consider a basic two row by two column table as follows:

If the cell shading in the first cell is set to red with revision tracking enabled, as follows:

🟥	

This revision is specified as follows in the associated WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcW w:w="4788" w:type="dxa"/>
    <w:shd w:val="clear" w:color="auto" w:fill="FF0000"/>
    <w:tcPrChange w:id="2" …>
      <w:tcPr>
        <w:tcW w:w="4788" w:type="dxa"/>
      </w:tcPr>
    </w:tcPrChange>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The tcPr element beneath the tcPrChange element specifies the set of table cell properties which were in place before the current revision to the document. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TcPrInner) is located in §A.1. *end note*]

17.4.71 tcW (Preferred Table Cell Width)

This element specifies the preferred width for this table cell. This preferred width is used as part of the table layout algorithm specified by the tblLayout element (§17.4.52; §17.4.53) - full description of the algorithm in the ST_TblLayout simple type (§17.18.87).

All widths in a table are considered preferred because:

- The table shall satisfy the shared columns as specified by the tblGrid element (§17.4.48)
- Two or more widths can have conflicting values for the width of the same grid column
- The table layout algorithm (§17.18.87) can require a preference to be overridden

This value is specified in the units applied via its type attribute. Any width value of type pct for this element shall be calculated relative to the overall width of the table.

If this element is omitted, then the cell width shall be of type auto.

[*Example*: Consider a WordprocessingML table defined as follows:

```
<w:tbl>
  <w:tr>
    <w:tc>
      <w:tcPr>
        <w:tcW w:type="pct" w:w="33.3%"/>
      </w:tcPr>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:tcW w:type="pct" w:w="33.3%"/>
      </w:tcPr>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:tcW w:type="pct" w:w="33.3%"/>
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
</w:tbl>
```

This table specifies that it has no preferred table width, but each cell must be exactly 33.3 percent of the overall table width. The resulting table would therefore be sized such that all columns are of the width of the maximum column, as follows:

	Hello world	

The text `Hello world` makes the middle cell larger, and the other two cells are increased in size to maintain the preferred widths of one-third of the overall table width. However, when the middle table cell requires a larger size to accommodate non-breaking text, that preference can be overridden as needed:

	Hello worldddddddddddddddddddddddddddddddddd	

In this case, the middle cell's long non breaking string caused the table to reach the text margins on the page, and therefore to override the preferred widths on the empty cells. *end example*]

This element’s content model is defined by the common table measurement definition in §17.4.87.

17.4.72 textDirection (Table Cell Text Flow Direction)

This element specifies the direction of the text flow for this table cell.

If this element is omitted on a given table cell, its value is determined by the setting previously set at any level of the style hierarchy (i.e. that previous setting remains unchanged). If this setting is never specified in the style hierarchy, then the table cell shall inherit the text flow settings from the parent section.

[*Example*: Consider a table with one cell in which all the table cell's text flow is oriented vertically, flowing from right to left horizontally within that cell:

Text in this table cell

This table cell would specify this text flow using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:textDirection w:val="rl" />
  </w:tcPr>
  …
</w:tc>
```

The textDirection element specifies via the `rl` value in the val attribute that the text flow is to be oriented vertically, with subsequent lines stacked from right to left. *end example*]

Attributes	Description

val (Direction of Text Flow)	Specifies the direction of the text flow for this object.
	[*Example*: Consider a document with a section in which text must be oriented vertically, flowing from left to right horizontally on the page. This setting requires the following WordprocessingML:
	`<w:sectPr>`
	`  …`
	`  <w:textDirection w:val="lr" />`
	`</w:sectPr>`
	The textDirection element specifies via the `lr` value in the val attribute that the text flow be oriented vertically, with subsequent lines stacked from left to right. *end example*]
	The possible values for this attribute are defined by the ST_TextDirection simple type (§17.18.93).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextDirection) is located in §A.1. *end note*]

17.4.73 tl2br (Table Cell Top Left to Bottom Right Diagonal Border)

This element specifies the border which shall be displayed on the top left side to bottom right diagonal within the current table cell.

If this element is omitted, then the top left to bottom right diagonal of this table cell shall not have a cell border, and its border can use the table's border settings as appropriate.

[*Example*: Consider a table in which the first cell in the first row specifies a top left to bottom right diagonal cell border as follows:

R1C1	R1C2
R2C1	R2C2

This diagonal cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
       <w:tl2br w:val="double" w:sz="4" w:space="0" w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The tl2br element specifies a ½ point border of type `double` on the table cell's diagonal. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.4.74 top (Table Cell Top Border)

This element specifies the border which shall be displayed at the top of the current table cell. The appearance of this table cell border in the document shall be determined by the following settings:

- If the net tblCellSpacing element value (§17.4.44;§17.4.43;§17.4.45) applied to the cell is non-zero, then the cell border shall always be displayed
- Otherwise, the display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

If this element is omitted, then the top of this table cell shall not have a cell border, and its border can use the table's border settings as appropriate.

[*Example*: Consider a table in which the first cell in the first row specifies a top cell border , as follows:

R1C1	R1C2
R2C1	R2C2

This top cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
      <w:top w:val="thinThickThinSmallGap" w:sz="24" w:space="0"
w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The top element specifies a three point border of type `thinThinThickSmallGap`. *end example*]

This element's content model is defined by the common border properties definition in §17.3.4.

17.4.75 top (Table Cell Top Margin Default)

This element specifies the amount of space which shall be left between the top extent of the cell contents and the top border of all table cells within the parent table. This setting can be overridden by the table cell top margin definition specified by the top element contained within the table cell's properties (§17.4.77).

This value is specified in the units applied via its type attribute. Any width value of type `pct` or auto for this element shall be ignored.

If this element is omitted, then it shall inherit the table cell margin from the associated table style. If a top margin is never specified in the style hierarchy, then this table shall have no top cell padding by default (excepting individual cell overrides).

[*Example*: Consider a two by two table in which the default table cell top margin is specified to be exactly 0.25 inches, as follows (marked with an arrow in the first table cell below):

↕ R1C1	R2C1
R2C1	R2C2

This table property is specified using the following WordprocessingML markup:

```
<w:tbl>
  <w:tblPr>
    <w:tblCellMar>
      <w:top w:w="360" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  …
</w:tbl>
```

Every cell in the table has a default cell margin setting it to 360 twentieths of a point. *end example*]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.76 top (Table Top Border)

This element specifies the border which shall be displayed at the top of the current table. The appearance of this table border in the document shall be determined by the following settings:

- The display of the border is subject to the conflict resolution algorithm defined by the tcBorders element (§17.4.66) and the tblBorders element (§17.4.39;§17.4.38)

If this element is omitted, then the top of this table shall have the border specified by the associated table style. If no top border is specified in the style hierarchy, then this table shall not have a top border.

[*Example*: Consider a table in which the table properties specifies a top table border, as follows:

R1C1	R1C2
R2C1	R2C2

This top table border is specified using the following WordprocessingML:

```
<w:tbl>
  <w:tblPr>
    <w:tblBorders>
      <w:top w:val="thinThickThinMediumGap" w:sz="24" w:space="0"
w:color="D0D0D0" w:themeColor="accent3" w:themeTint="99"/>
    </w:tblBorders>
  </w:tblPr>
  …
</w:tbl>
```

The top element specifies a three point top table border of type thinThinThickMediumGap. *end example*].

This element's content model is defined by the common border properties definition in §17.3.4.

17.4.77 top (Table Cell Top Margin Exception)

This element specifies the amount of space which shall be left between the top extent of the cell contents and the top border of a specific table cell within a table. This setting shall override the table cell top margin definition specified by the top element contained within the table properties (§17.4.75).

This value is specified in the units applied via its type attribute. Any width value of type `pct` or `auto` for this element shall be ignored.

If omitted, then this table cell shall use the bottom cell margins defined in the top element contained within the table properties (§17.4.75).

[*Example*: Consider a table with two cells in which the first table cell's top margin is specified via an exception to be ten times larger (0.2 inches vs. 0.02 inches) than the other table cell margins:

This text fills the extents of the cell.
So does this

The first cell in the table would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:tcMar>
      <w:top w:w="288" w:type="dxa" />
    </w:tcMar>
  </w:tcPr>
</w:tc>
```

The first cell in this table has an exception applied to the table cell top cell margin setting it to 288 twentieths of a point (0.2 inches). *end example*]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.78 tr (Table Row)

This element specifies a single table row, which contains the table's cells. Table rows in WordprocessingML are analogous to HTML `tr` elements.

A tr element has one formatting child element, trPr (§17.4.81), which defines the properties for the row. Each unique property on the table row is specified by a child element of this element. As well, a table row can contain any row-level content, which allows for the use of table cells.

If a table cell does not include at least one child element other than the row properties, then this document shall be considered corrupt.

[*Example*: Consider a table consisting of a single table cell, which contains the text `Hello, world`:

Hello World

This table row's content is represented by the following WordprocessingML:

```
<w:tr>
  <w:tc>
    <w:tcPr>
      <w:tcW w:w="0" w:type="auto"/>
    </w:tcPr>
    <w:p>
      <w:r>
        <w:t>Hello, world</w:t>
      </w:r>
    </w:p>
  </w:tc>
</w:tr>
```

The tr element contains a single row-level element - in this case, a table cell. *end example*]

Attributes	Description
rsidDel (Revision Identifier for Table Row Deletion)	Specifies a unique identifier used to track the editing session when the row was deleted from the main document. All rsid* attributes throughout this document with the same value, if present, shall indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidR (Revision Identifier for Table Row)	Specifies a unique identifier used to track the editing session when the table row was added to the main document. All rsid* attributes throughout this document with the same value, if present, shall indicate that those regions were modified during the same editing session (time between subsequent save actions).

Attributes	Description
	A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidRPr (Revision Identifier for Table Row Glyph Formatting)	Specifies a unique identifier used to track the editing session when the glyph character representing the table row mark was last modified in the main document. All rsid* attributes throughout this document with the same value, if present, shall indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidTr (Revision Identifier for Table Row Properties)	Specifies a unique identifier used to track the editing session when the table row's properties were last modified in this document. All rsid* attributes throughout this document with the same value, if present, shall indicate that those regions were modified during the same editing session (time between subsequent save actions). A producer can choose to increment the revision save ID value to indicate subsequent editing sessions to indicate the order of the modifications relative to other modifications in this document. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Row) is located in §A.1. *end note*]

17.4.79 tr2bl (Table Cell Top Right to Bottom Left Diagonal Border)

This element specifies the border which shall be displayed on the top right to bottom left diagonal within the current table cell.

If this element is omitted, then the top right to bottom left diagonal of this table cell shall not have a cell border, and its border can use the table's border settings as appropriate.

[*Example*: Consider a table in which the second cell in the second row specifies a top right to bottom left diagonal cell border as follows:

R1C1	R1C2
R2C1	R2C2

This diagonal cell border is specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    …
    <w:tcBorders>
       <w:tr2bl w:val="double" w:sz="4" w:space="0" w:color="FF0000"/>
    </w:tcBorders>
  </w:tcPr>
  <w:p/>
</w:tc>
```

The tr2bl element specifies a ½ point border of type double on the table cell's diagonal. *end example*

This element's content model is defined by the common border properties definition in §17.3.4.

17.4.80 trHeight (Table Row Height)

This element specifies the height of the current table row within the current table. This height shall be used to determine the resulting height of the table row, which can be absolute or relative (depending on its attribute values).

If omitted, then the table row shall automatically resize its height to the height required by its contents (the equivalent of an hRule value of auto).

[*Example*: Consider the following WordprocessingML table:

Some text in R1C1.	

Examining the WordprocessingML for this table, the trHeight element is not specified, so the row heights are automatically determined by their contents (in the first row, the text Some text in R1C1.). If the first row shall be restricted to 0.1 inches high (144 twentieths of a point) regardless of its contents, that would be specified using the trHeight element as follows:

```
<w:trPr>
  <w:trHeight w:val="144" w:hRule="exact"/>
</w:trPr>
```

The resulting table row would be exactly 144 twentieths of a point high:

Some text in R1C1	

end example]

Attributes	Description

hRule (Table Row Height Type)	Specifies the meaning of the height specified for this table row.
	
	The meaning of the value of the val attribute is defined based on the value of the hRule attribute for this table row as follows:
	• If the value of hRule is `auto`, then the table row's height should be automatically determined based on the height of its contents. The h value is ignored.
	• If the value of hRule is `atLeast`, then the table row's height should be at least the value the h attribute.
	• If the value of hRule is `exact`, then the table row's height should be exactly the value of the h attribute.
	
	If this attribute is omitted, then its value shall be assumed to be `auto`.
	
	[*Example*: Consider the following paragraph containing a table row:
	
	`<w:tr>`
	  `<w:trPr>`
	    `<w:trHeight w:val="2189" w:hRule="atLeast"/>`
	  `</w:trPr>`
	  …
	`</w:tr>`
	
	The hRule attribute specifies a value of `atLeast`, so the table row is a minimum of 2189 twentieths of a point high regardless of its contents, since its val value is 2189 twentieths of a point. *end example*]
	
	The possible values for this attribute are defined by the ST_HeightRule simple type (§17.18.37).

val (Table Row Height)	Specifies the table row's height. This height is expressed in twentieths of a point. If this attribute is omitted, then its value shall be assumed to be `0`. The meaning of the value of the val attribute is defined based on the value of the hRule attribute for this table row as follows: • If the value of hRule is `auto`, then the table row's height should be automatically determined based on the height of its contents. This value is ignored.

Attributes	Description

	• If the value of hRule is `atLeast`, then the table row's height should be at least the value of this attribute.
	• If the value of hRule is `exact`, then the table row's height should be exactly the value of this attribute.
	
	[*Example*: Consider the following table row:
	
	`<w:tr>`
	`  <w:trPr>`
	`    <w:trHeight w:val="2189" w:hRule="atLeast"/>`
	`  </w:trPr>`
	`  …`
	`</w:tr>`
	
	The val attribute specifies a value of `2189` twentieths of a point, so this table row is a minimum of `2189` twentieths of a point high regardless of its contents (growing if needed), since its hRule value is set to `atLeast`. *end example*]
	
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Height) is located in §A.1. *end note*]

17.4.81 trPr (Table Row Properties)

This element specifies the set of row-level properties applied to the current table row. Each unique property is specified by a child element of this element. These properties affect the appearance of all cells in the current row within the parent table, but can be overridden by individual cell-level properties, as defined by each property.

[*Example*: Consider the following WordprocessingML table:

Some text in R1C1	

The first row must have a table-row level property which specifies that it shall be restricted to 0.1 inches high (144 twentieths of a point) regardless of its contents, that would be specified using the trHeight element as follows:

```
<w:trPr>
  <w:trHeight w:val="144" w:hRule="exact"/>
  …
</w:trPr>
```

The trPr element specifies the set of table row properties applied to the current table row in the document, in this case a row height requirement using the trHeight element (§17.4.80). *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_TrPr) is located in §A.1. *end note*]

17.4.82 trPr (Previous Table Row Properties)

This element specifies a previous set of table cell properties, the modifications to which shall be attributed to a revision by a particular author and at a particular time. This element contains the table cell property settings which were previously in place before a specific set of revisions by one author. Each unique property is specified by a child element of this element. These properties affect the appearance of all cells in the current row within the parent table, but can be overridden by individual cell-level properties, as defined by each property.

[*Example*: Consider the following WordprocessingML table:

Some text in R1C1.	

This table has a row height for row one of exactly 0.1". If we change that to a row height of at least 0.1" with revision marking enabled, the table would appear as follows:

Some text in R1C1.	

The resulting WordprocessingML would be:

```
<w:tr>
  <w:trPr>
    <w:trHeight w:val="144"/>
    <w:trPrChange w:id="2" … >
      <w:trPr>
        <w:trHeight w:hRule="exact" w:val="144"/>
      </w:trPr>
    </w:trPrChange>
  </w:trPr>
  …
</w:tr>
```

The trPr element as a child of trPrChange specifies the set of table row properties which were in place before the current revision to the document. *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_TrPrBase) is located in §A.1. *end note*]

17.4.83 vAlign (Table Cell Vertical Alignment)

This element specifies the vertical alignment for text within the current table cell. The vertical alignment of this text is determined by the value of the val attribute.

[*Example*: Consider a table with a single cell with text vertically aligned to the bottom of the cell:

R1C1

This requirement would be specified using the following WordprocessingML:

```
<w:tc>
  <w:tcPr>
    <w:vAlign w:val="bottom" />
  </w:tcPr>
  <w:p>
    <w:r>
      <w:t>R1C1</w:t>
    </w:r>
  </w:p>
</w:tc>
```

The vAlign element specifies the vertical alignment of the cell contents, in the case, the `bottom` of the table cell. *end example*]

Attributes	Description
val (Vertical Alignment Setting)	Specifies the vertical alignment for text between the top and bottom margins of the parent container (page or table cell). [*Example*: Consider a region where the text must be vertically centered in the parent element. This would require a val value of `center`, in order to specify that all justification vertically must be centered relative to the parent. For a section, this setting would be specified as follows: `<w:vAlign w:val="center" />` The val attribute of `center` specifies that the content is centered relative to its container (in this case, the page). *end example*] The possible values for this attribute are defined by the ST_VerticalJc simple type (§17.18.101).

©ISO/IEC 2012 – All rights reserved

[*Note*: The W3C XML Schema definition of this element's content model (CT_VerticalJc) is located in §A.1. *end note*]

17.4.84 vMerge (Vertically Merged Cell)

This element specifies that this cell is part of a vertically merged set of cells in a table. The val attribute on this element determines how this cell is defined with respect to the previous cell in the table (i.e., whether this cell continues the vertical merge or starts a new merged group of cells).

If this element is omitted, then this cell shall not be part of any vertically merged grouping of cells, and any vertically merged group of preceding cells shall be closed. If a vertically merged group of cells do not span the same set of grid columns, then the document is non-conformant.

[*Example*: Consider a table with three rows and two columns with the last column completely vertically merged:

<table>
<tr><td></td><td rowspan="3"></td></tr>
<tr><td></td></tr>
<tr><td></td></tr>
</table>

The second cell in the first row starts a vertical merge that is completed in the last cell, resulting in the following WordprocessingML:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:vMerge w:val="restart"/>
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
```

©ISO/IEC 2012 – All rights reserved

```
    <w:tr>
      <w:tc>
        …
      </w:tc>
      <w:tc>
        …
      </w:tc>
      <w:tc>
        <w:tcPr>
          <w:vMerge w:val="continue"/>
        </w:tcPr>
        …
      </w:tc>
    </w:tr>
    <w:tr>
      <w:tc>
        …
      </w:tc>
      <w:tc>
        …
      </w:tc>
      <w:tc>
        <w:tcPr>
          <w:vMerge w:val="continue"/>
        </w:tcPr>
        …
      </w:tc>
    </w:tr>
  </w:tbl>
```

The vMerge element defines the cells that are vertically merged, and how each cell is merged together. *end example*]

Attributes	Description
val (Vertical Merge Type)	Specifies how the table cell is part of a vertically merged region. This determines whether the cell should join onto an existing grouping of merged cells if any exist, or start a new group of merged cells. Refer to the simple type definition for a full description of each type. If this attribute is omitted, its value shall be assumed to be `continue`. [*Example*: Consider a table cell where a vertical cell merge begins . This setting is represented as the following WordprocessingML:

Attributes	Description

	`<w:tcPr>`
	`    <w:vMerge w:val="restart"/>`
	`</w:tcPr>`
	
	The attribute value of `restart` specifies that this element must start a new vertically merged region in this table. *end example*]
	
	The possible values for this attribute are defined by the ST_Merge simple type (§17.18.57).

[*Note*: The W3C XML Schema definition of this element's content model (CT_VMerge) is located in §A.1. *end note*]

17.4.85 wAfter (Preferred Width After Table Row)

This element specifies the preferred width for the total number of grid columns after this table row as specified in the gridAfter element (§17.4.14). This preferred width is used as part of the table layout algorithm specified by the tblLayout element (§17.4.52; §17.4.53) - full description of the algorithm in the ST_TblLayout simple type (§17.18.87).

All widths in a table are considered preferred because:

- The table shall satisfy the shared columns as specified by the tblGrid element (§17.4.48)
- Two or more widths can have conflicting values for the width of the same grid column
- The table layout algorithm (§17.18.87) can require a preference to be overridden

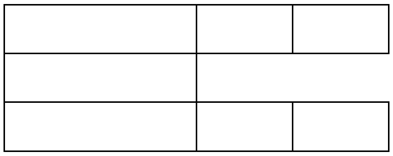
This value is specified in the units applied via its type attribute. Any width value of type `pct` for this element shall be calculated relative to the text extents of the page (page width excluding margins).

If this element is omitted, then the cell width shall be of type `auto`.

[*Example*: Consider a WordprocessingML table row defined as follows:

```
<w:tr>
  <w:trPr>
    <w:gridAfter w:val="2"/>
    <w:wAfter w:type="dxa" w:w="1440"/>
  </w:trPr>
  …
</w:tr>
```

This table specifies that it has a preferred table width of `1440` twentieths of a point (one inch) for the two grid columns after the end of that row. The resulting table would therefore be sized such that that set of grid columns is one inch whenever possible, for example the second row in this table:



end example]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.86 wBefore (Preferred Width Before Table Row)

This element specifies the preferred width for the total number of grid columns before this table row as specified in the gridAfter element (§17.4.14). This preferred width is used as part of the table layout algorithm specified by the tblLayout element (§17.4.52; §17.4.53) - full description of the algorithm in the ST_TblLayout simple type (§17.18.87).

All widths in a table are considered preferred because:

- The table shall satisfy the shared columns as specified by the tblGrid element (§17.4.48)
- Two or more widths can have conflicting values for the width of the same grid column
- The table layout algorithm (§17.18.87) might require a preference to be overridden

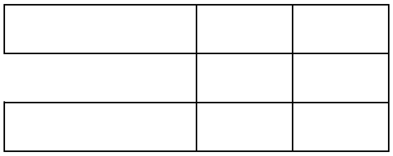
This value is specified in the units applied via its type attribute. Any width value of type `pct` for this element shall be calculated relative to the text extents of the page (page width excluding margins).

If this element is omitted, then the cell width shall be of type `auto`.

[*Example*: Consider a WordprocessingML table row defined as follows:

```
<w:tr>
  <w:trPr>
    <w:gridBefore w:val="1"/>
    <w:wBefore w:type="dxa" w:w="1440"/>
  </w:trPr>
  …
</w:tr>
```

This table specifies that it has a preferred table width of `1440` twentieths of a point (one inch) for the grid column before the start of the row. The resulting table would therefore be sized such that that grid column is one inch whenever possible, for example the second row in this table:



end example]

This element's content model is defined by the common table measurement definition in §17.4.87.

17.4.87 Table Measurement (CT_TblWidth)

This common complex type specifies a measurement to be used within a table. These properties contain two pieces of information:

- The type of measurement (percentage-based, absolute, or automatic)
- The value of the measurement

[*Example*: Consider the following table measurement:

```
<… w:type="pct" w:w="100%"/>
```

The type attribute specifies that the measurement is percentage-based, and the w attribute specifies the measurement is 100%. *end example*]

Attributes	Description
type (Table Width Type)	Specifies the units of the width property being defined by the parent element's w attribute. This property is used to define various properties of a table, including: cell spacing, preferred width, and table margins. If this attribute is omitted, then its value shall be assumed to be dxa (twentieths of a point). [*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows: `<w:bottom … w:type="dxa" />` This type must therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*] If the value of the type attribute and the actual measurement specified by the w attribute are contradictory, the type specified by the type attribute shall be ignored. The possible values for this attribute are defined by the ST_TblWidth simple type (§17.18.90).
w (Table Width Value)	Specifies the value of the width property being defined by the parent element. This property is used to define various properties of a table, including: cell spacing, preferred widths, and table margins. If this attribute is omitted, then its value shall be assumed to be `0`. [*Example*: Consider a table with a bottom margin with a width of `302`, as follows:

Attributes	Description
	`<w:bottom w:w="302" w:type="dxa" />` The value in the w attribute must therefore be used to determine the width being specified in the context of the units specified in the type attribute. In this case, the type is twentieths of a point (dxa), so the width is 302 twentieths of a point (.2097 inches). *end example*] The possible values for this attribute are defined by the ST_MeasurementOrPercent simple type (§17.18.107).

[*Note*: The W3C XML Schema definition of this complex type's content model (CT_TblWidth) is located in §A.1. *end note*]

17.5 Custom Markup

Within a WordprocessingML document, semantic information may be supplied beyond the presentation information specified by ISO/IEC 29500. [*Example*: An invoice document might wish to specify that a particular sentence of text is a customer name, in order for that information to be easily extracted from the document without the need to parse the text using regular expression matching or similar. *end example*]

There are three distinct forms in which extra-standard semantics can be inserted into a WordprocessingML document, each with their own specific intended usage:

- Smart tags
- Custom XML markup
- Structured document tags (content controls)

The elements and attributes which define each of these forms is described in the following clauses.

17.5.1 Custom XML and Smart Tags

The first form of extra-standard semantics that can be embedded in a WordprocessingML document is represented by smart tags. Implementations can establish sets of smart tags that allow semantic labels to be added around an arbitrary run or set of runs within a document to provide information about the type of data contained within.

[*Example*: Consider the following text in a WordprocessingML document, with a smart tag around the stock symbol 'CNTS':

> This is a stock symbol: CNTS

This text would translate to the following WordprocessingML markup:

```
<w:p w:rsidR="00672474" w:rsidRDefault="00672474">
  <w:r>
    <w:t xml:space="preserve">This is a stock symbol: </w:t>
  </w:r>
  <w:smartTag w:uri="http://www.example.com"
     w:element="stockticker">
    <w:r>
      <w:t>CNTS</w:t>
    </w:r>
  </w:smartTag>
</w:p>
```

As shown above, the smart tag is delimited by the smartTag element, which surrounds the run (or runs) which contain the text which is part of the smart tag. *end example*]

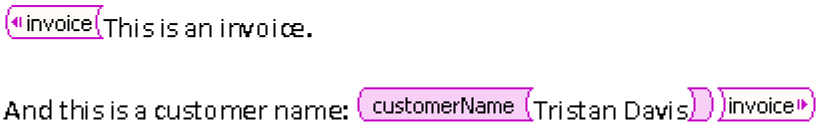
The smartTag element has two required attributes:

- The uri attribute is used to specify the namespace, in terms of a vocabulary or classification scheme, of which the term specified for this smart tag is a member. [*Example*: In the above example, the smart tag specifies `http://www.example.com` to identify the classification scheme. *end example*]
- The element attribute, in combination with the uri attribute, specifies the classification for this smart tag. [*Example*: In the above example, the smart tag specifies the classification term `stockticker` from the identified namespace. *end example*]

The second form of extra-standard semantics that can be embedded in a WordprocessingML document is custom XML markup. Custom XML markup allows the application of the XML elements defined in any schema syntax (XML Schema, NVDL, etc.) to be applied to the contents of a WordprocessingML document in two types of location: around a paragraph or set of paragraphs (at the block level); or around an arbitrary run or set of runs within a document (at the inline level) to provide semantics to that content within the context and structures defined by the associated schema definition.

The distinction between custom XML markup and smart tags is that custom XML markup is based on a schema, which may be specified using the attachedSchema element (§17.15.1.5). As a result, the custom XML elements can be validated against the schema. Also, as shown below, custom XML markup can be used at the block-level as well as on the inline (run) level.

[*Example*: Consider a simple XML Schema which defines two elements: a root element of invoice, and a child element of customerName - the first defining that this file's contents are an invoice, and the second specifying that the enclosed text as a customer's name:



This is an invoice.

And this is a customer name: Tristan Davis

This output would translate to the following WordprocessingML markup:

```
<w:customXml w:uri="http://www.example.com/2006/invoice" w:element="invoice">
  <w:p>
    <w:r>
      <w:t>This is an invoice.</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:r>
      <w:t xml:space="preserve">And this is a customer name: </w:t>
    </w:r>
    <w:customXml w:uri="http://www.example.com/2006/invoice"
w:element="customerName">
      <w:r>
        <w:t>Tristan Davis</w:t>
      </w:r>
    </w:customXml>
  </w:p>
</w:customXml>
```

As shown above, each of the XML elements from the supplied XML schema is represented within the document output as a customXml element. *end example*]

A custom XML element in a document has two required attributes.

- The first is the uri attribute, whose contents specify the namespace of the custom XML element in the document. In the example above, the elements each belong to the `http://www.example.com/2006/invoice` namespace.
- The second is the element attribute, whose contents specify the name of the custom XML element at this location in the document. In the example above, the root element is called invoice and the child element is called customerName.

As well as the required information specified above, custom XML elements can also specify any number of attributes (as specified in the associated XML Schema) on the element. To add this information, the customXmlPr (properties on the custom XML element) specify one or more attr elements.

[*Example*: Using the example above, we can add a type attribute to the customerName element as follows:

©ISO/IEC 2012 – All rights reserved

```
<w:customXml w:uri="http://www.example.com/2006/invoice"
w:element="customerName">
  <w:customXmlPr>
    <w:attr w:uri="http://www.example.com/2006/invoice" w:name="type"
w:val="individual"/>
  </w:customXmlPr>
  <w:r>
    <w:t>Tristan Davis</w:t>
  </w:r>
</w:customXml>
```

The resulting XML, as seen above, simply adds an attr element which specifies the attribute for the custom XML element. *end example*]…

17.5.1.1 attr (Custom XML Attribute)

This element specifies a custom XML attribute which shall be located on the parent custom XML element specified via the customXml element (§17.5.1.4;§17.5.1.5;§17.5.1.3; §17.5.1.6). The uri attribute can specify the Namespace of the custom XML attribute, and the name attribute shall specify the local name of the custom XML attribute. For any set of sibling attr elements, all the pairs of Namespace and local name shall be distinct.

[*Example*: Consider a custom XML element with the following properties:

```
<w:customXmlPr>
  <w:attr w:name="companyName" … />
  <w:attr w:name="companySymbol" … />
</w:customXmPr>
```

This set of custom XML properties specifies that the parent custom XML element must have two attributes associated with it, he first with a name of companyName, and the second with a name of companySymbol. *end example*]

Attributes	Description

name (Name)	Specifies the name of the current custom XML attribute or smart tag property.
	[*Example*: Consider a custom XML attribute which must have a name of companyName. This requirement would be specified using the following WordprocessingML:
	`<w:customXmlPr>`
	`  <w:attr w:name="companyName" … />`
	`</w:customXmlPr>`
	The name attribute specifies that the name for this property must be companyName. *end example*]
	The possible values for this attribute are defined by the ST_XmlName simple type

Attributes	Description
	(§22.9.2.21).

uri (Namespace)	Specifies the namespace URI of the current custom XML attribute or smart tag property.
	If this attribute is omitted, the URI shall be assumed to be null (no associated URI).
	[*Example*: Consider a smart tag property which must have a namespace URI of `http://schemas.openxmlformats.org/2006/example`. This requirement would be specified using the following WordprocessingML:
	`<w:smartTagPr>`
	`  <w:attr w:uri="http://schemas.openxmlformats`
	`.org/2006/example" … />`
	`</w:smartTagPr>`
	The uri attribute specifies that the namespace for this property must be `http://schemas.openxmlformats.org/2006/example`. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

val (Value)	Specifies the value of the current custom XML attribute or smart tag property.
	[*Example*: Consider a smart tag property which must have a value of `propertyValue`. This requirement would be specified using the following WordprocessingML:
	`<w:smartTagPr>`
	`  <w:attr … w:val="propertyValue" />`
	`</w:smartTagPr>`
	The val attribute specifies that the value for this property must be `propertyValue`. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Attr) is located in §A.1. *end note*]

17.5.1.2 attr (Smart Tag Property)

This element specifies a single smart tag property which shall be located on the parent smart tag, specified via the smartTag element (§17.5.1.9). The attributes on this element shall be used to specify the contents of smart tag property.

[*Example*: Consider a smart tag with the following properties:

```
<w:smartTagPr>
  <w:attr w:name="attributeOne" … />
  <w:attr w:name="attributeTwo" … />
</w:smartTagPr>
```

This property bag specifies that the parent smart tag must have two properties associated with it, the first with a name of `attributeOne`, and the second with a name of `attributeTwo`. *end example*].

Attributes	Description

name (Name)	Specifies the name of the current custom XML attribute or smart tag property.
	
	[*Example*: Consider a custom XML attribute which must have a name of `companyName`. This requirement would be specified using the following WordprocessingML:
	
	`<w:customXmlPr>`
	`  <w:attr w:name="companyName" … />`
	`</w:customXmlPr>`
	
	The name attribute specifies that the name for this property must be `companyName`. *end example*]
	
	The possible values for this attribute are defined by the ST_XmlName simple type (§22.9.2.21).

uri (Namespace)	Specifies the namespace URI of the current custom XML attribute or smart tag property.
	
	If this attribute is omitted, the URI shall be assumed to be null (no associated URI).
	
	[*Example*: Consider a smart tag property which must have a namespace URI of `http://schemas.openxmlformats.org/2006/example`. This requirement would be specified using the following WordprocessingML:
	
	`<w:smartTagPr>`
	`  <w:attr w:uri="http://schemas.openxmlformats`
	`.org/2006/example" … />`
	`</w:smartTagPr>`
	
	The uri attribute specifies that the namespace for this property must be `http://schemas.openxmlformats.org/2006/example`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

val (Value)	Specifies the value of the current custom XML attribute or smart tag property. [*Example*: Consider a smart tag property which must have a value of `propertyValue`. This requirement would be specified using the following WordprocessingML:

Attributes	Description

	`<w:smartTagPr>`
	`  <w:attr … w:val="propertyValue" />`
	`</w:smartTagPr>`
	
	The val attribute specifies that the value for this property must be `propertyValue`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Attr) is located in §A.1. *end note*]

17.5.1.3 customXml (Inline-Level Custom XML Element)

This element specifies the presence of a custom XML element around one or more inline level structures (runs, images, fields, etc.) within a paragraph. The attributes on this element shall be used to specify the name and namespace URI of the current custom XML element.

[*Example*: Consider a custom XML element with the name `firstName` that must be located around a two text runs in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:p>
  <w:customXml w:element="firstName" … >
    <w:r>
      …
    </w:r>
    <w:r>
      …
    </w:r>
  </w:customXml>
  …
</w:p>
```

The customXml element specifies that the name of the custom XML element is `firstName`, and the custom XML element contains a two text runs (it is an inline-level element). *end example*]

Attributes	Description
element (Element name)	Specifies the name of the current custom XML element or smart tag within the document. [*Example*: Consider a custom XML element which must have a name of `companyName`. This requirement would be specified using the following WordprocessingML:

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	`<w:customXml w:element="companyName" … >`
	`  …`
	`</w:customXml>`
	The element attribute specifies that the name for this element must be `companyName`. *end example*]
	The possible values for this attribute are defined by the ST_XmlName simple type (§22.9.2.21).

uri (Custom XML Markup Namespace)	Specifies the namespace URI of the current custom XML element or smart tag.
	If this attribute is omitted, the URI shall be assumed to be null (no associated URI).
	[*Example*: Consider a custom XML element which must have a namespace URI of `urn:customXmlExample`. This requirement would be specified using the following WordprocessingML:
	`<w:customXml … w:uri="urn:customXmlExample" >`
	`  …`
	`</w:customXml>`
	The uri attribute specifies that the namespace for this element must be `urn:customXmlExample`. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomXmlRun) is located in §A.1. *end note*]

17.5.1.4 customXml (Cell-Level Custom XML Element)

This element specifies the presence of a custom XML element around a single table cell. The attributes on this element shall be used to specify the name and namespace URI of the current custom XML element.

[*Example*: Consider a custom XML element with the name `company` that must be located around a single table cell in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tr>
  <w:customXml w:element="company" … >
    <w:tc>
      …
    </w:tc>
  </w:customXml>
  …
</w:tr>
```

The customXml element specifies that the name of the custom XML element is company, and the custom XML element contains a single table cell (it is a cell-level element). *end example*]

Attributes	Description

element (Custom XML Element Name)	Specifies the name of the current custom XML element or smart tag within the document.
	[*Example*: Consider a custom XML element which must have a name of companyName. This requirement would be specified using the following WordprocessingML:
	`<w:customXml w:element="companyName" … >`
	`…`
	`</w:customXml>`
	The element attribute specifies that the name for this element must be companyName. *end example*]
	The possible values for this attribute are defined by the ST_XmlName simple type (§22.9.2.21).

uri (Custom XML Element Namespace)	Specifies the namespace URI of the current custom XML element or smart tag.
	If this attribute is omitted, the URI shall be assumed to be null (no associated URI).
	[*Example*: Consider a custom XML element which must have a namespace URI of urn:customXmlExample. This requirement would be specified using the following WordprocessingML:
	`<w:customXml … w:uri="urn:customXmlExample" >`
	`…`
	`</w:customXml>`
	The uri attribute specifies that the namespace for this element must be urn:customXmlExample. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomXmlCell) is located in §A.1. *end note*]

17.5.1.5 customXml (Row-Level Custom XML Element)

This element specifies the presence of a custom XML element around a single table row. The attributes on this element shall be used to specify the name and namespace URI of the current custom XML element.

[*Example*: Consider a custom XML element with the name `invoiceItem` that must be located around a single table row in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tbl>
  <w:customXml w:element="invoiceItem" … >
    <w:tr>
      …
    </w:tr>
  </w:customXml>
  …
</w:tbl>
```

The customXml element specifies that the name of the custom XML element is `invoiceItem`, and the custom XML element contains a single table row (it is a row-level element). *end example*]

Attributes	Description

element (Custom XML Element Name)	Specifies the name of the current custom XML element or smart tag within the document.
	[*Example*: Consider a custom XML element which must have a name of `companyName`. This requirement would be specified using the following WordprocessingML:
	`<w:customXml w:element="companyName" … >`
	`…`
	`</w:customXml>`
	The element attribute specifies that the name for this element must be `companyName`. *end example*]
	The possible values for this attribute are defined by the ST_XmlName simple type (§22.9.2.21).

uri (Custom XML Element Namespace)	Specifies the namespace URI of the current custom XML element or smart tag. If this attribute is omitted, the URI shall be assumed to be null (no associated URI). [*Example*: Consider a custom XML element which must have a namespace URI of `urn:customXmlExample`. This requirement would be specified using the following

Attributes	Description

	WordprocessingML:
	`<w:customXml … w:uri="urn:customXmlExample" >`
	`  …`
	`</w:customXml>`
	The uri attribute specifies that the namespace for this element must be `urn:customXmlExample`. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomXmlRow) is located in §A.1. *end note*]

17.5.1.6 customXml (Block-Level Custom XML Element)

This element specifies the presence of a custom XML element around one or more block level structures (paragraphs, tables, etc.). The attributes on this element shall be used to specify the name and namespace URI of the current custom XML element.

[*Example*: Consider a custom XML element with the name `address` that must be located around a single paragraph in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:body>
  <w:customXml w:element="address" … >
    <w:p>
      …
    </w:p>
  </w:customXml>
  …
</w:body>
```

The customXml element specifies that the name of the custom XML element is `address`, and the custom XML element contains a single paragraph (it is a block-level element). *end example*]

Attributes	Description
element (Custom XML Element Name)	Specifies the name of the current custom XML element or smart tag within the document. [*Example*: Consider a custom XML element which must have a name of `companyName`. This requirement would be specified using the following WordprocessingML:

Attributes	Description

	`<w:customXml w:element="companyName" … >`
	`  …`
	`</w:customXml>`
	The element attribute specifies that the name for this element must be `companyName`. *end example*]
	The possible values for this attribute are defined by the ST_XmlName simple type (§22.9.2.21).

uri (Custom XML Element Namespace)	Specifies the namespace URI of the current custom XML element or smart tag.
	If this attribute is omitted, the URI shall be assumed to be null (no associated URI).
	[*Example*: Consider a custom XML element which must have a namespace URI of `urn:customXmlExample`. This requirement would be specified using the following WordprocessingML:
	`<w:customXml … w:uri="urn:customXmlExample" >`
	`  …`
	`</w:customXml>`
	The uri attribute specifies that the namespace for this element must be `urn:customXmlExample`. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomXmlBlock) is located in §A.1. *end note*]

17.5.1.7 customXmlPr (Custom XML Element Properties)

This element specifies the set of properties which shall be applied to the parent custom XML element.

[*Example*: Consider a custom XML element with the following properties specified:

```
<w:customXmlPr>
  <w:placeholder w:val="[Fill in your name]"/>
  <w:attr w:name="status" w:val="draft"/>
</w:customXmlPr>
```

This custom XML element specifies two properties: the presence of placeholder text via the placeholder element (§17.5.1.8) and a single custom XML attribute via the attr element (§17.5.1.1). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomXmlPr) is located in §A.1. *end note*]

17.5.1.8 placeholder (Custom XML Element Placeholder Text)

This element specifies the placeholder text which shall be displayed in place of this custom XML element when the contents of this custom XML markup are empty (i.e. there are no runs of text within the current custom XML element). If this custom XML element does contain run content, then this text shall not be displayed.

The val attribute stores the string of text which shall be displayed as the placeholder text. This string can be displayed in any font face/size desired by the hosting application.

[*Example*: Consider a custom XML element with the following properties specified:

```
<w:customXmlPr>
  <w:placeholder w:val="[Fill in your name]"/>
  <w:attr w:name="status" w:val="draft"/>
</w:customXmlPr>
```

The placeholder element specifies that this custom XML element must display the text contents `[Fill in your name]` whenever there is no run content within the parent custom XML element. For example, if the custom XML element was specified as follows:

```
<w:customXml>
  <w:customXmlPr>
    <w:placeholder w:val="[Fill in your name]"/>
  </w:customXmlPr>
  <w:p/>
</w:customXml>
```

This custom XML element has no run content and the placeholder text would be displayed. However, if there is run content, as follows:

```
<w:customXml>
  <w:customXmlPr>
    <w:placeholder w:val="[Fill in your name]"/>
  </w:customXmlPr>
  <w:p>
    <w:r>
      <w:t>Name</w:t>
    </w:r>
  </w:p>
</w:customXml>
```

This custom XML element now contains run content, and the placeholder text must not be displayed. *end example*]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>val (String Value)</td><td>
<p>Specifies that its contents contain a string.</p>
<p>The contents of this string are interpreted based on the context of the parent XML element.</p>
<p>[<i>Example</i>: Consider the following WordprocessingML fragment:</p>
<pre>
<w:pPr>
  <w:pStyle w:val="Heading1" />
</w:pPr>
</pre>
<p>The value of the val attribute is the ID of the associated paragraph style's <code>styleId</code>.</p>
<p>However, consider the following fragment:</p>
<pre>
<w:sdtPr>
  <w:alias w:val="SDT Title Example" />
  …
</w:sdtPr>
</pre>
<p>In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. <i>end example</i>]</p>
<p>The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).</p>
</td></tr>
</table>

[*Note*: The W3C XML Schema definition of this element’s content model (CT_String) is located in §A.1. *end note*]

17.5.1.9 smartTag (Inline-Level Smart Tag)

This element specifies the presence of a smart tag around one or more inline structures (runs, images, fields, etc.) within a paragraph. The attributes on this element shall be used to specify the name and namespace URI of the current smart tag.

[*Example*: Consider a smart tag with the name `firstName` that must be located around a two text runs in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:p>
  <w:smartTag w:element="firstName" … >
    <w:r>
      …
    </w:r>
    <w:r>
      …
    </w:r>
  </w:smartTag>
  …
</w:p>
```

The smartTag element specifies that the name of the smart tag is `firstName`, and the smart tag contains a two text runs (it is an inline-level smart tag). *end example*]

Attributes	Description

element (Smart Tag Name)	Specifies the name of the current custom XML element or smart tag within the document.
	[*Example*: Consider a custom XML element which must have a name of `companyName`. This requirement would be specified using the following WordprocessingML:
	`<w:customXml w:element="companyName" … >`
	`  …`
	`</w:customXml>`
	The element attribute specifies that the name for this element must be `companyName`. *end example*]
	The possible values for this attribute are defined by the ST_XmlName simple type (§22.9.2.21).

uri (Smart Tag Namespace)	Specifies the namespace URI of the current custom XML element or smart tag.
	If this attribute is omitted, the URI shall be assumed to be null (no associated URI).
	[*Example*: Consider a custom XML element which must have a namespace URI of `urn:customXmlExample`. This requirement would be specified using the following WordprocessingML:
	`<w:customXml … w:uri="urn:customXmlExample" >`
	`  …`
	`</w:customXml>`
	The uri attribute specifies that the namespace for this element must be `urn:customXmlExample`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SmartTagRun) is located in §A.1. *end note*]

17.5.1.10 smartTagPr (Smart Tag Properties)

This element specifies the set of properties which shall be applied to the parent smart tag.

[*Example*: Consider a smart tag with the following properties specified:

```
<w:smartTagPr>
  <w:attr w:name="date" w:val="01/01/2006"/>
  <w:attr w:name="status" w:val="draft"/>
</w:smartTagPr>
```

This smart tag specifies two properties: the presence of two smart tag properties via the attr element (§17.5.1.2). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SmartTagPr) is located in §A.1. *end note*]

17.5.2 Structured Document Tags

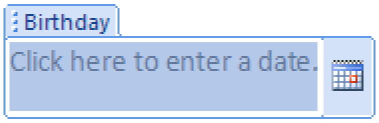
The final form of extra-standard semantics that can be embedded in a WordprocessingML document is represented by structured document tags (SDTs).

As shown above, smart tags and custom XML markup each provide a facility for embedding extra-standard semantics into the document: smart tags, via the ability to provide a basic namespace/name for a run or set of runs within a documents; and custom XML markup, via the ability to tag the document with XML elements and attributes specified by any XML Schema file.

However, each of these techniques, while they each provide a way to add the desired semantic information, does not provide a way to affect the presentation or interaction within the document. To bridge these two worlds, structured document tags allow both the specification of extra-standard semantics as well as the ability to influence the presentation of that data in the document.

This means that the implementation can define the semantics and context of the tag, but can then use a rich set of pre-defined properties to define its behavior and appearance within the WordprocessingML document's presentation.

[*Example*: Consider a region which should be tagged with the semantic of "birthday", for the user to enter their date or birth into the document. Ideally, this region would also utilize a date picker to allow the user to enter the date from a calendar:



This content would be specified using the following WordprocessingML:

```
<w:sdt>
  <w:sdtPr>
    <w:alias w:val="Birthday"/>
    <w:id w:val="8775518"/>
    <w:placeholder>
      <w:docPart w:val="DefaultPlaceholder_22479095"/>
    </w:placeholder>
    <w:showingPlcHdr/>
    <w:date>
      <w:dateFormat w:val="M/d/yyyy"/>
      <w:lid w:val="EN-US"/>
    </w:date>
  </w:sdtPr>
  <w:sdtContent>
    <w:p>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="PlaceholderText"/>
        </w:rPr>
        <w:t>Click here to enter a date…</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>
```

end example]

As shown above, each of the structured document tags in the WordprocessingML file is represented using the sdt element.

Within a structured document tag, there are two child elements which contain the definition and the content of this SDT. The first of these is the sdtPr element, which contains the set of properties specified for this structured document tag. The second is the sdtContent element, which contains all the content which is contained within this structured document tag.

17.5.2.1 alias (Friendly Name)

This element specifies the friendly name associated with the current structured document tag. The string representing the friendly name shall be stored on this element's val attribute.

If this element is omitted, then no friendly name shall be associated with the given structured document tag.

[*Example*: Consider the following properties on a structured document tag:

```
<w:sdtPr>
  <w:alias w:val="Birthday"/>
  …
</w:sdtPr>
```

This set of properties specifies via the alias element that the friendly name for the nearest ancestor structured document tag must be `Birthday`. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.5.2.2 bibliography (Bibliography Structured Document Tag)

This element specifies that the nearest ancestorstructured document tag shall be of type `bibliography`.

This setting does not require or imply that the contents of the structured document tag shall contain only a field of type BIBLIOGRAPHY, it shall only be used to specify that the structured document tag is of this kind, which can be used by an application as desired.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:bibliography/>
  </w:sdtPr>
  …
</w:sdt>
```

The bibliography element in this structured document tag's properties specify that the type of structured document tag is bibliography. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.5.2.3 calendar (Date Picker Calendar Type)

This element specifies the calendar which shall be displayed for the current date picker structured document tag, if a user interface is present for the structured document tag. The calendar information is stored on this element's val attribute.

If this element is omitted, then the calendar shall be gregorian.

[*Example*: Consider the following structured document tag properties:

```
<w:sdtPr>
  <w:date w:fullDate="2006-01-01T06:30:00Z">
    <w:calendar w:val="gregorian"/>
  </w:date>
</w:sdtPr>
```

The calendar element specifies that the calendar information for a calendar which might be displayed in the document must be the Gregorian calendar format (gregorian). *end example*]

Attributes	Description
val (Calendar Type Value)	Specifies a type of calendar, the use of which is determined by the parent XML element. If this attribute is omitted, then the calendar type shall be gregorian. [*Example*: Consider the following WordprocessingML for a document containing a structured document tag: `<w:sdtPr>`

Attributes	Description

	`    <w:date … >`
	`      <w:calendar w:val="japan"/>`
	`    </w:date>`
	`  </w:sdtPr>`
	The val attribute value of japan specifies that the Japanese Emperor Era calendar must be used; in this case, it is used for the calendar displayed for a date structured document tag. *end example*]
	The possible values for this attribute are defined by the ST_CalendarType simple type (§22.9.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CalendarType) is located in §A.1. *end note*]

17.5.2.4 citation (Citation Structured Document Tag)

This element specifies that the nearest ancestorstructured document tag shall be of type `citation`.

This setting does not require or imply that the contents of the structured document tag shall contain only a field of type `CITATION`, it shall only be used to specify that the structured document tag is of this type, which can be used by an application as desired.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:citation/>
  </w:sdtPr>
  …
</w:sdt>
```

The citation element in this structured document tag's properties specify that the type of structured document tag is `citation`. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.5.2.5 comboBox (Combo Box Structured Document Tag)

This element specifies that the nearest ancestorstructured document tag shall be a combo box when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- The child elements of this element specify choices which shall be presented to the user

- Formatting applied to any part of this structured document tag's contents shall apply to its entire contents

As well, the structured document tag shall satisfy the following restraints or the document shall be considered non-conformant:

- The contents shall only be contain a single run (one set of formatting properties)
- The contents shall not contain more than a single paragraph or table cell and shall not contain a table row or table

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:comboBox>
      …
    </w:comboBox>
  </w:sdtPr>
  …
</w:sdt>
```

The comboBox element in this structured document tag's properties specify that the type of structured document tag is a combo box. *end example*]

Attributes	Description
lastValue (Combo Box Last Saved Value)	Specifies the value associated with the current display text for the combo box structured document tag. If this structured document tag is not mapped to XML using the dataBinding element (§17.5.2.6), then this attribute shall be ignored. If this structured document tag is mapped to XML, it shall be used to determine whether the current display text in the combo box structured document tag shall be retained when the document is opened, as follows: • When the XML mapping is created, the content in the custom XML data is retrieved • If this content has an associated list item (matching its value attribute), then the corresponding display text shall be displayed in the structured document tag • If no list item exists, this content shall be matched against the lastValue attribute value. If the values match, the current display text shall be retained. If the values do not match, the current custom XML data content shall be the new display text (since no match exists in the combo box list items) [*Example*: Consider a combo box structured document tag defined as follows: `<w:sdt>`

Attributes	Description

	`  <w:sdtPr>`
	`    <w:dataBinding … />`
	`    <w:comboBox w:lastValue="2"/>`
	`  </w:sdtPr>`
	`  <w:sdtContent>`
	`    <w:r>`
	`      <w:t>Hello, world</w:t>`
	`    </w:r>`
	`  </w:sdtContent>`
	`</w:sdt>`
	
	The current run content of the structured document tag reads `Hello, world`. When this document is opened, if the current value of the associated custom XML data is 2, the matching lastValue attribute specifies that the contents of the combo box must continue to be the current display text of the combo boxeven though there is no listItem whose value is 2 (and normally, the content of the structured document tag would be set to 2. Essentially, this attribute specifies a listItem whose value is 2 and whose displayText is `Hello, world` (the current structured document tag contents). *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtComboBox) is located in §A.1. *end note*]

17.5.2.6 dataBinding (XML Mapping)

This element specifies the information that shall be used to establish a mapping between the nearest ancestorstructured document tag and an XML element stored within a Custom XML Data part in the current WordprocessingML document.

If this element is omitted, then no XML mapping shall be associated with the current structured document tag. If the nearest ancestorstructured document tag is of type rich text or document part gallery, then this property shall be ignored.

If this element is present and the nearest ancestorstructured document tag is not of a rich text type, then the current value of the structured document tag shall be determined by finding the XML element (if any) which is determined by the attributes on this element. If this information does not result in an XML element, then the application can use any algorithm desired to find the closest available match. If this information does result in an XML element, then the contents of that element shall be used to replace the current run content within the document.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:dataBinding w:xpath="/root/name/first" … />
    <w:text/>
  </w:sdtPr>
  <w:sdtContent>
    <w:r>
      <w:t>old text</w:t>
    </w:r>
  </w:sdtContent>
</w:sdt>
```

This structured document tag specifies that it contains only plain text via the text element, and that it must be mapped to the element in the first Custom XML Data part which contains an element that matches the XPath expression `/root/name/first`. When that element is located, its contents must replace the existing run content in the document (for example, if its contents are `new text`, then the contents of the run for this structured document tag must be `new text` when the document is displayed. *end example*]

Attributes	Description

prefixMappings (XML Namespace Prefix Mappings)	Specifies the set of prefix mappings which shall be used to interpret the XPath expression specified on the xpath attribute when the XPath expression is evaluated against the custom XML data parts in the current document.
	This attribute's value shall be specified using the following syntax: `xmlns:prefix='namespace'`, where `prefix` is the namespace prefix to be mapped, and `namespace` is the namespace to be mapped to the current prefix. Each prefix mapping shall be delimited by one or more whitespace characters in the attribute's contents.
	If this attribute is omitted, then the prefix mappings specified on each of the custom XML data parts itself shall be used to evaluate the given XPath expression.
	[*Example*: Consider the following structured document tag properties:
	`<w:sdtPr>`
	`  <w:dataBinding w:xpath="//ns0:book"`
	`w:prefixMappings="xmlns:ns0=`
	`'http://example.com/example'"/>`
	`  <w:text/>`
	`</w:sdtPr>`
	This structured document tag specifies that it contains an XML mapping, and that mapping's prefixMapping attribute must signify that the set of namespace prefix mappings to be used to evaluate the xpath attribute value must be `xmlns:ns0='http://example.com/example'`. *end example*]

©ISO/IEC 2012 – All rights reserved

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).</td></tr>
<tr><td>storeItemID (Custom XML Data Storage ID)</td><td>Specifies the custom XML data identifier for the custom XML data part which shall be used to evaluate the given XPath expression. The <i>custom XML data identifier</i>, specified using the storeItemID attribute of the dataStoreItem element (§22.5.2.1) on the Custom XML Data Properties part is a string that uniquely identifies a particular custom XML data part in a WordprocessingML document (as multiple parts can have the same namespace for their root element).

If specified, then the XPath expression specified on the xpath attribute shall only be evaluated against the custom XML data part whose properties part has a matching custom XML data identifier. If no custom XML data part exists with a matching identifier, then the XML mapping shall not be connected.

If omitted, then the XPath expression shall be evaluated against each custom XML data part in turn until the given XPath expression is resolved to an XML element.

[<i>Example</i>: Consider the following structured document tag properties:<pre>
<w:sdtPr>
  <w:dataBinding w:xpath="//ns0:book"
w:storeItemID="testXmlPart"
/&gt;
  <w:text/>
</w:sdtPr>
</pre>This structured document tag specifies that it contains an XML mapping, and that mapping must only be evaluated against the custom XML part whose identifier is equal to testXmlPart (if one exists). <i>end example</i>]

The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).</td></tr>
<tr><td>xpath (XPath)</td><td>Specifies the XPath expression which shall be evaluated to find the custom XML node which is mapped to the nearest ancestorstructured document tag. This XPath expression shall be specified using the syntax defined in the XML Path Language (XPath) Version 1.0 specification (see Annex A for bibliographic reference information).

[<i>Example</i>: Consider the following structured document tag properties:<pre>
<w:sdtPr>
  <w:dataBinding w:xpath="//ns0:book"
w:prefixMappings="xmlns:ns0=
'http://example.com/example'"/>
  <w:text/>
</w:sdtPr>
</pre></td></tr>
</table>

Attributes	Description
	This structured document tag specifies that it contains an XML mapping, and that mapping's xpath attribute must signify that the XPath expression to be evaluated must be //ns0:book. Because the prefixMapping attribute is also specified, those prefix mappings must be used to evaluate this XPath expression. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_DataBinding) is located in §A.1. *end note*]

17.5.2.7 date (Date Structured Document Tag)

This element specifies that the nearest ancestorstructured document tag shall be a date picker when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- The child elements of this element specify how the dates in this structured document tag shall be stored in any mapped custom XML data and displayed in the document
- Formatting applied to any part of this structured document tag's contents shall apply to its entire contents

As well, the structured document tag shall satisfy the following restraints or the document shall be considered non-conformant:

- The contents shall only be contain a single run (one set of formatting properties)
- The contents shall not contain more than a single paragraph or table cell and shall not contain a table row or table cell

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:date>
      …
    </w:date>
  </w:sdtPr>
  …
</w:sdt>
```

The date element in this structured document tag's properties specifies that the type of structured document tag is a date picker. *end example*]

Attributes	Description

fullDate (Last Known Date in XML Schema DateTime Format)	Specifies the full date and time last entered into the nearest ancestorstructured document tag using the standard XML Schema DateTime syntax.
	[*Note*: This cache is used because the date display mask stored on the dateFormat element (§17.5.2.8) might not contain all of the information about the date, which might be needed if the date display mask is later changed. *end note*]
	If this attribute is specified, then the current fullDate attribute shall be used to populate the run content of the nearest ancestorstructured document tag by filtering it through the date display mask specified in the dateFormat element, if one is present.
	If this attribute is omitted, then the current display text shall be maintained when the document is displayed.
	[*Example*: Consider the following structured document tag properties:
	`<w:sdtPr>`
	`  …`
	`  <w:date w:fullDate="2006-01-01T05:30:00Z">`
	`    …`
	`  </w:date>`
	`</w:sdtPr>`
	The full XML Schema DateTime format for the current structured document tag is specified via the fullDate attribute value as `2006-01-01T05:30:00Z`. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtDate) is located in §A.1. *end note*]

17.5.2.8 dateFormat (Date Display Mask)

The element specifies the display format which shall be used to format any date entered into the nearest ancestorstructured document tag in full DateTime format [*Example*: Through a user interface (a date picker), or through custom XML data associated with this structured document tag via the dataBinding element (§17.5.2.6). *end example*] before displaying it in the structured document tag's run content.

If this element is omitted, then the date shall be formatted using the standard date display mask for the language ID specified on the lid element (§17.5.2.20) if present, or the language ID of the run contents otherwise.

The date display mask specified in the val attribute shall be interpreted using the semantics specified in §17.16.4.1.

[*Example*: Consider the following structured document tag properties:

```
<w:sdtPr>
  <w:date w:fullDate="2006-01-01T06:30:00Z">
    <w:dateFormat w:val="MM-YYYY"/>
  </w:date>
</w:sdtPr>
```

The full XML Schema DateTime format for the current structured document tag is specified via the fullDate attribute value as `2006-01-01T06:30:00Z`, and the date display mask is `MM-YYYY`, therefore the resulting date displayed in the document must be `01-2006` (the month and long year from the full date value, respectively). *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestorstructured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.5.2.9 docPart (Document Part Reference)

This element specifies the name of the document part that shall be displayed in the nearest ancestorstructured document tag when its run contents are empty. If this element is specified, then a document part whose name element (§17.12.12) specifies a name matching the value of this element, and which belongs to the bbPlcHdr style shall be located to be used as the placeholder text for the nearest ancestorstructured document tag.

If no document part is located matching the criteria specified by this element, then five non-breaking spaces shall be used as the default placeholder text.

[*Example*: Consider a structured document tag defined as follows:

```
<w:sdt>
  <w:sdtPr>
    <w:placeholder>
      <w:docPart w:val="DefaultPlaceholder_22610170" />
    </w:placeholder>
    …
  </w:sdtPr>
  <w:sdtContent>
  …
  </w:sdtContent>
</w:sdt>
```

This structured document tag specifies through the docPart element that its placeholder text must be specified in the document part of style bbPlcHdr whose name is equal to DefaultPlaceholder_22610170. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's styleId.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`

Attributes	Description
	In this case, the decimal number in the val attribute is the caption of the nearest ancestorstructured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.5.2.10 docPartCategory (Document Part Category Filter)

This element specifies the category of document parts that shall be used as the filter when determining the possible choices of document parts that are displayed for insertion into the nearest ancestorstructured document tag. A document part *category* is a sub-classification within a given document part gallery which can be used to further categorize the parts in a given gallery. [*Example*: Gallery `custom1` might have categories of `Legal Clauses`, `Conformance Clauses`, etc. *end example*]. The category which shall be used as a filter is stored in this element's val attribute.

If this element is omitted, then the nearest ancestorstructured document tag shall display all document parts in the specified gallery regardless their specified category. If this element is present, but no document parts of the specified gallery and category combination are located by the application, then no document parts shall be displayed (i.e. the application shall not fall back to showing document parts in all categories in the specified gallery).

[*Example*: Consider the following properties for a structured document tag:

```
<w:sdtPr>
  <w:docPartList>
    <w:docPartGallery w:val="custom1"/>
    <w:docPartCategory w:val="Legal Clauses"/>
  </w:docPartList>
</w:sdtPr>
```

This structured document tag specifies that it must present a selection of document parts for insertion via the docPartList element (§17.5.2.12) , and those document parts must only be the parts which are in the `custom1` gallery via the docPartType element (§17.5.2.11), and within that gallery, only the document parts which are in a category called `Legal Clauses` via this element. *end example*]

Attributes	Description
val (String Value)	Specifies that its contents contain a string. The contents of this string are interpreted based on the context of the parent XML

Attributes	Description

	element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestorstructured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_String) is located in §A.1. *end note*]

17.5.2.11 docPartGallery (Document Part Gallery Filter)

This element specifies the gallery of document parts that shall be used as the filter when determining the possible choices of document parts that are displayed for insertion into the nearest ancestorstructured document tag. A document part *gallery* is a classification of document parts, which might then be subdivided into categories. [*Example*: A gallery with a name of `custom1` might have categories of `Legal Clauses`, `Conformance Clauses`, etc. *end example*]. The gallery which shall be used is stored in this element's val attribute.

If this element is omitted, then the nearest ancestorstructured document tag shall display all document parts in its default gallery. If this element is present, but no document parts of the specified gallery are located by the application, then document parts in the default gallery hall be displayed (i.e. the application shall behave as if the value was omitted).

[*Example*: Consider the following properties for a structured document tag:

```
<w:sdtPr>
  <w:docPartList>
    <w:docPartGallery w:val="custom1"/>
  </w:docPartList>
</w:sdtPr>
```

This structured document tag specifies that it must present a selection of document parts for insertion via the docPartList element (§17.5.2.12), and those document parts must only be the parts which are in the `custom1` gallery via this element. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the nearest ancestorstructured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.5.2.12 docPartList (Document Part Gallery Structured Document Tag)

This element specifies that the nearest ancestorstructured document tag shall be of a document part gallery type.

©ISO/IEC 2012 – All rights reserved

This setting does not require or imply that the contents of the structured document tag shall contain only the exact contents of a document part of the specified gallery and category which is present on the current machine, it shall only be used to specify that the structured document tag is of this kind, which shall be used by an application to present the possible list of choices for insertion into the nearest ancestorstructured document tag.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:docPartList>
      …
    </w:docPartList>
  </w:sdtPr>
  …
</w:sdt>
```

The docPartList element in this structured document tag's properties specifies that the type of structured document tag is a document part gallery. The child elements must specify the gallery and category filters for this list, if any. *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_SdtDocPart) is located in §A.1. *end note*]

17.5.2.13 docPartObj (Built-In Document Part Structured Document Tag)

This element specifies that the nearest ancestor structured document tag shall be of a document part type.

This setting does not require or imply that the contents of the structured document tag shall contain only the exact contents of a document part of the specified gallery and category which is present on the current machine, it shall only be used to specify that the structured document tag is of this kind, which shall be used by an application to present the possible list of choices for insertion into the nearest ancestor structured document tag.

This element differs from the docPartList element (§17.5.2.12) in that it can be used to semantically tag a set of block-level objects in a WordprocessingML document without requiring the ability to specify a category and gallery of objects which can be swapped with it via the user interface.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:docPartObj>
      …
    </w:docPartObj>
  </w:sdtPr>
  …
</w:sdt>
```

The docPartObj element in this structured document tag's properties specify that the type of structured document tag is a document part. The child elements must specify the gallery and category semantics for this part, if any. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtDocPart) is located in §A.1. *end note*]

17.5.2.14 docPartUnique (Built-In Document Part)

This element specifies that this structured document tag is being used to encapsulate a built-in document part (i.e. this element appears as a child element of the docPartObj element).

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:docPartObj>
      …
      <w:docPartUnique/>
    </w:docPartObj>
  </w:sdtPr>
  …
</w:sdt>
```

The docPartUnique element in this structured document tag's properties specify that the type of structured document tag is a container for a document part. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.5.2.15 dropDownList (Drop-Down List Structured Document Tag)

This element specifies that the nearest ancestor structured document tag shall be a drop-down list when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- The contents shall not be editable when displayed by a hosting application regardless of the locking settings
- The child elements of this element specify choices which shall be displayed in a standard drop-down list format

As well, the structured document tag shall satisfy the following restraints or the document shall be considered non-conformant:

- The contents shall only be contain a single run (one set of formatting properties)
- The contents shall not contain more than a single paragraph or table cell and shall not contain a table row or table

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:dropDownList>
      …
    </w:dropDownList>
  </w:sdtPr>
  …
</w:sdt>
```

The dropDownList element in this structured document tag's properties specify that the type of structured document tag is a drop-down list. *end example*]

Attributes	Description
lastValue (Drop-down List Last Saved Value)	Specifies the value associated with the current display text for the drop-down list structured document tag. If this structured document tag is not mapped to XML using the dataBinding element (§17.5.2.6), then this attribute shall be ignored. If this structured document tag is mapped to XML, it shall be used to determine whether the current display text in the combo box structured document tag shall be retained when the document is opened, as follows: • When the XML mapping is created, the content in the custom XML data is retrieved • If this content has an associated list item (matching its value attribute), then the corresponding display text shall be displayed in the structured document tag • If no list item exists, this content shall be matched against the lastValue attribute value. If the values match, the current display text shall be retained. If the values do not match, the current custom XML data content shall be the new display text (since no match exists in the combo box list items) [*Example*: Consider a drop-down list structured document tag defined as follows:

Attributes	Description

	`<w:sdt>`
	`  <w:sdtPr>`
	`    <w:dataBinding … />`
	`    <w:dropDownList w:lastValue="2"/>`
	`  </w:sdtPr>`
	`  <w:sdtContent>`
	`    <w:r>`
	`      <w:t>Hello, world</w:t>`
	`    </w:r>`
	`  </w:sdtContent>`
	`</w:sdt>`
	
	The current run content of the structured document tag reads `Hello, world`. When this document is opened, if the current value of the associated custom XML data is 2, the matching lastValue attribute specifies that the contents of the combo box must continue to be the current display text of the combo boxeven though there is no listItem whose value is 2 (and normally, the content of the structured document tag would be set to 2. Essentially, this attribute specifies a listItem whose value is 2 and whose displayText is `Hello, world` (the current structured document tag contents). *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtDropDownList) is located in §A.1. *end note*]

17.5.2.16 equation (Equation Structured Document Tag)

This element specifies that the nearest ancestor structured document tag shall be of type `equation`.

This setting does not require or imply that the contents of the structured document tag shall contain only an equation or associated placeholder text, it shall only be used to specify that the structured document tag is of this kind, which can be used by an application as desired.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:equation/>
  </w:sdtPr>
  …
</w:sdt>
```

The equation element in this structured document tag's properties specify that the type of structured document tag is `equation`. *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Empty) is located in §A.1. *end note*]

17.5.2.17 group (Group Structured Document Tag)

This element specifies that the nearest ancestor structured document tag shall be a restricted grouping when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- The contents of this structured document tag shall not be editable when displayed by a hosting application regardless of the locking settings. This restriction can be superseded by any structured document tag contained within the group, as each structured document tag specifies the locking properties for its own content.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:group/>
  </w:sdtPr>
  …
</w:sdt>
```

The group element in this structured document tag's properties specify that the type of structured document tag is a restricted group. *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Empty) is located in §A.1. *end note*]

17.5.2.18 id (Unique ID)

This element specifies a unique numerical ID for the nearest ancestor structured document tag. This ID shall be persisted through multiple sessions (i.e. shall not be changed once specified).

If multiple structured document tags specify the same decimal number value for the id attribute, then the first structured document tag in the document shall maintain this original ID, and all subsequent structured document tags shall have new identifiers assigned to them when the document is opened.

If this element is omitted, then the nearest ancestor structured document tag shall have a new unique identifier assigned to it when the document is opened.

[*Example*: Consider the following structured document tag properties:

```
<w:sdtPr>
  <w:id w:val="8775518"/>
  …
</w:sdtPr>
```

This set of properties specifies via the val attribute on the id element that the ID for the parent structured document must be 8775518 (subject, of course, to the conflict management and resolution discussed above). *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.5.2.19 label (Structured Document Tag Label)

This element specifies the label identifier associated with the current structured document tag. The identifier representing the label shall be stored on this element's val attribute and is used to reference the unique identifier value of a structured document tag. The contents of the structured document tag resolved by a specific unique identifier shall be used as the label content for the structured document tag that references that specific unique identifier of the structured document tag. If multiple instances of the label element are present, the labels referenced are ordered from most general to most specific. [*Example*: A form element for specifying country name might reference the label for these three items (in order): "Sender", "Home Address", and "Country". *end example*]

If this element is omitted or the value of the label identifier cannot be resolved, then no label shall be associated with the given structured document tag.

[*Example*: Consider the following two structured document tags where one structured document tag references another structured document tag as a label:

```
<w:sdt>
  <w:sdtPr>
    <w:id w:val="5" />
  </w:sdtPr>
  <w:sdtContent>
    <w:p>
      <w:r>
        <w:t>Name</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>
…
<w:sdt>
  <w:sdtPr>
    <w:id w:val="6" />
    <w:label w:val="5" />
  </w:sdtPr>
  …
</w:sdt>
```

The label element specifies that the structured document tag with an identifier value of 6 uses the contents of the structured document tag with an identifier value of 5 as a label source. In this example, the label contents are “Name”. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.5.2.20 lid (Date Picker Language ID)

This element specifies the language ID that shall be used for displaying a calendar for the current date picker structured document tag, if a user interface is present for the structured document tag.

If this element is omitted, then the language ID shall be the language ID of the run contents of the nearest ancestor structured document tag.

[*Example*: Consider the following structured document tag properties:

```
<w:sdtPr>
  <w:date w:fullDate="2006-01-01T06:30:00Z">
    <w:lid w:val="ja-JP"/>
  </w:date>
</w:sdtPr>
```

The calendar language ID for a calendar which can be displayed in the document must be the default calendar format for the Japanese (Japan) language format (`ja-JP`). *end example*]

Attributes	Description
val (Language Code)	Specifies an identifier for a specific language. This code is interpreted in the context of the parent XML element. [*Example*: Consider an object which must specify the English(Canada) language. That object would use an identifier of en-CA to specify this language. *end example*] The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Lang) is located in §A.1. *end note*]

17.5.2.21 listItem (Combo Box List Item)

This element specifies a single list item within the parent combo box structured document tag. Each list item shall be displayed in the list displayed for the nearest ancestor structured document tag (if a user interface is present).

[*Example*: Consider the following combo box structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:comboBox>
      <w:listItem w:displayText="Zero" w:value="0"/>
      <w:listItem w:displayText="One" w:value="1"/>
    </w:comboBox>
  </w:sdtPr>
  …
</w:sdt>
```

Each listItem element within the comboBox element specifies a single list item entry, in this case resulting in two list items within the parent combo box structured document tag. *end example*]

Attributes	Description

displayText (List Entry Display Text)	Specifies the text to display in the run content (as well as any supplied user interface) in place of the value attribute contents for this drop-down list entry.
	This value shall be used as follows:
	• If the nearest ancestor structured document tag is mapped to a custom XML element, the value in that custom XML element shall be mapped the content of the value attribute, and the resulting displayText attribute value (if one is present) shall be displayed in the run content. If no displayText attribute is present, then the value shall be displayed.
	• If the corresponding entry is selected via a user interface, this value shall be stored in the parent element's run content in the document (this is the value that shall be shown in the document's WordprocessingML content).
	If this attribute is omitted, then the content of the value attribute shall be used as the display text for the current list item entry.
	[*Example*: Consider the following drop-down list structured document tag:
	`<w:sdt>`
	`  <w:sdtPr>`
	`    <w:dropDownList>`
	`      <w:listItem w:displayText="The Letter A" w:value="a"/>`
	`      <w:listItem w:displayText="The Letter B" w:value="b"/>`
	`    </w:dropDownList>`
	`  </w:sdtPr>`
	`  …`
	`</w:sdt>`
	The displayText attribute for the first entry is `The Letter A` and the second is `The Letter B`, therefore, these values are used to determine the display text if the nearest ancestor structured document tag is mapped to custom XML data in a custom XML data part. *end example*]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).</td></tr>
<tr><td>value (List Entry Value)</td><td>

Specifies the value for the current list item entry.

This value shall be used as follows:

- If the nearest ancestor structured document tag is mapped to a custom XML element, the value in that custom XML element shall be mapped to this value, and the resulting displayText attribute value (if one is present) shall be displayed in the run content. If no displayText attribute is present, then the value shall be displayed.
- If the corresponding entry is selected via a user interface, this value shall be stored in the parent element's listItem attribute value.

[*Example*: Consider the following combo box structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:comboBox>
      <w:listItem w:displayText="Zero" w:value="0"/>
      <w:listItem w:displayText="One" w:value="1"/>
    </w:comboBox>
  </w:sdtPr>
  …
</w:sdt>
```

The value attribute for the first entry is `0` and the second is `1`, therefore, these values are used to determine the display text if the nearest ancestor structured document tag is mapped to custom XML data in a custom XML data part. *end example*]

The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

</td></tr>
</table>

[*Note*: The W3C XML Schema definition of this element’s content model (CT_SdtListItem) is located in §A.1. *end note*]

17.5.2.22 listItem (Drop-Down List Item)

This element specifies a single list item within the parent drop-down list structured document tag. Each list item shall be displayed in the list displayed for the nearest ancestor structured document tag (if a user interface is present).

[*Example*: Consider the following combo box structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:dropDownList>
      <w:listItem w:displayText="The Letter A" w:value="a"/>
      <w:listItem w:displayText="The Letter B" w:value="b"/>
    </w:dropDownList>
  </w:sdtPr>
  …
</w:sdt>
```

Each listItem element within the dropDownList element specifies a single list item entry, in this case resulting in two list items within the parent drop-down list structured document tag. *end example*]

Attributes	Description

displayText (List Entry Display Text)	Specifies the text to display in the run content (as well as any supplied user interface) in place of the value attribute contents for this drop-down list entry.
	This value shall be used as follows:
	• If the nearest ancestor structured document tag is mapped to a custom XML element, the value in that custom XML element shall be mapped the content of the value attribute, and the resulting displayText attribute value (if one is present) shall be displayed in the run content. If no displayText attribute is present, then the value shall be displayed.
	• If the corresponding entry is selected via a user interface, this value shall be stored in the parent element's run content in the document (this is the value that shall be shown in the document's WordprocessingML content).
	If this attribute is omitted, then the content of the value attribute shall be used as the display text for the current list item entry.
	[*Example*: Consider the following drop-down list structured document tag:
	`<w:sdt>`
	`  <w:sdtPr>`
	`    <w:dropDownList>`
	`      <w:listItem w:displayText="The Letter A" w:value="a"/>`
	`      <w:listItem w:displayText="The Letter B" w:value="b"/>`
	`    </w:dropDownList>`
	`  </w:sdtPr>`
	`  …`
	`</w:sdt>`
	The displayText attribute for the first entry is `The Letter A` and the second is `The Letter B`, therefore, these values are used to determine the display text if the nearest ancestor structured document tag is mapped to custom XML data in a custom XML data part. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

value (List Entry Value)	Specifies the value for the current list item entry.
	This value shall be used as follows:
	• If the nearest ancestor structured document tag is mapped to a custom XML element, the value in that custom XML element shall be mapped to this value, and the resulting displayText attribute value (if one is present) shall be displayed in the run content. If no displayText attribute is present, then the value shall be displayed.
	• If the corresponding entry is selected via a user interface, this value shall be stored in the parent element's listItem attribute value.
	[*Example*: Consider the following combo box structured document tag:
	`<w:sdt>`
	`  <w:sdtPr>`
	`    <w:comboBox>`
	`      <w:listItem w:displayText="Zero" w:value="0"/>`
	`      <w:listItem w:displayText="One" w:value="1"/>`
	`    </w:comboBox>`
	`  </w:sdtPr>`
	`  …`
	`</w:sdt>`
	The value attribute for the first entry is `0` and the second is `1`, therefore, these values are used to determine the display text if the nearest ancestor structured document tag is mapped to custom XML data in a custom XML data part. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtListItem) is located in §A.1. *end note*]

17.5.2.23 lock (Locking Setting)

This element specifies the set of behaviors that shall be applied to the contents of the nearest ancestorstructured document tag when the contents of this documents are edited by an application (whether through a user interface or directly). The type of locking applied to the structured document tag is specified via the value of the associated val attribute.

If this element is omitted, then the locking settings implied for the structured document tag shall be as follows:

- If the structured document tag specifies that it is a group via the group element (§17.5.2.17), then the contents of the structured document tag shall be editable, but the entire tag can be deleted.
- For all other kinds, no locking settings shall be applied to the structured document tag.

[*Example*: Consider the following plain text structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:lock w:val="sdtLocked"/>
    …
    <w:text/>
  </w:sdtPr>
  …
</w:sdt>
```

This plain text structured document tag's properties contain a lock element, specifying locking behaviors for the structured document tag. Since the locking val attribute value is sdtLocked, this locking setting must specify that the contents of the structured document tag can be edited, but the structured document tag itself must not be deleted from the document. *end example*]

Attributes	Description

val (Locking Type)	Specifies the type of locking which shall be applied to the nearest ancestor structured document tag.
	If this attribute is omitted, this its value shall be assumed to be unlocked (using the defaults stated above).
	[*Example*: Consider the following plain text structured document tag properties:
	`<w:sdtPr>`
	`  <w:lock w:val="contentLocked"/>`
	`  …`
	`  <w:text/>`
	`</w:sdtPr>`
	The val attribute value is contentLocked, therefore this locking setting shall specify that the contents of the structured document tag must not be edited, but the structured document tag itself can be deleted from the document. *end example*]
	The possible values for this attribute are defined by the ST_Lock simple type (§17.18.49).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Lock) is located in §A.1. *end note*]

17.5.2.24 picture (Picture Structured Document Tag)

This element specifies that the nearest ancestor structured document tag shall be a picture when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- The contents shall always be restricted to a single picture using the DrawingML (§20.1) syntax

As well, the structured document tag shall satisfy the following restraints or the document shall be considered non-conformant:

- The contents shall only be a single picture using the DrawingML (§20.1) syntax
- The contents shall not contain more than a single paragraph or table cell and shall not contain a table row or table

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:picture/>
  </w:sdtPr>
  …
</w:sdt>
```

The text element in this structured document tag's properties specify that the type of structured document tag is a picture. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.5.2.25 placeholder (Structured Document Tag Placeholder Text)

This element specifies the placeholder text which should be displayed when this structured document tag's run contents are empty, the associated mapped XML element is empty as specified via the dataBinding element (§17.5.2.6) or the showingPlcHdr element (§17.5.2.39) is set in the structured document tag's properties. The placeholder text which shall be shown is itself specified via the child element docPart.

If this element is omitted, then five non-breaking spaces shall be used as the default placeholder text for this structured document tag.

[*Example*: Consider a structured document tag defined as follows:

```
<w:sdt>
  <w:sdtPr>
    <w:placeholder>
      <w:docPart w:val="DefaultPlaceholder_22610170" />
    </w:placeholder>
    …
  </w:sdtPr>
  <w:sdtContent>
    …
  </w:sdtContent>
</w:sdt>
```

This structured document tag specifies through the placeholder element that its placeholder text must be specified in the document part of type bbPlcHdr whose name is equal to DefaultPlaceholder_22610170. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Placeholder) is located in §A.1. *end note*]

17.5.2.26 richText (Rich Text Structured Document Tag)

This element specifies that the nearest ancestor structured document tag shall be a rich text box when displayed in the document.

If no type element (the xsd:choice block in the XML Schema fragment for the parent sdtPr element) is specified, then the nearest ancestor structured document tag shall be of type richText.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:richText/>
  </w:sdtPr>
  …
</w:sdt>
```

The richText element in this structured document tag's properties specify that the type of structured document tag is a rich text box. *end example*].

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.5.2.27 rPr (Run Properties For Structured Document Tag Contents)

This element specifies the set of run properties that shall be applied to the text entered into the nearest ancestor structured document tag in replacement of placeholder text. When placeholder text is present in a structured document tag, its formatting is often different than the desired underlying formatting, and this element specifies the formatting which shall be used for non-placeholder text contents when they are initially added to the control.

If this element is not present, the inserted is unformatted, as with any other run of text - it shall not inherit the properties of the placeholder text.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:placeholder>
      <w:docPart w:val="TestPlaceholderDocPart"/>
    </w:placeholder>
    <w:showingPlcHdr/>
    <w:rPr>
      <w:rStyle w:val="UserName"/>
    </w:rPr>
    …
  </w:sdtPr>
  <w:sdtContent>
    <w:r>
      <w:rPr>
        <w:rStyle w:val="PlaceholderText"/>
      </w:rPr>
      <w:t>[Type Your Name Here]</w:t>
    </w:r>
  </w:sdtContent>
</w:sdt>
```

This structured document tag specifies that its current contents are placeholder text via the showingPlcHdr element (§17.5.2.39), and that text has the `PlaceholderText` character style applied to it.

Now, assume that that style created grey shaded text (typical for placeholder text). This formatting would clearly not be desirable for any text entered into the structured document tag. Therefore, when this text is added, the rPr element in the sdtPr is used to store the formatting on the resulting text.

In this example, the text which initially populates the control shall be formatted with the `UserName` character style. *end example*]

The W3C XML Schema definition of this element's content model (CT_RPr) is located in §A.1. Each child element from the above table shall not occur more than once. [*Note*: This restriction is not reflected in the element's content model due to limitations of W3C XML Schema language. *end note*]

©ISO/IEC 2012 – All rights reserved

17.5.2.28 rPr (Structured Document Tag End Character Run Properties)

This element specifies the set of run properties which shall be applied to the character present to delimit the end of the structured document tag's contents. When these properties are applied, they shall be applied in addition to the run properties specified for the entire structured document tag via the rPr element (§17.5.2.27) stored in the tag's main property container.

If this element is not present, the inserted closing tag shall be formatting identically to the start tag.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:placeholder>
      <w:docPart w:val="TestPlaceholderDocPart"/>
    </w:placeholder>
    <w:showingPlcHdr/>
    <w:rPr>
      <w:rStyle w:val="UserName"/>
    </w:rPr>
    …
  </w:sdtPr>
  <w:sdtEndPr>
    <w:rPr>
      <w:b/>
      <w:i/>
    </w:rPr>
  </w:sdtEndPr>
  <w:sdtContent>
    …
  </w:sdtContent>
</w:sdt>
```

The rPr elements under the tag's properties specify that this structured document tag specifies that its start character must have formatting in the character style `UserName`, and that the end character must have the formatting in the character style `UserName` as well as bold and italic direct formatting. *end example*]

The W3C XML Schema definition of this element’s content model (CT_RPr) is located in §A.1. Each child element from the above table shall not occur more than once. [*Note*: This restriction is not reflected in the element's content model due to limitations of W3C XML Schema language. *end note*]

17.5.2.29 sdt (Block-Level Structured Document Tag)

This element specifies the presence of a structured document tag around one or more block-level structures (paragraphs, tables, etc.). The two child elements of this element shall be used to specify the properties and content of the current structured document tag via the sdtPr and sdtContent elements, respectively.

[*Example*: Consider a structured document tag with the friendly name address that must be located around a single paragraph in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:body>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="address"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:p>
        …
      </w:p>
    </w:sdtContent>
  </w:sdt>
  …
</w:body>
```

The sdt element specifies the structured document tag, the child sdtPr element contains the friendly name property set to address, and the sdtContent element contains a single paragraph (it is a block-level structured document tag). *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_SdtBlock) is located in §A.1. *end note*]

17.5.2.30 sdt (Row-Level Structured Document Tag)

This element specifies the presence of a structured document tag around a single table row. The two child elements of this element shall be used to specify the properties and content of the current structured document tag via the sdtPr and sdtContent elements, respectively.

[*Example*: Consider a structured document tag with the friendly name invoiceItem that must be located around a single table row in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tbl>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="invoiceItem"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:tr>
        …
      </w:tr>
    </w:sdtContent>
  </w:sdt>
  …
</w:tbl>
```

The sdt element specifies the structured document tag, the child sdtPr element contains the friendly name property set to `invoiceItem`, and the sdtContent element contains a single table row (it is a row-level structured document tag). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtRow) is located in §A.1. *end note*]

17.5.2.31 sdt (Inline-Level Structured Document Tag)

This element specifies the presence of a structured document tag around one or more inline-level structures (runs, DrawingML objects, fields, etc.) in the current paragraph. The two child elements of this element shall be used to specify the properties and content of the current structured document tag via the sdtPr and sdtContent elements, respectively.

[*Example*: Consider a structured document tag with the friendly name `firstName` that must be located around two runs in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:p>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="firstName"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:r>
        …
      </w:r>
      <w:r>
        …
      </w:r>
    </w:sdtContent>
  </w:sdt>
  …
</w:p>
```

The sdt element specifies the structured document tag, the child sdtPr element contains the friendly name property set to firstName, and the sdtContent element contains two runs (it is an inline-level structured document tag). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtRun) is located in §A.1. *end note*]

17.5.2.32 sdt (Cell-Level Structured Document Tag)

This element specifies the presence of a structured document tag around a single table cell. The two child elements of this element shall be used to specify the properties and content of the current structured document tag via the sdtPr and sdtContent elements, respectively.

[*Example*: Consider a structured document tag with the friendly name company that must be located around a single table cell in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tr>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="company"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:tc>
        …
      </w:tc>
    </w:sdtContent>
  </w:sdt>
  …
</w:tr>
```

The sdt element specifies the structured document tag, the child sdtPr element contains the friendly name property set to company, and the sdtContent element contains a single table cell (it is a cell-level structured document tag). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtCell) is located in §A.1. *end note*]

17.5.2.33 sdtContent (Cell-Level Structured Document Tag Content)

This element specifies the last known contents of a structured document tag around a single table cell. This element's contents shall be treated as a cache of the contents to be displayed in the structured document tag for the following reasons:

- If the structured document tag specifies an XML mapping via the dataBinding element (§17.5.2.6), changes to the custom XML data part shall be reflected in the structured document tag as needed
- If the contents of the structured document tag are placeholder text via the showingPlcHdr element (§17.5.2.39), then this content can be updated with the placeholder text stored in the Glossary Document part

[*Example*: Consider a structured document tag with the friendly name company that must be located around a single table cell in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tr>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="company"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:tc>
        …
      </w:tc>
    </w:sdtContent>
  </w:sdt>
  …
</w:tr>
```

The sdtContent element contains a single table cell (it is an cell-level structured document tag content container). *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_SdtContentCell) is located in §A.1. *end note*]

17.5.2.34 sdtContent (Block-Level Structured Document Tag Content)

This element specifies the last known contents of a structured document tag around one or more block-level structures (paragraphs, tables, etc.). This element's contents shall be treated as a cache of the contents to be displayed in the structured document tag for the following reasons:

- If the structured document tag specifies an XML mapping via the dataBinding element (§17.5.2.6), changes to the custom XML data part shall be reflected in the structured document tag as needed
- If the contents of the structured document tag are placeholder text via the showingPlcHdr element (§17.5.2.39), then this content can be updated with the placeholder text stored in the Glossary Document part

[*Example*: Consider a structured document tag with the friendly name `address` that must be located around a single paragraph in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:body>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="address"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:p>
        …
      </w:p>
    </w:sdtContent>
  </w:sdt>
  …
</w:body>
```

The sdtContent element contains a single paragraph (it is a block-level structured document tag content container). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtContentBlock) is located in §A.1. *end note*]

17.5.2.35 sdtContent (Row-Level Structured Document Tag Content)

This element specifies the last known contents of a structured document tag around a single table row.

[*Note*: Unlike other types of structured document tags, this type of structure document tag cannot show placeholder text or have mapped XML data, therefore it is never a cache. *end note*]

[*Example*: Consider a structured document tag with the friendly name `invoiceItem` that must be located around a single table row in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:tbl>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="invoiceItem"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:tr>
        …
      </w:tr>
    </w:sdtContent>
  </w:sdt>
  …
</w:tbl>
```

The sdtContent element contains a single table row (it is an row-level structured document tag content container). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtContentRow) is located in §A.1. *end note*]

17.5.2.36 sdtContent (Inline-Level Structured Document Tag Content)

This element specifies the last known contents of a structured document tag around one or more inline-level structures (runs, DrawingML objects, fields, etc.). This element's contents shall be treated as a cache of the contents to be displayed in the structured document tag for the following reasons:

- If the structured document tag specifies an XML mapping via the dataBinding element (§17.5.2.6), changes to the custom XML data part shall be reflected in the structured document tag as needed
- If the contents of the structured document tag are placeholder text via the showingPlcHdr element (§17.5.2.39), then this content can be updated with the placeholder text stored in the Glossary Document part

[*Example*: Consider a structured document tag with the friendly name `firstName` that must be located around two runs in a WordprocessingML document. This requirement would be specified as follows in the WordprocessingML:

```
<w:p>
  <w:sdt>
    <w:sdtPr>
      <w:alias w:val="firstName"/>
    </w:sdtPr>
    <w:sdtContent>
      <w:r>
        …
      </w:r>
      <w:r>
        …
      </w:r>
    </w:sdtContent>
  </w:sdt>
  …
</w:p>
```

The sdtContent element contains two adjacent runs (it is an inline-level structured document tag content container). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtContentRun) is located in §A.1. *end note*]

17.5.2.37 sdtEndPr (Structured Document Tag End Character Properties)

This element specifies the properties which shall be applied to the physical character which delimits the end of a structured document tag.

[*Example*: Consider a structured document tag with the following properties specified for the end tag:

```
<w:sdtEndPr>
  <w:rPr>
    …
  </w:rPr>
</w:sdtEndPr>
```

This structured document tag specifies properties for its end character within the sdtEndPr element. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtEndPr) is located in §A.1. *end note*]

17.5.2.38 sdtPr (Structured Document Tag Properties)

This element specifies the set of properties that shall be applied to the nearest ancestor structured document tag.

[*Example*: Consider a structured document tag with the following properties specified:

```
<w:sdtPr>
  <w:alias w:val="Birthday"/>
  <w:id w:val="8775518"/>
  <w:date>
    <w:dateFormat w:val="M/d/yyyy"/>
    <w:lid w:val="EN-US"/>
  </w:date>
</w:sdtPr>
```

This structured document tag specifies three properties: the a friendly name of `Birthday` via the alias element (§17.5.2.1), a unique ID of `8775518` via the id element (§17.5.2.18), and a structured document tag type of date picker via the date element (§17.5.2.7) which itself has a set of date-specific properties. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtPr) is located in §A.1. *end note*]

17.5.2.39 showingPlcHdr (Current Contents Are Placeholder Text)

This element specifies whether the content of the sdtContent element (§17.5.2.34; §17.5.2.33; §17.5.2.35; §17.5.2.36) for the nearest ancestor structured document tag shall be interpreted to contain placeholder text for this structured document tag (as opposed to regular text contents within the structured document tag). If this element is present and set to `true`, this state shall be resumed (showing placeholder text) upon opening this document.

If this element is omitted, then the structured document tag shall not be interpreted to be showing placeholder text when the document is displayed.

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:showingPlcHdr/>
    …
    <w:richText/>
  </w:sdtPr>
  <w:sdtContent>
    <w:r>
      <w:t>[Type your name here]</w:t>
    </w:r>
 </w:sdtContent>
</w:sdt>
```

This structured document tag has run contents which read `[Type your name here]`, which would typically be interpreted as the current contents of the structured document tag. However, since the showingPlcHdr element has been specified in the structured document tag's properties, this content must instead be interpreted as the placeholder text for the structured document tag. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.5.2.40 storeMappedDataAs (Custom XML Data Date Storage Format)

This element specifies the translation which shall be performed on the displayed date in a date picker structured document tag when the current contents are saved into the associated custom XML data via the dataBinding element (§17.5.2.6).

If this element is omitted, then the value of the associated custom XML element shall be placed into the custom XML data part with no translation.

[*Example*: Consider the following date picker structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:date w:fullDate="01-01-2006T06:30:00Z">
      <w:storeMappedDateAs w:val="text"/>
      …
    </w:date>
  </w:sdtPr>
  <w:sdtContent>
    <w:r>
      <w:t>January 1</w:t>
    </w:r>
  </w:sdtContent>
</w:sdt>
```

The value of the storeMappedDateAs element's attribute value is `text`, therefore the current run contents must be sent to the mapped XML element without any translation (in this case, the value must be `January 1`). *end example*]

Attributes	Description

val (Date Storage Type)	Specifies the date translation which shall be applied to the parent date picker structured document tag.
	
	If this attribute is omitted, this its value shall be assumed to be `text`.
	
	[*Example*: Consider the following date picker structured document tag:
	
	`<w:sdt>`
	`  <w:sdtPr>`
	`    <w:date … >`
	`      <w:storeMappedDateAs w:val="date"/>`
	`      …`
	`    </w:date>`
	`  </w:sdtPr>`
	`  <w:sdtContent>`
	`    <w:r>`
	`      <w:t>January 1</w:t>`
	`    </w:r>`
	`  </w:sdtContent>`
	`</w:sdt>`
	
	The value of the val attribute is `text`, therefore the current run contents must be sent to the mapped XML element after being translated into xsd:date format (in this case, the value must be 01-01-2006). *end example*]
	
	The possible values for this attribute are defined by the ST_SdtDateMappingType simple type (§17.18.76).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtDateMappingType) is located in §A.1. *end note*]

17.5.2.41 tabIndex (Structured Document Tag Navigation Order Index)

This element specifies the position of the current structured document tag in the navigation (tab) order used in the document. The index shall be stored on this element's val attribute and is analogous to the tabIndex attribute in HTML.

Objects that support tab index shall be navigated by consumers in the following order:

- Objects for which the XML specifies a non-zero tabIndex value are navigated first. Navigation proceeds with the element with the lowest resolved value of tabIndex to the element with the highest resolved value of tabIndex.
  - Objects that specify identical resolved values of tabIndex is navigated in the lexical order in which the elements appear in the underlying WordprocessingML.
- Objects for which the XML does not specify an index or objects for which the XML specifies a resolved tabIndex value of 0 are navigated last. These objects are navigated in the lexical order in which they appear in the underlying WordprocessingML.

[*Example*: Consider the following two structured document tags where each structured document tag specifies a tab index:

```
<w:sdt>
  <w:sdtPr>
    <w:id w:val="5" />
    <w:tabIndex w:val="1" />
  </w:sdtPr>
  <w:sdtContent>
    <w:p>
      <w:r>
        <w:t>First Name</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>
…
<w:sdt>
  <w:sdtPr>
    <w:id w:val="6" />
    <w:tabIndex w:val="2" />
  </w:sdtPr>
```

```
    <w:sdtContent>
      <w:p>
        <w:r>
          <w:t>Last Name</w:t>
        </w:r>
      </w:p>
    </w:sdtContent>
  </w:sdt>
```

The tabIndex element specifies that the structured document tag with an identifier value of 5 must be the first content to be reached via tabbing, whereas the structured document tag with an identifier value of 6 must be the second content to be reached via tabbing. *end example*]

Attributes	Description
val (Positive Decimal Number Value)	Specifies that the contents of this attribute contains a positive decimal number. The contents of this positive decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_UnsignedDecimalNumber: `<… w:val="15" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_UnsignedDecimalNumber simple type (§22.9.2.16).

[*Note*: The W3C XML Schema definition of this element's content model (CT_UnsignedDecimalNumber) is located in §A.1. *end note*]

17.5.2.42 tag (Programmatic Tag)

This element specifies a programmatic tag associated with the current structured document tag. A *programmatic tag* is an arbitrary string which applications can associate with a structured document tag in order to identify it without providing a visible friendly name. The string representing the programmatic tag shall be stored on this element's val attribute.

If this element is omitted, then no programmatic tag shall be associated with the given structured document tag.

[*Example*: Consider the following properties on a structured document tag:

```
<w:sdtPr>
  <w:tag w:val="Clause_3246"/>
  …
</w:sdtPr>
```

This set of properties specifies via the tag element that the programmatic tag for the nearest ancestor structured document tag must be `Clause_3246`. This information can then be used as needed by applications. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.5.2.43 temporary (Remove Structured Document Tag When Contents Are Edited)

This element specifies whether the nearest ancestor structured document tag shall be removed from the WordprocessingML document when the its contents are modified.

[*Note*: This setting is primarily intended for creating structured document tags whose sole purpose is one-time placeholder text, and which should not return once replaced with content. *end note*]

If this element is omitted, then the nearest ancestor structured document tag shall not be automatically removed when its contents are modified.

[*Example*: Consider the following plain text structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:temporary/>
    <w:text/>
  </w:sdtPr>
  …
</w:sdt>
```

This plain text structured document tag's properties contain a temporary element, specifying that the structured document tag itself must be deleted from the document whenever its contents are first modified. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.5.2.44 text (Plain Text Structured Document Tag)

This element specifies that the nearest ancestor structured document tag shall be a plain text box when displayed in the document.

This setting specifies that the behavior for this structured document tag shall be as follows:

- Formatting applied to any part of this structured document tag's contents shall apply to its entire contents

As well, the structured document tag shall satisfy the following restraints or the document shall be considered non-conformant:

- The contents shall only be contain a single run (one set of formatting properties) with exceptions for soft carriage returns via the multiLine attribute on this element
- The contents shall not contain more than a single paragraph or table cell and shall not contain a table row or table

[*Example*: Consider the following structured document tag:

```
<w:sdt>
  <w:sdtPr>
    …
    <w:text/>
  </w:sdtPr>
  …
</w:sdt>
```

The text element in this structured document tag's properties specify that the kind of structured document tag is a plain text box. *end example*]

Attributes	Description

multiLine (Allow Soft Line Breaks)	Specifies whether soft line breaks can be added to the contents of this structured document tag when this document is modified. This setting shall not affect the ability of the structured document tag to display existing soft line breaks (which shall be preserved) and shall only affect the ability to add line breaks when the document is modified by an application.
	If this attribute is omitted, then the parent plain text structured document control shall not allow soft line breaks to be added to its contents.
	[*Example*: Consider the following structured document tag:
	`<w:sdt>`
	`  <w:sdtPr>`
	`    …`
	`    <w:text w:multiLine="true"/>`
	`  </w:sdtPr>`
	`  …`
	`</w:sdt>`
	The multiLine attribute on the text element in this structured document tag's properties specify that an application can allow soft line breaks to be added to the run contents of the structured document tag. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SdtText) is located in §A.1. *end note*]

17.6 Sections

WordprocessingML does not natively store the concept of pages, since it is based on paragraphs and runs (which are laid out on to pages by consumers of this content). However, although there is no concept of storing pages in the WordprocessingML format, it is often necessary to store information about a page or group of pages in a document, in order to store information that is to be used to format the pages on which a set of paragraphs appear. In WordprocessingML, this information is stored via the use of *sections*.

In WordprocessingML, *sections* are groupings of paragraphs that have a specific set of properties used to define the pages on which the text appears, as well as other section-level (applying to all paragraphs' appearance) properties.

[*Example*: Consider a document with four paragraphs of text that is to be printed on a page in landscape mode, followed by ten paragraphs of text that are to be printed in portrait mode. This requirement implies information about the page(s) used to lay out each grouping of text—the first four paragraphs could require one page, or ten.

Therefore, rather than try to cache knowledge of the number of pages and their properties (which is likely to become incorrect if the XML is manipulated by a producer that does not understand page layout), this information is stored by breaking the document into two sections, as follows:

```
<w:p>
  …
</w:p>
<w:p>
  …
</w:p>
<w:p>
  …
</w:p>
<w:p>
  <w:sectPr>
    …
    (section one properties go here)
    <w:pgSz … w:orient="landscape" />
    …
  </w:sectPr>
  …
</w:p>
…
<w:p>
  <w:sectPr>
    …
    (section two properties go here)
    <w:pgSz … w:orient="landscape" />
    …
  </w:sectPr>
  …
</w:p>
```

end example]

17.6.1 bidi (Right to Left Section Layout)

This element specifies that this section shall be presented using a right-to-left page direction. This property only affects section-level properties, and does not affect the layout of text within the contents of this section.

[*Example*: Consider a section with the bidi property set as follows:

```
<w:sectPr>
  …
  <w:bidi/>
</w:sectPr>
```

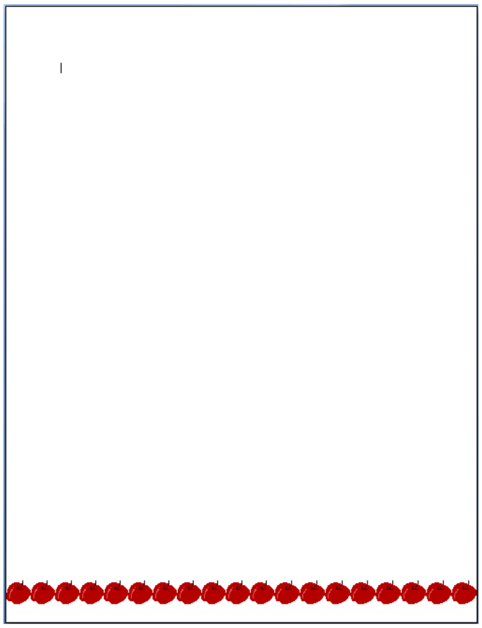
This section direction is now right-to-left, which means that all section level properties are displayed right-to-left (e.g., page numbers are displayed on the right of text; columns are populated from right-to-left). However, the layout of text is determined by properties applied at the text level. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.6.2 bottom (Bottom Border)

This element specifies the presentation and display of the page border displayed at the bottom of each page in this section.

[*Example*: Consider a section in which all pages must have a bottom border consisting of a repeated image of an apple, like this:



This border would result in the following WordprocessingML:

```
<w:sectPr>
  …
  <w:pgBorders>
    <w:bottom w:val="apples" …/>
  </w:pgBorders>
  …
</w:sectPr>
```

Because the page only has a border at the bottom, only the bottom element is specified within the set of page borders. *end example*]

When a document has a bottom border that is relative to the page edges (using the offsetFrom attribute on pgBorders), it shall span the bottom edge of the page at the location defined by its properties, stopping when:

- It intersects with the corresponding left or right page border (if one is specified).
- It reaches the edge of the page.

[*Example*: In the example above, no left or right border was specified in the WordprocessingML, so a consumer must draw the border from one edge of the page to the other. *end example*]

When a document has a bottom border that is relative to the text (using the offsetFrom attribute on pgBorders), it shall span only the necessary width to satisfy the requirement of spanning the width of the text.

When a document has custom border art specified by the bottomLeft, bottomRight, and/or id attributes, it shall use the corresponding relationship part item as an image for the bottom left corner, bottom right corner, and/or bottom border, respectively. If the corresponding relationship part item cannot be located, the consumer shall use the border specified by the value of the val attribute. If the corresponding value of the val attribute cannot be resolved no bottom left corner, bottom right corner, or bottom border is present when the page is displayed.

When a document has a custom border art specified by the id attribute without specifying the bottomRight and/or bottomLeft attributes, the bottom border as resolved by the corresponding relationship part item of the id attribute shall span to the corners not specified by bottomRight and/or bottomLeft attributes.

Attributes	Description
bottomLeft (Custom Defined Bottom Left Border Relationship Reference) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID for the relationship which contains the custom bottom left border image for the parent element. This custom border image is contained in a separate part within the WordprocessingML package. The relationship explicitly targeted by this attribute shall be of type http://purl.oclc.org/ooxml/officeDocument/relationships/image or the document shall be considered non-conformant. If this attribute is omitted, then no custom bottom left border shall be used. [*Example*: Consider the following WordprocessingML markup for a custom bottom left border in a document: `<w:bottom w:val="custom" r:bottomLeft="rIdCustomBottomLeftBorder" …/>` The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rIdCustomBottomLeftBorder` must contain the custom bottom left border image for the document. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

bottomRight (Custom Defined Bottom Right Border Relationship Reference)
Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships
	Specifies the relationship ID for the relationship which contains the custom bottom right border image for the parent element. This custom border image is contained in a separate part within the WordprocessingML package.
	The relationship explicitly targeted by this attribute shall be of type http://purl.oclc.org/ooxml/officeDocument/relationships/image or the document shall be considered non-conformant.
	If this attribute is omitted, then no custom bottom left border shall be used.
	[*Example*: Consider the following WordprocessingML markup for a custom bottom right border in a document:
	`<w:bottom w:val="custom"`
	`r:bottomRight="rIdCustomBottomRightBorder" …/>`
	The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rIdCustomBottomRightBorder` must contain the custom bottom right border image for the document. *end example*]
	The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

color (Border Color)	Specifies the color for this border. This value can be defined as either: • A color value using the RGB color model whose red, green, and blue values are written as numbers in the range 0 to 255, hex encoded, and concatenated. [*Example*: Full intensity red would be 255 red, 0 green, 0 blue, encoded to FF, 00, 00, and concatenated to `FF0000`. *end example*] . RGB colors are specified in the sRGB color space. • `auto` to allow a consumer to automatically determine the border color in order to make the document's text readable. [*Example*: A document with white text and a background color of `auto` might result in the use of a black background, in order to ensure legibility of the content. *end example*] [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore can be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*]

Attributes	Description
	If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§17.18.38).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which must appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" … />` This frame's val is `true`, indicating that the border frame effect must be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
id (Custom Defined Border Relationship Reference) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID for the relationship which contains the custom border image for the parent element. This custom border image is contained in a separate part within the WordprocessingML package. The relationship explicitly targeted by this attribute shall be of type http://purl.oclc.org/ooxml/officeDocument/relationships/image or the document shall be considered non-conformant. If this attribute is omitted, then no custom border shall be used. [*Example*: Consider the following WordprocessingML markup for a custom bottom border in a document: `<w:bottom w:val="custom" r:id="rIdCustomBottomBorder" …/>` The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rIdCustomBottomBorder` must contain the custom bottom border image for the document. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow.

Attributes	Description
	For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which must appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" … />` This frame's val is `true`, indicating that the shadow effect must be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§17.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§17.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24" />`
	`</w:pgBorders`
	The offsetFrom attribute specifies that the space value provides the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset must be 24 points. *end example*]
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§17.18.68).

sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range can be reassigned to a more appropriate value.

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range can be reassigned to a more appropriate value.
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§17.18.23).

themeColor (Border Theme Color)	Specifies the base theme color used to generate the border color. The border color is the RGB value associated with themeColor as further transformed by themeTint or themeShade (if one is present), else the background color is the RGB value associated with themeColor.
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part (§14.2.7 and §20.1.6.9), which allows color information to be set centrally in the document.
	To determine the color to display, the following actions are performed:
	• Using the mapping specified in the ST_ThemeColor simple type (§17.18.97), the appropriate attribute on the clrSchemeMapping element (§17.15.1.20) is read.
	• Using that value and the mapping specified in the ST_ColorSchemeIndex simple type (§17.18.103), the appropriate element in the document's Theme part is read to get the base theme color.
	• The specified color is modified based on the presence of the themeTint or themeShade attribute.
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	`<w:top … w:themeColor="accent2" w:themeTint="99" />`
	`<w:bottom … w:themeColor="accent2" w:themeTint="99" />`
	`<w:left … w:themeColor="accent2" w:themeTint="99" />`
	`<w:right … w:themeColor="accent2" w:themeTint="99" />`
	If the Settings part contained the following markup:

Attributes	Description

	`<w:clrSchemeMapping … w:accent2="accent2"/>`
	and the Theme part contained the following XML markup:
	`<a:accent2>`
	`  <a:srgbClr val="4F81BD"/>`
	`</a:accent2>`
	the resulting border color would be `95B3D7` (the result of a 60% tint applied to the original theme color; see the calculations in themeTint below for details). *end example*]
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§17.18.97).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeColor attribute is not present, then this attribute shall not be used. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $S_{xml} = 0.4 * 255 = 102 = 66(hex)$ The resulting themeShade value in the file format would be `66`. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48,0.53\right)$.

Attributes	Description

	Applying the shade formula with a shade percentage of 75% to the luminance, we get:
	$L' = 0.53 * 0.75$
	$= 0.39698$
	Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48,0.39698\right)$and converting back to RGB, we get 943634.
	This transformed value can be seen in the resulting background's color attribute:
	`<w:top w:val="single" w:sz="4" w:space="24"`
	`  w:color="943634" w:themeColor="accent2"`
	`  w:themeShade="BF"/>`
	end example]
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeColor attribute is not present, then this attribute shall not be used. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $T_{xml} = 0.6 * 255$ $= 153$ $= 99(hex)$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: - Convert the color to the HSL color format (values from 0 to 1) - Modify the luminance factor as follows: $\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$ - Convert the resultant HSL color to RGB

Attributes	Description

	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`.
	The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45,0.53\right)$.
	Applying the tint formula with a tint percentage of 60% to the luminance, we get:
	$L' = 0.53 * 0.6 + (1 - .6)$
	$= 0.71$
	Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45,0.71\right)$and converting back to RGB, we get `95B3D7`.
	This transformed value can be seen in the resulting background's color attribute:
	`<w:top w:val="single" w:sz="4" w:space="24"`
	`w:color="95B3D7" w:themeColor="accent2"`
	`w:themeTint="99"/>`
	end example]
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - shall only be used for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§17.18.2).

[*Note*: The W3C XML Schema definition of this element's content model (CT_BottomPageBorder) is located in §A.1. *end note*]

17.6.3 col (Single Column Definition)

This element specifies the properties for a single column of text within this section.

[*Example*: Consider a single column with a width of two inches, which also has a one-inch space after the column, resulting in the following WordprocessingML:

```
<w:cols … >
  <w:col w:w="2880" w:space="1440"/>
  …
</w:cols>
```

The resulting column specifies its width of 2,880 twentieths of a point and space following of 1,440 twentieths of a point. *end example*]

The contents of the col element are only used to calculate the number and size of columns if the fixedWidth attribute is set to `false` or omitted.

Attributes	Description
space (Space Before Following Column)	Specifies the spacing (in twentieths of a point) between the current column and the next column. [*Example*: Consider a text column that is to have a one-inch space after it. This text column spacing would therefore be 1x72=144 points wide, which translates to 1,440 twentieths of a point. The resulting WordprocessingML specifies that spacing width in twentieths of a point: `<w:col … w:space="1440"/>` *end example*] For the last text column in the section, no spacing is allowed after the column, and, if present, any space value is ignored. The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

w (Column Width)	Specifies the width (in twentieths of a point) of this text column.
	[*Example*: Consider a text column, which is to be two inches wide. This text column would therefore be 2x72=144 points wide, which translates to 2,880 twentieths of a point. The resulting WordprocessingML specifies that column width in twentieths of a point:
	`<w:col … w:w="2880"/>`
	end example]

Attributes	Description
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Column) is located in §A.1. *end note*]

17.6.4 cols (Column Definitions)

This element specifies the set of columns defined for this section in the document.

[*Example*: Consider a document in which a section defines two columns of 4.16" and 1.83", respectively, resulting in the following WordprocessingML:

```
<w:cols w:equalWidth="0">
  <w:col w:w="2640" w:space="720"/>
  <w:col w:w="6000"/>
</w:cols>
```

The cols element defines the set of columns defined for this section, which because equalWidth is `0`, are defined by the number of col elements contained in the column definition. In this case, the first column is `2,640` twentieths of a point wide (as 2640/1440ths of an inch equals 1.83 inches) with one-half of an inch space after, and the second column is `6,000` twentieths of a point wide (4.16 inches). *end example*]

Based on the presence of the equalWidth attribute, a consumer shall render the columns using:

- If equalWidth is `true`, then the columns are defined using the data stored as attributes of the cols element (defined below).
- If equalWidth is `false`, then the columns are defined using the presence and data on each child col element (§17.6.3).

Attributes	Description
equalWidth (Equal Column Widths)	Specifies whether all text columns in the current section are of equal width. If this attribute is present and its value is set to `true` or `1`, then all columns for this text section are of an equal width and are calculated as follows: • Take width of page (from margin to margin) • Divide by number of columns specified in num attribute • For each column, leave space after as defined in the space attribute • Remaining width of each column is the text column width. If this attribute is present and its value is set to `false` or `0`, then all columns for this text section are of different widths and are defined by each col element as follows: • Each col element defines a single column • Each w attribute defines the text column width

Attributes	Description

	- Each space attribute defines the space after the text column
	
	[*Example*: Consider a section with column information defined as follows:
	
	`<w:cols w:num="3" w:space="1440"`
	`  w:equalWidth="1">`
	`  <w:col w:w="2880" w:space="2880" />`
	`  <w:col w:w="2880" w:space="1440" />`
	`  <w:col w:w="2880" />`
	`</w:cols>`
	
	This set of columns has a equalWidth value set to 1, therefore the col elements are ignored, and there are three equally sized columns (num value of 3), each with one inch (space value of 1440 twentieths of a point) of space after. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

num (Number of Equal Width Columns)	Specifies the number of text columns in the current section.
	
	If all columns are not of equal width (the equalWidth attribute is not set), then this element is ignored, and the number of columns is defined by the number of col elements defined under the cols element.
	
	[*Example*: Consider a section with column information defined as follows:
	
	`<w:cols w:num="3" w:space="1440"`
	`  w:equalWidth="1">`
	`  …`
	`</w:cols>`
	
	This set of columns has a equalWidth value set to 1, therefore there are three equally sized columns, as the num attribute has a value of 3. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

sep (Draw Line Between Columns)	Specifies if a vertical line is drawn between each of the text columns in this section.
	
	If set to true or 1, then a vertical line shall be drawn in the center of the spacing between each column in this section.
	
	[*Example*: Consider a section with column information defined as follows:
	
	`<w:cols w:sep="1">`
	`  …`
	`</w:cols>`
	
	This set of columns has a sep value set to 1, therefore there must be a vertical line

Attributes	Description
	separating each column in this section. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

space (Spacing Between Equal Width Columns)	Specifies the spacing between text columns in the current section.
	If all columns are not of equal width (the equalWidth attribute is not set), then this element is ignored, and the spacing after columns is defined by the space attribute on each of the col elements defined under the cols element.
	[*Example*: Consider a section with column information defined as follows:
	`<w:cols w:num="3" w:space="1440"`
	`  w:equalWidth="1">`
	`  …`
	`</w:cols>`
	This set of columns has a equalWidth value set to 1, therefore there are three equally sized columns, each with one inch (space value of 1440 twentieths of a point) of space after. *end example*]
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

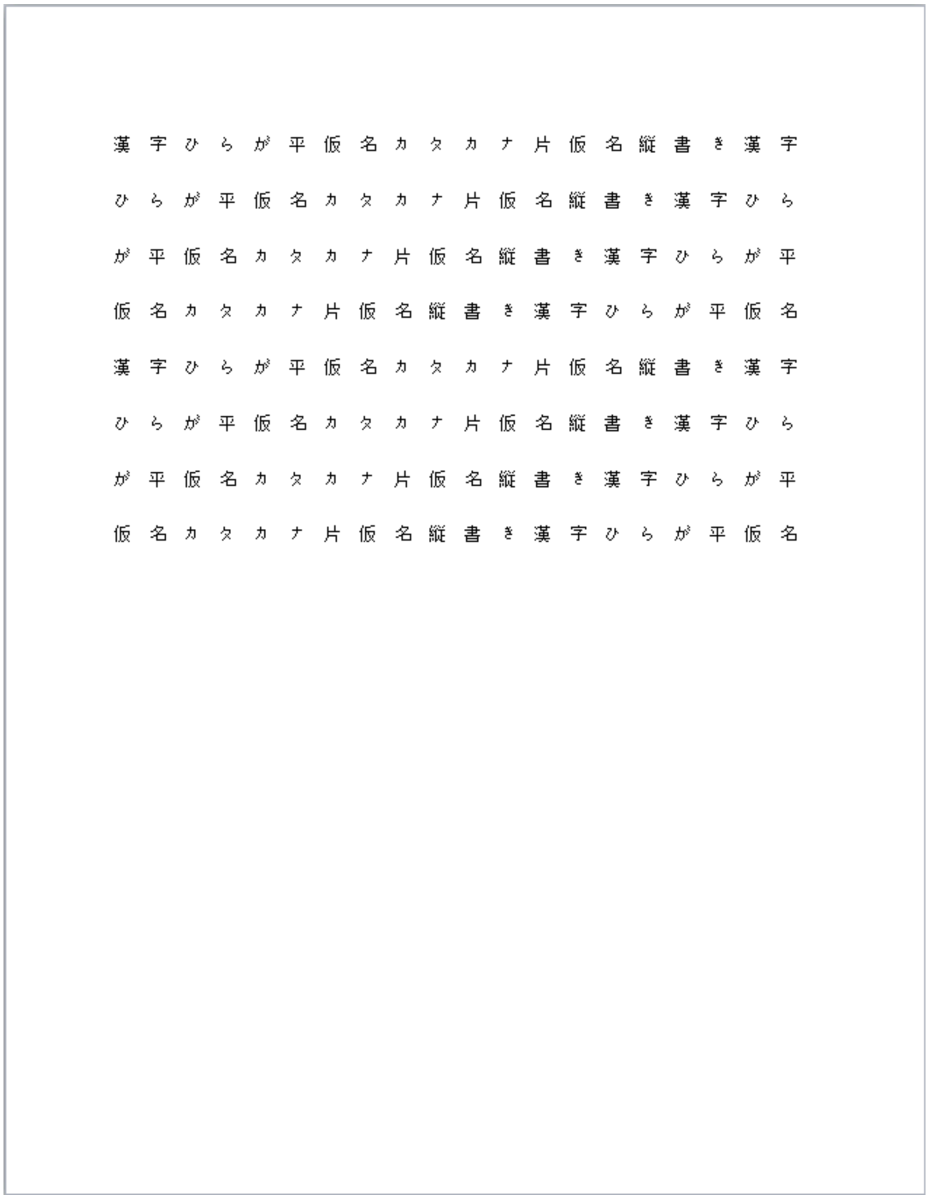
[*Note*: The W3C XML Schema definition of this element's content model (CT_Columns) is located in §A.1. *end note*]

17.6.5 docGrid (Document Grid)

This element specifies the settings for the document grid, which enables precise layout of full-width East Asian language characters within a document by specifying the desired number of characters per line and lines per page for all East Asian text content in this section.

[*Example*: Consider a document with the document grid defined to allow 20 characters per line, and 20 lines per page by snapping characters to the grid (type attribute of snapToChars) as follows:

©ISO/IEC 2012 – All rights reserved



漢字ひらが平仮名カタカナ片仮名縦書き漢字

ひらが平仮名カタカナ片仮名縦書き漢字ひら

が平仮名カタカナ片仮名縦書き漢字ひらが平

仮名カタカナ片仮名縦書き漢字ひらが平仮名

漢字ひらが平仮名カタカナ片仮名縦書き漢字

ひらが平仮名カタカナ片仮名縦書き漢字ひら

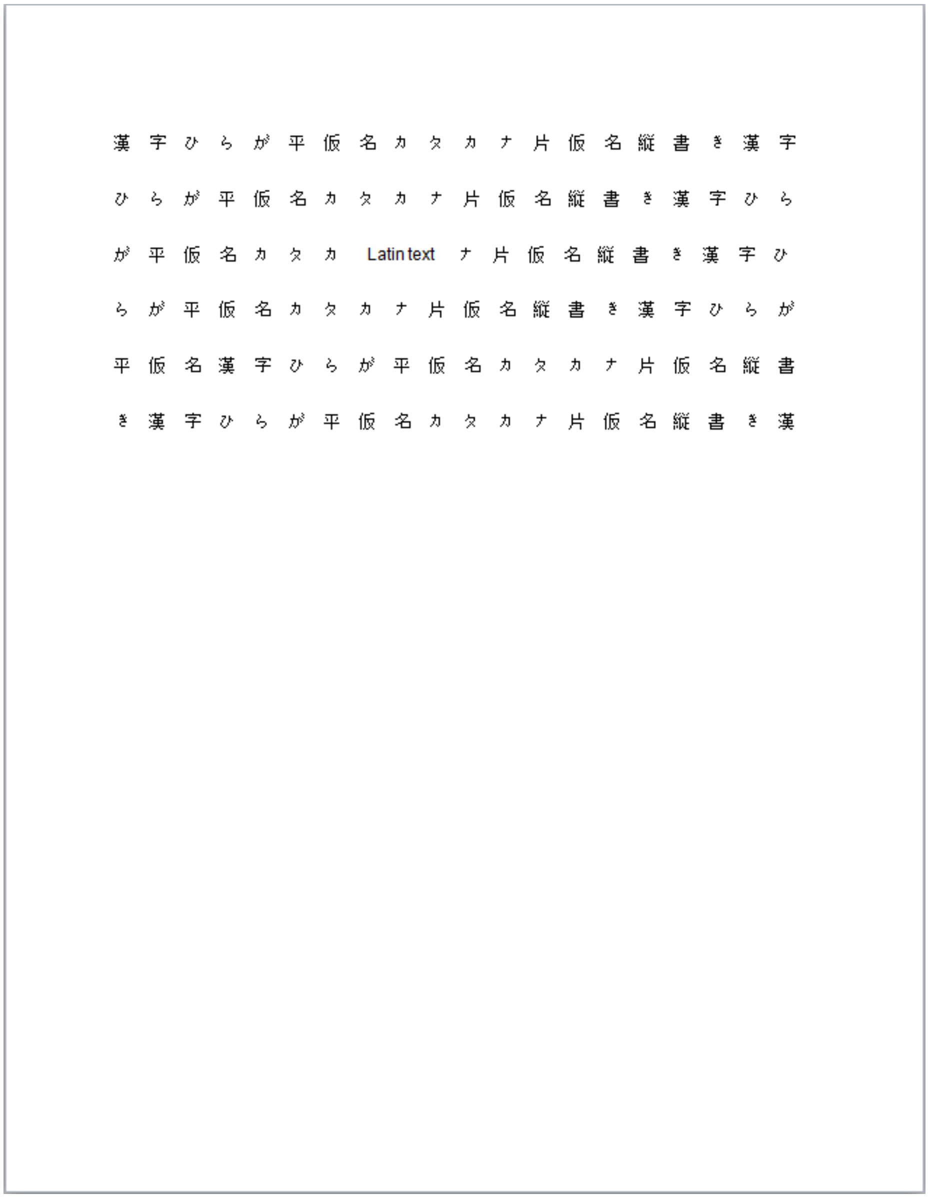
が平仮名カタカナ片仮名縦書き漢字ひらが平

仮名カタカナ片仮名縦書き漢字ひらが平仮名

As shown, this document allows for only 20 East Asian characters per line by adjusting the inter-character spacing to ensure that there are only 20 characters per line. *end example*]

If Latin text is interspersed on this line, then it is placed across the number of grid units needed to fit the content, but all other grid positions are unaffected.

[*Example*: Consider the example above with the addition of the text "Latin text" in English, as follows:



漢字ひらが平仮名カタカナ片仮名縦書き漢字

ひらが平仮名カタカナ片仮名縦書き漢字ひら

が平仮名カタカLatin textナ片仮名縦書き漢字ひ

らが平仮名カタカナ片仮名縦書き漢字ひらが

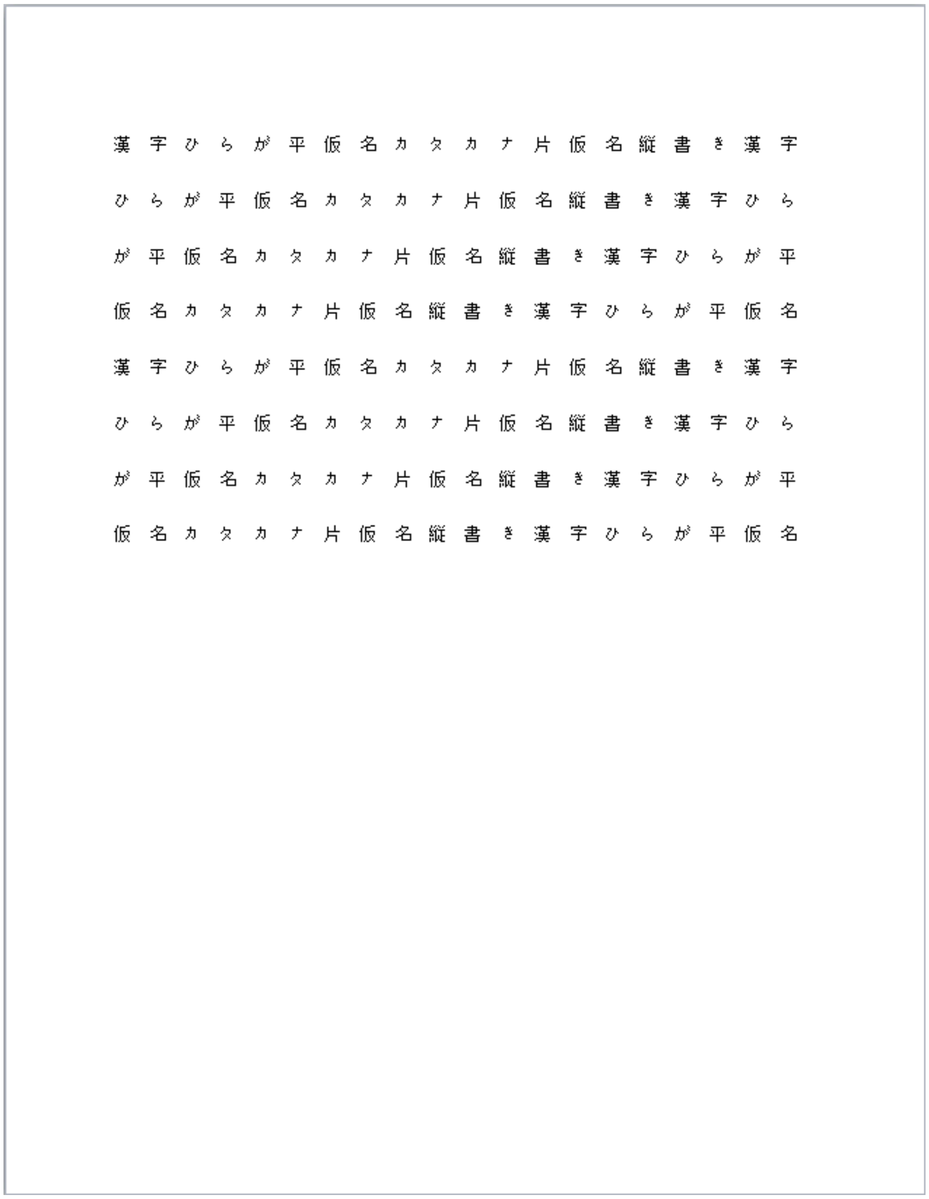
平仮名漢字ひらが平仮名カタカナ片仮名縦書

き漢字ひらが平仮名カタカナ片仮名縦書き漢

The Latin text spans two grid units, so it is placed in the center of those two units; no other grid positions are affected, so the text on the second line now spans two additional grid units. *end example*]

Attributes	Description
charSpace (Document Grid Character Pitch)	Specifies the number of characters to be allowed on the document grid for each line in this section. This attribute's value shall be specified by multiplying the difference between the desired character pitch and the character pitch for that character in the font size of the Normal font by 4096. If this attribute is omitted, the default value is zero. This value shall then be used to add the character pitch for the specified point size to each character in the section [Note: This results in text in the Normal style having a

Attributes	Description
	specific number of characters per line. end note] [*Example*: Consider a section with a Normal font size of 11 points on which a 21 point pitch document grid has been defined.. The resulting WordprocessingML would be defined as follows: `<w:docGrid w:charSpace="40960" …/>` The charSpace attribute specifies a value of 40960, which means that the delta between the character pitch of each character in the grid and the Normal font is 10 points, resulting in a character pitch of 11+10 = 21 points for all characters in this section. *end example*] Individual runs of text can override the line pitch information specified for the document grid by specifying that the run text shall not snap to the document grid via the snapToGrid element (§17.3.2.34). The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).
linePitch (Document Grid Line Pitch)	Specifies the number of lines to be allowed on the document grid for the current page assuming all lines have equal line pitch applied to them. This line pitch shall not be added to any line which appears within a table cell unless the adjustLineHeightInTable element (§17.15.3.1) is present in the document's compatibility settings. This attribute is specified in twentieths of a point, and defines the pitch for each line of text on this page such that the desired number of single spaced lines of text fits on the current page. [*Example*: Consider a standard 8.5x11" page on which a 20 character wide, 20 line document grid has been defined. The resulting WordprocessingML would be defined as follows: `<w:docGrid w:linePitch="684" …/>` The linePitch attribute specifies that 34.2 points is to the amount of pitch allowed for each line on this page in order to maintain the specific document grid. *end example*] Individual paragraphs can override the line pitch information specified for the document grid by either: • Specifying an exact line spacing value using the lineRule attribute of value exact on the spacing element (§17.3.1.33). • Specifying that the paragraph text shall not snap to the document grid via the snapToGrid element (§17.3.1.32). The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

Attributes	Description
type (Document Grid Type)	Specifies the style of the current document grid, which defines the grid behavior. The grid can define a grid which snaps all East Asian characters to grid positions, but leaves Latin text with its default spacing; a grid which adds the specified character pitch to each character on each row; or a grid which affects only the line pitch for the current section. [*Example*: Consider the document discussed above with the document grid defined to allow 20 characters per line, and 20 lines per page by snapping characters to the grid as follows:  漢字ひらが平仮名カタカナ片仮名縦書き漢字 ひらが平仮名カタカナ片仮名縦書き漢字ひら が平仮名カタカナ片仮名縦書き漢字ひらが平 仮名カタカナ片仮名縦書き漢字ひらが平仮名 漢字ひらが平仮名カタカナ片仮名縦書き漢字 ひらが平仮名カタカナ片仮名縦書き漢字ひら が平仮名カタカナ片仮名縦書き漢字ひらが平 仮名カタカナ片仮名縦書き漢字ひらが平仮名

Attributes	Description
	This document has a type attribute of snapToChars, which specifies that the grid must force East Asian characters to fit 20 to a line. *end example*] If this attribute is omitted, the value “default” shall be used. [*Note:* Disabling the document grid in this way rather than omitting the docGrid element allows for the preservation of document grid settings if and when the document grid is re-enabled. *end note*] The possible values for this attribute are defined by the ST_DocGrid simple type (§17.18.14).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_DocGrid) is located in §A.1. *end note*]

17.6.6 formProt (Only Allow Editing of Form Fields)

This element specifies that the contents of the current section shall be protected such that they cannot be edited by a user (if the consumer is displaying the document and allowing the user to make modification) except for the text contained in any form field or embedded control that is part of the current section.

[*Example*: Consider a section consisting of three paragraphs of text and a single text form field, located at the beginning of the second paragraph. If this section is protected in this manner, a user would only be permitted to edit the contents of the text form field, and all other contents would be locked to prevent user edits. *end example*]

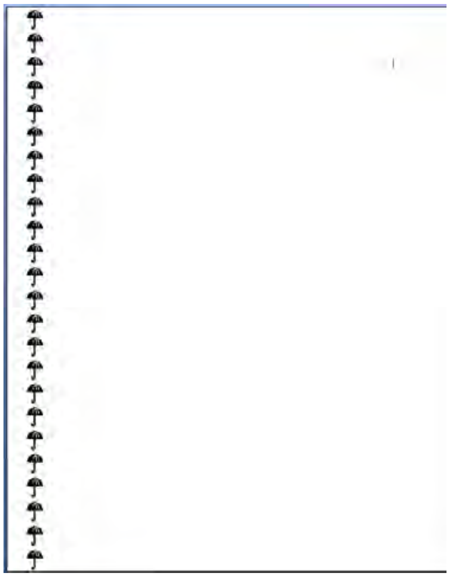
The enforcement of this property is determined by the documentProtection element (§17.15.1.29), as it is possible to specify protection without turning it on.

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.6.7 left (Left Border)

This element specifies the presentation and display of the page border displayed at the left of each page in this section.

[*Example*: Consider a section in which all pages have a left border consisting of a repeated image of an umbrella, like this:



This border would result in the following WordprocessingML:

```
<w:sectPr>
  …
  <w:pgBorders>
    <w:left w:val="seattle" …/>
  </w:pgBorders>
  …
</w:sectPr>
```

Because the page only has a border at the left, only the left element is specified within the set of page borders. *end example*]

When a document has a left border that is relative to the page edges (using the offsetFrom attribute value of page on pgBorders), it shall span the left edge of the page at the location defined by its properties, stopping when:

- It intersects with the corresponding top or bottom page border (if one is specified).
- It reaches the edge of the page.

[*Example*: In the example above, no top or bottom border was specified in the WordprocessingML, so a consumer must draw the border from one edge of the page to the other. *end example*]

When a document has a left border that is relative to the text (using the offsetFrom attribute value of text on pgBorders), it shall span only the necessary width to satisfy the requirement of spanning the width of the text.

When a document has custom border art specified by the id attribute, it shall use the corresponding relationship part item as an image for the left border. If the corresponding relationship part item cannot be located, the consumer shall use the border specified by the value of the val attribute. If the corresponding value of the val attribute cannot be resolved, no left border is present when the page is displayed.

Attributes	Description
color (Border Color)	Specifies the color for this border. This value can be defined as either: - A color value using the RGB color model whose red, green, and blue values are written as numbers in the range 0 to 255, hex encoded, and concatenated. [*Example*: Full intensity red would be 255 red, 0 green, 0 blue, encoded to FF, 00, 00, and concatenated to FF0000. *end example*] . RGB colors are specified in the sRGB color space. - auto to allow a consumer to automatically determine the border color in order to make the document's text readable. [*Example*: A document with white text and a background color of auto might result in the use of a black background, in order to ensure legibility of the content. *end example*] [*Example*: Consider a border color with value auto, as follows: `<w:bottom … w:color="auto"/>` This color therefore can be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§17.18.38).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which must appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" … />` This frame's val is true, indicating that the border frame effect must be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
id (Custom Defined Border Relationship Reference)	Specifies the relationship ID for the relationship which contains the custom border image for the parent element. This custom border image is contained in a separate part within the WordprocessingML package.

Attributes	Description
Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	The relationship explicitly targeted by this attribute shall be of type http://purl.oclc.org/ooxml/officeDocument/relationships/image or the document shall be considered non-conformant. If this attribute is omitted, then no custom border shall be used. [*Example*: Consider the following WordprocessingML markup for a custom bottom border in a document: `<w:bottom w:val="custom" r:id="rIdCustomBottomBorder" …/>` The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rIdCustomBottomBorder` must contain the custom bottom border image for the document. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which must appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" … />` This frame's val is `true`, indicating that the shadow effect must be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object. When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§17.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points. When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§17.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.

Attributes	Description

	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24" />`
	`</w:pgBorders`
	The offsetFrom attribute specifies that the space value provides the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset must be 24 points. *end example*]
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§17.18.68).

sz (Border Width)	Specifies the width of the current border.
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range can be reassigned to a more appropriate value.
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range can be reassigned to a more appropriate value.
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§17.18.23).

themeColor (Border Theme Color)	Specifies the base theme color used to generate the border color. The border color is the RGB value associated with themeColor as further transformed by themeTint or themeShade (if one is present), else the background color is the RGB value associated with themeColor. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part (§14.2.7 and §20.1.6.9), which allows color information to

Attributes	Description

	be set centrally in the document.
	To determine the color to display, the following actions are performed:
	• Using the mapping specified in the ST_ThemeColor simple type (§17.18.97), the appropriate attribute on the clrSchemeMapping element (§17.15.1.20) is read.
	• Using that value and the mapping specified in the ST_ColorSchemeIndex simple type (§17.18.103), the appropriate element in the document's Theme part is read to get the base theme color.
	• The specified color is modified based on the presence of the themeTint or themeShade attribute.
	[*Example*: Consider a set of borders configured to use the `accent2` theme color, resulting in the following WordprocessingML markup:
	`<w:top … w:themeColor="accent2" w:themeTint="99" />`
	`<w:bottom … w:themeColor="accent2" w:themeTint="99" />`
	`<w:left … w:themeColor="accent2" w:themeTint="99" />`
	`<w:right … w:themeColor="accent2" w:themeTint="99" />`
	If the Settings part contained the following markup:
	`<w:clrSchemeMapping … w:accent2="accent2"/>`
	and the Theme part contained the following XML markup:
	`<a:accent2>`
	`  <a:srgbClr val="4F81BD"/>`
	`</a:accent2>`
	the resulting border color would be `95B3D7` (the result of a 60% tint applied to the original theme color; see the calculations in themeTint below for details). *end example*]
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§17.18.97).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeColor attribute is not present, then this attribute shall not be used. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows:

Attributes	Description

	$\begin{aligned} S_{xml} &= 0.4 * 255 \\ &= 102 \\ &= 66(hex) \end{aligned}$
	The resulting themeShade value in the file format would be `66`. *end example*]
	Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows:
	• Convert the color to the HSL color format (values from 0 to 1)
	• Modify the luminance factor as follows:
	$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$
	• Convert the resultant HSL color to RGB
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`.
	The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48{,}0.53\right)$.
	Applying the shade formula with a shade percentage of 75% to the luminance, we get:
	$\begin{aligned} L' &= 0.53 * 0.75 \\ &= 0.39698 \end{aligned}$
	Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48{,}0.39698\right)$and converting back to RGB, we get `943634`.
	This transformed value can be seen in the resulting background's color attribute:
	`<w:top w:val="single" w:sz="4" w:space="24"`
	`  w:color="943634" w:themeColor="accent2"`
	`  w:themeShade="BF"/>`
	end example]
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeColor attribute is not present, then this attribute shall not be used. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to

Attributes	Description

	the current border.
	[*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows:
	$$\begin{aligned} T_{xml} &= 0.6 * 255 \\ &= 153 \\ &= 99(hex) \end{aligned}$$
	The resulting themeTint value in the file format would be 99. *end example*]
	Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows:
	- Convert the color to the HSL color format (values from 0 to 1)
	- Modify the luminance factor as follows:
	$$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$
	- Convert the resultant HSL color to RGB
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`.
	The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45, 0.53\right)$.
	Applying the tint formula with a tint percentage of 60% to the luminance, we get:
	$$\begin{aligned} L' &= 0.53 * 0.6 + (1 - .6) \\ &= 0.71 \end{aligned}$$
	Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$and converting back to RGB, we get `95B3D7`.
	This transformed value can be seen in the resulting background's color attribute:
	`<w:top w:val="single" w:sz="4" w:space="24"`
	`  w:color="95B3D7" w:themeColor="accent2"`
	`  w:themeTint="99"/>`
	end example]
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

val (Border Style)	Specifies the style of border used on this object.

Attributes	Description
	This border can either be an art border (a repeated image along the borders - shall only be used for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§17.18.2).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageBorder) is located in §A.1. *end note*]

17.6.8 lnNumType (Line Numbering Settings)

This element specifies the settings for line numbering to be displayed before each column of text in this section in the document.

[*Example*: Consider the line numbering used on each page of this document, which specifies: line numbering for each line, restarting at one at the top of each new page. This line-numbering scheme would be defined using the following WordprocessingML:

```
<w:sectPr>
  …
  <w:lnNumType w:countBy="1" />
</w:sectPr>
```

This content specifies that line numbers shall be included on each line, restart on each page (the default), be placed automatically based on the text (the default), and shall restart at one (the default). *end example*]

Attributes	Description
countBy (Line Number Increments to Display)	Specifies the line number increments to be displayed in the current document. Although each line has an associated line number, only lines which are an even multiple of this value shall be displayed. [*Example*: Consider a document in which only every fifth line must have a line number. The resulting WordprocessingML for this setting would be: `<w:lnNumType … w:countBy="5"/>`

Attributes	Description
	This setting ensures that only lines whose number is a multiple of (e.g. 5, 10, and 15) has a line number displayed. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).
distance (Distance Between Text and Line Numbering)	Specifies the distance between the text margin and the edge of any line numbers appearing in that section. [*Example*: Consider a document in which the line numbering must appear one-half inch from the text margin. The WordprocessingML for this setting is: `<w:lnNumType … w:distance="720"/>` The distance attribute specifies that there must be a `720` twip spacing between the text margin and the *line numbering. end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

restart (Line Numbering Restart Setting)	Specifies when the line numbering in this section shall be reset to the line number specified by the start attribute's value.
	The line numbering increments for each line (even if it is not displayed) until it reaches the restart point specified by this element.
	[*Example*: Consider the line numbering used on each page of this document, which specifies that line numbering must restart at the top of each new page. This line numbering setting would be defined using the following WordprocessingML:
	`<w:sectPr>`
	`  …`
	`  <w:lnNumType w:restart="newPage" … />`
	`</w:sectPr>`
	The value of `newPage` specifies that the line numbers must restart at the top of each page to the value specified by the start attribute. In this case, `newPage` is the default, so this value could have been omitted entirely. *end example*]
	The possible values for this attribute are defined by the ST_LineNumberRestart simple type (§17.18.47).

start (Line Numbering Starting Value)	Specifies the starting value used for the first line whenever the line numbering is restarted by use of the restart attribute. [*Example*: Consider a document in which line numbering must appear on every fifth line, but the first line must be treated as line number . This setting would require the following WordprocessingML syntax:

Attributes	Description
	`<w:lnNumType w:start="3" w:countBy="5"/>` The start attribute specifies that line numbers must be counted starting from the number 3. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_LineNumber) is located in §A.1. *end note*]

17.6.9 paperSrc (Paper Source Information)

This element specifies printer-specific settings for the printer tray(s) that shall be used to print different pages in this section in the document.

[Example: Consider a section which must use the best possible tray when printing all pages in this section. This information is specified using the following WordprocessingML:

```
<w:paperSrc w:first="1" w:other="1" />
```

The attributes on the paperSrc element specify the printer codes for the trays to be used when printing this section. end example]

Attributes	Description
first (First Page Printer Tray Code)	Specifies a printer-specific code that uniquely identifies a specific printer tray to be used to print the first page of this section in the document. A first value of 1 (the default) is specifically used to indicate that the printer shall automatically select the appropriate printer tray based on the printed page size. [*Example*: Consider a section which must use the best possible tray when printing the first page in this section. This information is specified using the following WordprocessingML: `<w:paperSrc w:first="1" w:other="1" />` The first attribute on the paperSrc element specifies that the printer must automatically select the appropriate printer tray based on the printed page size when printing the first page in this section. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

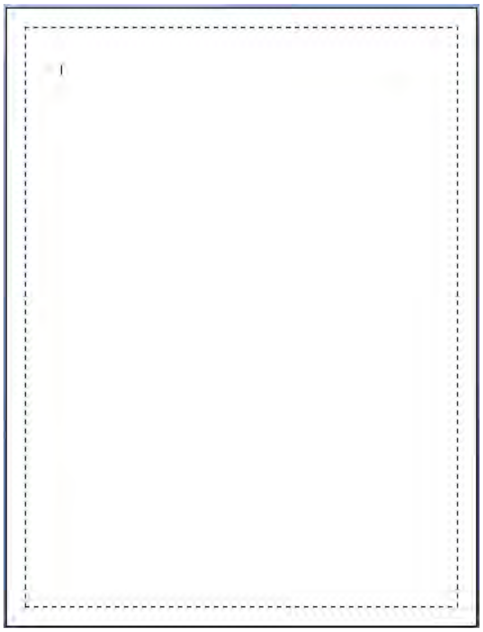
Attributes	Description
other (Non-First Page Printer Tray Code)	Specifies a printer-specific code that uniquely identifies a specific printer tray to be used to print the each subsequent (non-first) page of this section in the document. An value of 1 (the default) is specifically used to indicate that the printer shall automatically select the appropriate printer tray based on the printed page size. [*Example*: Consider a section which must use the best possible tray when printing the all pages in this section. This information is specified using the following WordprocessingML: `<w:paperSrc w:first="1" w:other="1" />` The other attribute on the paperSrc element specifies that the printer must automatically select the appropriate printer tray based on the printed page size when printing all pages after the first in this section. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PaperSource) is located in §A.1. *end note*]

17.6.10 pgBorders (Page Borders)

This element specifies the page borders for each page in this section. Each child element of the pgBorders element specifies a specific of border (left, right, bottom, or top).

[*Example*: Consider a page that specifies a dashed line border around each of the four sides of the page, as follows:



This page border setting would be specified using the following WordprocessingML:

```
<w:pgBorders w:offsetFrom="page">
  <w:top w:val="dashed" w:sz="4" w:space="24" w:color="auto" />
  <w:left w:val="dashed" w:sz="4" w:space="24" w:color="auto" />
  <w:bottom w:val="dashed" w:sz="4" w:space="24" w:color="auto" />
  <w:right w:val="dashed" w:sz="4" w:space="24" w:color="auto" />
</w:pgBorders>
```

The four page borders are each uniquely defined by the top, left, bottom, and right elements, respectively. Global settings that define the placement of all page borders are stored on the pgBorders element directly. *end example*]

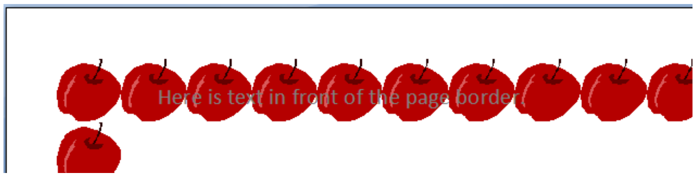
Attributes	Description

display (Pages to Display Page Borders)	Specifies the pages in this section on which the page border shall be printed.
	If this attribute is omitted, then the page borders shall be displayed on all pages in this section (equivalent to a value of `allPages`).
	[*Example*: Consider a section in a document for which the page border must only be printed on the first page. This setting is specified using the following WordprocessingML:
	`<w:pgBorders w:display="firstPage">`
	`  …`
	`</w:pgBorders>`
	The display attribute specifies that only the first page must display the page border defined for this section. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_PageBorderDisplay simple type (§17.18.62).

offsetFrom (Page Border Positioning)	Specifies how the relative positioning of the page borders shall be calculated.
	If the value of this attribute is page, then the space attribute on each page border shall be interpreted as the distance from the edge of the page that shall be left before the page border.
	If the value of this attribute is `text`, then the space attribute on each page border shall be interpreted as the distance from the text margins that shall be left before the page border.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:top w:val="dashed" w:space="24" />`
	`  <w:left w:val="dashed" w:space="24" />`
	`  <w:bottom w:val="dashed" w:space="24"/>`
	`  <w:right w:val="dashed" w:space="24"/>`
	`</w:pgBorders>`
	This fragment specifies that the page borders must be indented 24 points from the page extents.
	This is distinct from the following fragment with identical space attribute values:
	`<w:pgBorders w:offsetFrom="text">`
	`  <w:top w:val="dashed" w:space="24" />`
	`  <w:left w:val="dashed" w:space="24" />`
	`  <w:bottom w:val="dashed" w:space="24"/>`
	`  <w:right w:val="dashed" w:space="24"/>`
	`</w:pgBorders>`
	In this case, the page borders is offset by 24 points, but in this case, that offset is calculated relative to the text margins. *end example*]
	The possible values for this attribute are defined by the ST_PageBorderOffset simple type (§17.18.63).

zOrder (Z-Ordering of Page Border)	Specifies whether the page border is positioned above or below intersecting texts and objects in this document. [*Example*: Consider a document in which the page border must be displayed below any intersecting text as follows:

Attributes	Description
	 This setting is specified by setting the zOrder attribute to back, which specifies that the page border must be display behind all intersecting text and objects. *end example*] The possible values for this attribute are defined by the ST_PageBorderZOrder simple type (§17.18.64).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageBorders) is located in §A.1. *end note*]

17.6.11 pgMar (Page Margins)

This element specifies the page margins for all pages in this section.

[*Example*: Consider a page with a one-inch margin on all sides. Specifying these margins requires the following WordprocessingML:

```
<w:sectPr>
  <w:pgMar w:header="720" w:bottom="1440" w:top="1440" w:right="1440"
w:left="1440" w:footer="720" w:gutter="0" />
  …
</w:sectPr>
```

This section specifies page margins of 1,440 twentieths of a point (one inch) on all sides. *end example*]

[*Note*: With reference to Fig. 48 of *Requirements for Japanese Text Layout*, typical positions (a), (b), (c), (d), and (f), but not (e), of running heads and page numbers for vertically set books with double running heads, and typical positions (a), (b), (c), and (d) of running heads and page numbers for horizontally set books with double running heads, can be represented using WordprocessingML. *end note*]

Attributes	Description
bottom (Page Bottom Spacing)	Specifies the distance (in twentieths of a point) between the bottom of the text margins for the main document and the bottom of the page for all pages in this section. If the value of bottom is non-negative, then the text is placed at the greater of: • The value of bottom • The extent of the footer text [*Example*: Consider a document where the footer must start one inch of the bottom of

Attributes	Description

	the page extent, but the contents of the main document story specify that they must start one-half of an inchh from the page extents. To specify these boundaries, the following page margins can specified in the WordprocessingML:
	
	`<w:pgMar … w:bottom="720"`
	`  w:footer="1440"/>`
	
	This fragment specifies that the footer must start 1440 twentieths of a point from the bottom of the page, but the main document story must start 720 twentieths of a point from the bottom of the page. Since the footer extent is guaranteed to be greater in this case, the bottom text extent ends at the top of the footer region. *end example*]
	
	A negative value indicates that the contents of the main document shall be measured from the bottom of the page extent regardless of the footer for that document, and therefore shall overlap the footer text.
	
	[*Example*: Consider a document where the footer must start one inch of the bottom of the page extent, but the contents of the main document story must start one-half of an inch from the page extents. To specify these boundaries, the following page margins can be specified in the WordprocessingML:
	
	`<w:pgMar … w:bottom="-720"`
	`  w:footer="1440"/>`
	
	This fragment specifies that the footer must start 1440 twentieths of a point from the bottom of the page, and the main document story must start 720 twentieths of a point from the bottom of the page. Since the value of bottom is negative in this case, the bottom text extent starts one-half of an inch from the bottom of the page and overlaps any footer text. *end example*]
	
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

footer (Spacing to Bottom of Footer)	Specifies the distance (in twentieths of a point) from the bottom edge of the page to the bottom edge of the footer. [*Example*: Consider a document where the footer must start one inch above the bottom of the page extent. To specify this boundary, the following page margins must specified in the WordprocessingML: `<w:pgMar … w:footer="1440"/>` This fragment specifies that the footer must start 1440 twentieths of a point from the bottom of the page. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).
gutter (Page Gutter Spacing)	Specifies the page gutter for each page in the current section. The *page gutter* defines the amount of extra space added to the specified margin, above any existing margin values. [*Note*: This setting is typically used when a document is being created for binding, in order to ensure that the resulting margins are present after the binding gutter is consumed by the printed matter binding. *end note*] [*Example*: Consider a document where the margin must start one inch of the left edge of the page extent after one-half of an inch is hidden by the page binding. To specify this condition, a user could simply use a left margin of 1.5 inches, which would be lost if the margins are later changed, or could specify a one-half inch gutters follows in the WordprocessingML: `<w:pgMar … w:gutter="720"/>` This fragment specifies that the gutter must span 720 twentieths of a point, after which any margin value must be added. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).
header (Spacing to Top of Header)	Specifies the distance (in twentieths of a point) from the top edge of the page to the top edge of the header. [*Example*: Consider a document where the header must start two inches below the top of the page extent. To specify this boundary, the following page margins must specified in the WordprocessingML: `<w:pgMar … w:header="2880"/>` This fragment specifies that the header must start 2880 twentieths of a point from the top of the page. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).
left (Left Margin Spacing)	Specifies the distance (in twentieths of a point) between the left edge of the page and the left edge of the text extents for this document. [*Example*: Consider a document where the left text extent must start two inches inside the page extent. To specify this boundary, the following page margins must specified in the WordprocessingML: `<w:pgMar … w:left="2880"/>`

Attributes	Description
	This fragment specifies that the left margin must span 2880 twentieths of a point from the left edge of the page. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).
right (Right Margin Spacing)	Specifies the distance (in twentieths of a point) between the right edge of the page and the right edge of the text extents for this document. [*Example*: Consider a document where the right text extent must start one inch inside the page. To specify this boundary, the following page margins must specified in the WordprocessingML: `<w:pgMar … w:right="1440"/>` This fragment specifies that the right margin must span 1440 twentieths of a point from the right edge of the page. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).
top (Top Margin Spacing)	Specifies the distance (in twentieths of a point) between the top of the text margins for the main document and the top of the page for all pages in this section. If the value of top is non-negative, then the text is placed at the greater of: • The value of top • The extent of the header text [*Example*: Consider a document where the header must start one inch from the top of the page extent, but the contents of the main document story specify that they must start one-half of an inch from the page extents. To specify these boundaries, the following page margins can be specified in the WordprocessingML: `<w:pgMar … w:top="720" w:header="1440"/>` This fragment specifies that the header must start 1440 twentieths of a point from the top of the page, but the main document story must start 720 twentieths of a point from the top of the page. Since the header extent is guaranteed to be greater in this case, the main text extent ends at the bottom of the header region. *end example*] A negative value indicates that the contents of the main document shall be measured from the top of the page extent regardless of the header for that document, and therefore shall overlap the header text.

Attributes	Description

	[*Example*: Consider a document where the header must start one inch from the top of the page extent, but the contents of the main document story must start one-half of an inch from the page extents. To specify these boundaries, the following page margins can be specified in the WordprocessingML:
	`<w:pgMar … w:top="-720"`
	`  w:header="1440"/>`
	This fragment specifies that the header must start 1440 twentieths of a point from the top of the page, and the main document story must start 720 twentieths of a point from the top of the page. Since the value of top is negative in this case, the top text extent starts one-half of an inch from the top of the page and overlaps any header text. *end example*]
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageMar) is located in §A.1. *end note*]

17.6.12 pgNumType (Page Numbering Settings)

This element specifies the page numbering settings for all page numbers that appear in the contents of the current section.

[*Example*: Consider a section in which the page numbers must start at page 25. The following WordprocessingML syntax specifies that requirement:

```
<w:sectPr>
  …
  <w:pgNumType w:start="25"/>
</w:sectPr>
```

The pgNumType element specifies that numbering on this section must start from page number 25. *end example*]

Attributes	Description
chapSep (Chapter Separator Character)	Specifies the separator character that shall appear between the chapter and page number, if a chapter style has been set for page numbers in this section. If the chapStyle attribute is not present, or its specified heading level does not have an associated numbering format, then this value is ignored, since no chapter number is output by the field. [Example: Consider a section in a document in which the chapter must be separated from

Attributes	Description
	the page number using a colon character. This constraint would be specified using the following WordprocessingML: `<w:pgNumType w:chapSep="colon" w:chapStyle="1" />` The chapSep attribute declares that the chapter and page number must be separated by a `colon` (e.g. 1:1 for chapter one, page one). end example] The possible values for this attribute are defined by the ST_ChapterSep simple type (§17.18.6).
chapStyle (Chapter Heading Style)	Specifies the one-based index of the heading style applied to chapter titles in the document which shall be used as chapter headings in all page numbers for this section, by locating the nearest heading of that style and extracting the numbering information. If the specified heading style does not exist in the current section, or does not have a numbering format, then any previous level heading format shall be used as needed as the specified chapter number. If no heading has numbering information and/or is used in the section, then the chapter and chapter separator shall be omitted from the page numbering data. [*Example*: Consider a page number in a section with page numbering properties that specify a chapStyle of `1` (Heading 1 style) and a chapSep of `dash`. This means that for each page number in this section, the numbering value of the nearest Heading 1 style is used for the chapter value, and is followed by a dash, then the page n umber in that section. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).
fmt (Page Number Format)	Specifies the number format that shall be used for all page numbering in this section. [*Example*: A fmt value of `lowerLetter` indicates that a consumer must use lowercase letters for each page in this section: a,b,c… *end example*] The possible values for this attribute are defined by the ST_NumberFormat simple type (§17.18.59).
start (Starting Page Number)	Specifies the page number that appears on the first page of the section. If this value is omitted, numbering continues from the highest page number in the previous section. [*Example*: Consider the following WordprocessingML: `<w:pgNumType w:fmt="lowerLetter"/>`

Attributes	Description
	Because the start value is omitted, the page numbers in this section begin at the value of the highest page in the previous section. This means that if the previous section ended in page 7, this section would start with page 8. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageNumber) is located in §A.1. *end note*]

17.6.13 pgSz (Page Size)

This element specifies the properties (size and orientation) for all pages in the current section.

[*Example*: Consider a section that must be printed on A4 paper. The WordprocessingML for this paper size is as follows:

```
<w:pgSz w:w="11907" w:h="16839" />
```

This output states that all pages in this section must be 11907 twentieths of a point wide (11907 twentieths of a point = 8.269") and 16839 twentieths of a point high (16939 twentieths of a point = 11.694"). *end example*]

Attributes	Description
code (Printer Paper Code)	Specifies an optional value which can be used to store an identifier for the current paper size. This code is stored solely to show a description for the current paper size. This setting should not be used to determine the target paper size (i.e. if the w and h attributes are omitted, this setting has no meaning). [Example: Consider the following WordprocessingML fragment: `<w:pgSz w:w="12240" w:h="15840" w:code="240" />` The code attribute specifies a value of 240, whichis, since the w and h attributes specify a page size of 8.5 inches by 11 inches, can be used to determine the appropriate user label for this paper size – for example, "Letter" or "8.5 by 11". The attribute code specifies that the producing application's unique identifier value for this paper size was 240. This value does not itself determine the paper size, regardless of the presence of the w and h attributes. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type

Attributes	Description
	(§17.18.10).
h (Page Height)	Specifies the height (in twentieths of a point) for all pages in the current section. [*Example*: Consider the following WordprocessingML: `<w:pgSz w:w="15840" w:h="12240" />` All pages in this section are displayed on a page that is `12240` twentieths of a point (8.5") tall. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).
orient (Page Orientation)	Specifies the orientation of all pages in this section. This information is used to determine the actual paper size to use on the printer. [*Example*: Pages 11" wide by 8.5" long in landscape mode use 8.5"x11" paper, because the width and height are reversed for pages in this landscape section with respect to the printed page. *end example*] This implies that the actual paper size width and height are reversed for pages in this section. If this attribute is omitted, then `portrait` shall be implied. [*Example*: Consider the following WordprocessingML: `<w:pgSz w:w="15840" w:h="12240" w:orient="landscape" />` Although the page width is 11", and page height is 8.5", according to the w and h attributes, because the orient attribute is set to `landscape`, pages in this section are printed on 8.5x11" paper in landscape mode. *end example*] The possible values for this attribute are defined by the ST_PageOrientation simple type (§17.18.65).
w (Page Width)	This attribute indicates the width (in twentieths of a point) for all pages in the current section. [*Example*: Consider the following WordprocessingML: `<w:pgSz w:w="15840" w:h="12240" />` All pages in this section are displayed on a page that is `15840` twentieths of a point (11") wide. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageSz) is located in §A.1. *end note*]

17.6.14 printerSettings (Reference to Printer Settings Data)

This element specifies an explicit relationship to a Printer Settings part containing information about the printer settings used for this section.

If this element is omitted, than no additional settings are associated with this section.

[*Example*: Consider a producer which needed to store additional printer settings for each section. A document from such a producer would have the following section properties:

```
<w:sectPr>
  …
  <w:printerSettings r:id="rId10" />
</w:sectPr>
```

The resulting Main Document part would a relationship to the appropriate Printer Settings part with a relationship ID of `rId10`. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*]

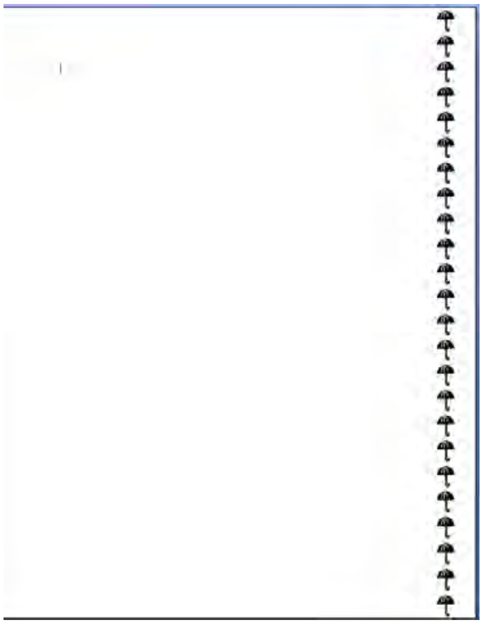
Attributes	Description
	The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rel) is located in §A.1. *end note*]

17.6.15 right (Right Border)

This element specifies the presentation and display of the page border displayed at the right of each page in this section.

[*Example*: Consider a section in which all pages must have a right border consisting of a repeated image of an umbrella, like this:



This border would result in the following WordprocessingML:

```
<w:sectPr>
  …
  <w:pgBorders>
    <w:right w:val="seattle" …/>
  </w:pgBorders>
  …
</w:sectPr>
```

Because the page has a border at the right only, only the right element is specified within the set of page borders. *end example*]

©ISO/IEC 2012 – All rights reserved

When a document has a right border that is relative to the page edges (using the offsetFrom attribute value of page on pgBorders), it shall span the right edge of the page at the location defined by its properties, stopping when:

- It intersects with the corresponding top or bottom page border (if one is specified)
- It reaches the edge of the page.

[*Example*: In the example above, no top or bottom border was specified in the WordprocessingML, so a consumer must draw the border from one edge of the page to the other. *end example*]

When a document has a right border that is relative to the text (using the offsetFrom attribute value of `text` on pgBorders), it shall only span the necessary width to satisfy the requirement of spanning the width of the text.

When a document has custom border art specified by the id attribute, it shall use the corresponding relationship part item as an image for the right border. If the corresponding relationship part item cannot be located, the consumer shall use the border specified by the value of the val attribute. If the corresponding value of the val attribute cannot be resolved, no right border is present when the page is displayed.

Attributes	Description
color (Border Color)	Specifies the color for this border. This value can be defined as either: • A color value using the RGB color model whose red, green, and blue values are written as numbers in the range 0 to 255, hex encoded, and concatenated. [*Example*: Full intensity red would be 255 red, 0 green, 0 blue, encoded to FF, 00, 00, and concatenated to `FF0000`. *end example*] . RGB colors are specified in the sRGB color space. • `auto` to allow a consumer to automatically determine the border color in order to make the document's text readable. [*Example*: A document with white text and a background color of `auto` might result in the use of a black background, in order to ensure legibility of the content. *end example*] [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore can be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§17.18.38).

Attributes	Description
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which must appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" … />` This frame's val is `true`, indicating that the border frame effect must be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
id (Custom Defined Border Relationship Reference) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID for the relationship which contains the custom border image for the parent element. This custom border image is contained in a separate part within the WordprocessingML package. The relationship explicitly targeted by this attribute shall be of type http://purl.oclc.org/ooxml/officeDocument/relationships/image or the document shall be considered non-conformant. If this attribute is omitted, then no custom border shall be used. [*Example*: Consider the following WordprocessingML markup for a custom bottom border in a document: `<w:bottom w:val="custom" r:id="rIdCustomBottomBorder" …/>` The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rIdCustomBottomBorder` must contain the custom bottom border image for the document. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which must appear with a shadow effect, resulting in

Attributes	Description
	the following WordprocessingML: `<w:bottom w:shadow="true" … />` This frame's val is `true`, indicating that the shadow effect must be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

space (Border Spacing Measurement)	Specifies the spacing offset that shall be used to place this border on the parent object.
	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§17.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§17.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24" />`
	`</w:pgBorders>`
	The offsetFrom attribute specifies that the space value provides the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset must be 24 points. *end example*]
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§17.18.68).

sz (Border Width)	Specifies the width of the current border. If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range can be reassigned to a more appropriate value. If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range can be reassigned to a more appropriate value. [*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:

Attributes	Description

	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (dashed), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple type (§17.18.23).

themeColor (Border Theme Color)	Specifies the base theme color used to generate the border color. The border color is the RGB value associated with themeColor as further transformed by themeTint or themeShade (if one is present), else the background color is the RGB value associated with themeColor.
	
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part (§14.2.7 and §20.1.6.9), which allows color information to be set centrally in the document.
	
	To determine the color to display, the following actions are performed:
	- Using the mapping specified in the ST_ThemeColor simple type (§17.18.97), the appropriate attribute on the clrSchemeMapping element (§17.15.1.20) is read.
	- Using that value and the mapping specified in the ST_ColorSchemeIndex simple type (§17.18.103), the appropriate element in the document's Theme part is read to get the base theme color.
	- The specified color is modified based on the presence of the themeTint or themeShade attribute.
	
	[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:
	
	`<w:top … w:themeColor="accent2" w:themeTint="99" />`
	`<w:bottom … w:themeColor="accent2" w:themeTint="99" />`
	`<w:left … w:themeColor="accent2" w:themeTint="99" />`
	`<w:right … w:themeColor="accent2" w:themeTint="99" />`
	
	If the Settings part contained the following markup:
	
	`<w:clrSchemeMapping … w:accent2="accent2"/>`
	
	and the Theme part contained the following XML markup:
	
	`<a:accent2>`
	`  <a:srgbClr val="4F81BD"/>`

Attributes	Description
	`</a:accent2>` the resulting border color would be 95B3D7 (the result of a 60% tint applied to the original theme color; see the calculations in themeTint below for details). *end example*] The possible values for this attribute are defined by the ST_ThemeColor simple type (§17.18.97).
themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeColor attribute is not present, then this attribute shall not be used. If the themeShade is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border. [*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows: $S_{xml} = 0.4 * 255$ $= 102$ $= 66(hex)$ The resulting themeShade value in the file format would be 66. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the accent2 theme color, whose RGB value (in RRGGBB hex format) is C0504D. The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48{,}0.53\right)$. Applying the shade formula with a shade percentage of 75% to the luminance, we get: $L' = 0.53 * 0.75$ $= 0.39698$ Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48{,}0.39698\right)$and converting back to RGB,

Attributes	Description
	we get 943634. This transformed value can be seen in the resulting background's color attribute: `<w:top w:val="single" w:sz="4" w:space="24" w:color="943634" w:themeColor="accent2" w:themeShade="BF"/>` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).
themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeColor attribute is not present, then this attribute shall not be used. If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border. The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border. [*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows: $$\begin{aligned} T_{xml} &= 0.6 * 255 \\ &= 153 \\ &= 99(hex) \end{aligned}$$ The resulting themeTint value in the file format would be 99. *end example*] Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows: • Convert the color to the HSL color format (values from 0 to 1) • Modify the luminance factor as follows: $$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$$ • Convert the resultant HSL color to RGB [*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is 4F81BD. The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45,0.53\right)$. Applying the tint formula with a tint percentage of 60% to the luminance, we get:

Attributes	Description

	$\begin{aligned} L' &= 0.53 * 0.6 + (1 - .6) \\ &= 0.71 \end{aligned}$
	Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45, 0.71\right)$ and converting back to RGB, we get `95B3D7`.
	This transformed value can be seen in the resulting background's color attribute:
	`<w:top w:val="single" w:sz="4" w:space="24"`
	`  w:color="95B3D7" w:themeColor="accent2"`
	`  w:themeTint="99"/>`
	end example]
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - shall only be used for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§17.18.2).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageBorder) is located in §A.1. *end note*]

17.6.16 rtlGutter (Gutter on Right Side of Page)

This element specifies that the page gutter shall be placed on the right side of the page for this section only. The *page gutter* defines the amount of extra space added to the specified margin, above any existing margin values. [*Note*: This setting is typically used when a document is being created for binding, in order to ensure that the resulting margins are present after the binding gutter is consumed by the printed matter binding. *end note*]

If the gutter is set to the side of the page by the omission of the gutterAtTop element (§17.15.1.50), then each section's gutter is placed at the left by default, unless that default is overridden by the rtlGutter element.

[*Example*: Consider a document with three sections, with gutter properties defined as follows:

```
<w:p>
  <w:pPr>
    <w:sectPr>
      <w:pgMar w:gutter="1440" …/>
      …
    </w:sectPr>
  </w:pPr>
</w:p>
…
<w:p>
  <w:pPr>
    <w:sectPr>
      <w:pgMar w:gutter="1440" …/>
      <w:rtlGutter w:val="0" />
      …
    </w:sectPr>
  </w:pPr>
</w:p>
…
<w:p>
  <w:pPr>
    <w:sectPr>
      <w:pgMar w:gutter="1440" …/>
      <w:rtlGutter />
      …
    </w:sectPr>
  </w:pPr>
</w:p>
```

The first and second sections both place the gutter on the left side, the first by omission of the rtlGutter attribute, and the second by explicitly turning it off. The third section, however, moves the gutter to the right side via the use of the rtlGutter attribute. *end example*]

If the gutterAtTop element (§17.15.1.50) is specified and true, then each section's gutter is at the top and this setting is ignored.

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.6.17 sectPr (Document Final Section Properties)

This element defines the section properties for the final section of the document. [*Note*: For any other section the properties are stored as a child element of the paragraph element corresponding to the last paragraph in the given section. *end note*]

[*Example*: Consider a document with multiple sections. For all sections except the final section, the sectPr element is stored as a child element of the last paragraph in the section. For the final section, this information is stored as the last child element of the body element, as follows:

```
<w:body>
  <w:p>
  …
  </w:p>
  …
  <w:sectPr>
    (final section's properties)
  </w:sectPr>
</w:body>
```

end example]

Attributes	Description
rsidDel (Section Deletion Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was deleted from the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer can choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidR (Section Addition Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was added to the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer can choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidRPr (Physical Section Mark Character Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session.

Attributes	Description
	A producer can choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidSect (Section Properties Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer can choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SectPr) is located in §A.1. *end note*]

17.6.18 sectPr (Section Properties)

This element defines the section properties for the a section of the document. [*Note*: For the last section in the document, the section properties are stored as a child element of the body element. *end note*]

[*Example*: Consider a document with multiple sections. For all sections except the final section, the sectPr element is stored as a child element of the last paragraph in the section, as follows:

```
<w:body>
  <w:p>
    <w:pPr>
      <w:sectPr>
        (final section's properties)
      </w:sectPr>
    </w:pPr>
  …
  </w:p>
  …
  <w:sectPr>
    (final section's properties)
  </w:sectPr>
</w:body>
```

end example]

Attributes	Description
rsidDel (Section Deletion Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was deleted from the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer can choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidR (Section Addition Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was added to the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer can choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidRPr (Physical Section Mark Character Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer can choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidSect (Section Properties Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer can choose to increment the revision save ID value to indicate subsequent

Attributes	Description
	editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SectPr) is located in §A.1. *end note*]

17.6.19 sectPr (Previous Section Properties)

When specified as a child element of sectPrChange, the sectPr element specifies a set of section properties that were modified when the document was set to track all revisions.

[*Example*: If the page orientation was changed with revision tracking enabled, the following WordprocessingML defines the contents of that change:

```
<w:sectPr>
  …
  <w:sectPrChange …>
    <w:sectPr>
      <w:pgSz w:w="15840" w:h="12240"/>
    </w:sectPr>
  </w:sectPrChange>
</w:sectPr>
```

The properties that were changed as part of this revision are stored in this sectPr element. *end example*]

Attributes	Description
rsidDel (Section Deletion Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was deleted from the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer can choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidR (Section Addition Revision ID)	Specifies a unique identifier used to track the *editing session* when the section mark for this section was added to the document. All rsid* attributes throughout this document of an equal value, if present, shall indicate

Attributes	Description
	that those regions were modified during the same editing session. A producer can choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidRPr (Physical Section Mark Character Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer can choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).
rsidSect (Section Properties Revision ID)	Specifies a unique identifier used to track the editing session when the physical character representing this section mark was last formatted. All rsid* attributes throughout this document of an equal value, if present, shall indicate that those regions were modified during the same editing session. A producer can choose to increment the revision save ID value to indicate subsequent editing sessions (editing between save actions) to indicate the order of the saves performed. The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_SectPrBase) is located in §A.1. *end note*]

17.6.20 textDirection (Text Flow Direction)

This element specifies the direction of the text flow for this section.

[*Example*: Consider a document with a section in which text be oriented vertically, flowing from left to right horizontally on the page. This setting requires the following WordprocessingML:

```
<w:sectPr>
  …
  <w:textDirection w:val="lr" />
</w:sectPr>
```

The textDirection element specifies via the lr value in the val attribute that the text flow be oriented vertically, with subsequent lines stacked from left to right. *end example*]

Attributes	Description

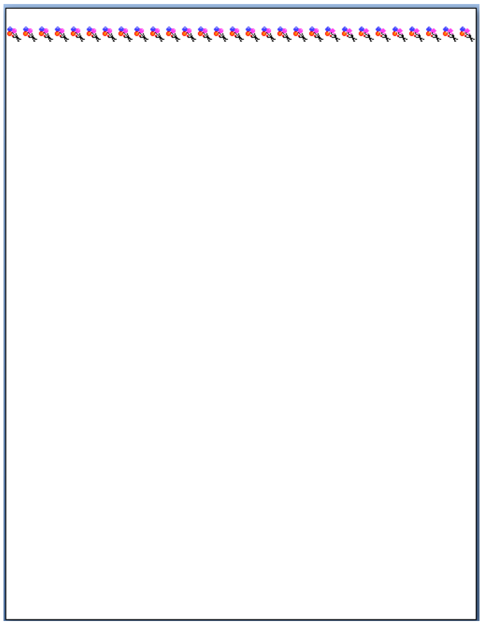
val (Direction of Text Flow)	Specifies the direction of the text flow for this object.
	[*Example*: Consider a document with a section in which text must be oriented vertically, flowing from left to right horizontally on the page. This setting requires the following WordprocessingML:
	`<w:sectPr>`
	`  …`
	`  <w:textDirection w:val="lr" />`
	`</w:sectPr>`
	The textDirection element specifies via the lr value in the val attribute that the text flow must be oriented vertically, with subsequent lines stacked from left to right.*end example*]
	The possible values for this attribute are defined by the ST_TextDirection simple type (§17.18.93).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextDirection) is located in §A.1. *end note*]

17.6.21 top (Top Border)

This element specifies the presentation and display of the page border displayed at the top of each page in this section.

[*Example*: Consider a section in which all pages must have a top border consisting of a repeated image of balloons, like this:



This border would result in the following WordprocessingML:

```
<w:sectPr>
  …
  <w:pgBorders>
    <w:top w:val="balloons3Colors" …/>
  </w:pgBorders>
  …
</w:sectPr>
```

Because the page only has a border at the top, only the top element is specified within the set of page borders. *end example*]

When a document has a top border that is relative to the page edges (using an offsetFrom attribute value of page on pgBorders), it shall span the top edge of the page at the location defined by its properties, stopping when:

- It intersects with the corresponding left or right page border (if one is specified)
- It reaches the edge of the page.

[*Example*: In the example above, no left or right border was specified in the WordprocessingML, so a consumer must draw the border from one edge of the page to the other. *end example*]

When a document has a top border that is relative to the text (using the offsetFrom attribute value of text on pgBorders), it shall only span the necessary width to satisfy the requirement of spanning the width of the text.

When a document has custom border art specified by attributes topLeft, topRight, and/or id, it shall use the corresponding relationship part item as an image for the top left corner, top right corner, and/or top border, respectively. If the corresponding relationship part item cannot be resolved the consumer shall use the border specified by the value of the val attribute. If the corresponding value of the val attribute cannot be resolved no top left corner, top right corner, or top border is present when the page is displayed.

When a document has a custom border art specified by attribute id without specifying either attributes topRight and/or topLeft, the top border as resolved by the corresponding relationship part item of attribute id shall span to the corners not specified by topRight and/or topLeft attributes.

Attributes	Description
color (Border Color)	Specifies the color for this border. This value can be defined as either: • A color value using the RGB color model whose red, green, and blue values are written as numbers in the range 0 to 255, hex encoded, and concatenated. [*Example*: Full intensity red would be 255 red, 0 green, 0 blue, encoded to FF, 00, 00, and concatenated to `FF0000`. *end example*] . RGB colors are specified in the sRGB color space. • `auto` to allow a consumer to automatically determine the border color in order to make the document's text readable. [*Example*: A document with white text and a background color of `auto` might result in the use of a black background, in order to ensure legibility of the content. *end example*] [*Example*: Consider a border color with value `auto`, as follows: `<w:bottom … w:color="auto"/>` This color therefore can be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*] If the border style (the val attribute) specifies the use of an art border, this attribute is ignored. As well, if the border specifies the use of a theme color via the themeColor attribute, this value is superseded by the theme color value. The possible values for this attribute are defined by the ST_HexColor simple type (§17.18.38).
frame (Create Frame Effect)	Specifies whether the specified border should be modified to create a frame effect by reversing the border's appearance from the edge nearest the text to the edge furthest from the text. If this attribute is omitted, then the border is not given any frame effect. [*Example*: Consider a bottom border which must appear with a frame effect, which is specified in the following WordprocessingML: `<w:bottom w:frame="true" … />`

Attributes	Description
	This frame's val is `true`, indicating that the border frame effect must be applied. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
id (Custom Defined Border Relationship Reference) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID for the relationship which contains the custom border image for the parent element. This custom border image is contained in a separate part within the WordprocessingML package. The relationship explicitly targeted by this attribute shall be of type http://purl.oclc.org/ooxml/officeDocument/relationships/image or the document shall be considered non-conformant. If this attribute is omitted, then no custom border shall be used. [*Example*: Consider the following WordprocessingML markup for a custom bottom border in a document: `<w:bottom w:val="custom" r:id="rIdCustomBottomBorder" …/>` The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rIdCustomBottomBorder` must contain the custom bottom border image for the document. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
shadow (Border Shadow)	Specifies whether this border should be modified to create the appearance of a shadow. For the right and bottom borders, this is accomplished by duplicating the border below and right of the normal border location. For the right and top borders, this is accomplished by moving the order down and to the right of its original location. If this attribute is omitted, then the border is not given the shadow effect. [*Example*: Consider a top border which must appear with a shadow effect, resulting in the following WordprocessingML: `<w:bottom w:shadow="true" … />` This frame's val is `true`, indicating that the shadow effect must be applied to the border. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
space (Border Spacing	Specifies the spacing offset that shall be used to place this border on the parent object.

Attributes	Description

Measurement)	When a document has a page border that is relative to the page edges (using a value of page in the offsetFrom attribute on pgBorders (§17.6.10)), it shall specify the distance between the edge of the page and the beginning of this border in points.
	
	When a document has a page border that is relative to the text extents (using a value of `text` in the offsetFrom attribute on pgBorders (§17.6.10)), or any other border type, it shall specify the distance between the edge of the object and the beginning of this border in points.
	
	[*Example*: Consider a document with a set of page borders all specified to appear 24 points from the edge of the page. The resulting WordprocessingML would be as follows:
	
	`<w:pgBorders w:offsetFrom="page">`
	`  <w:bottom … w:space="24" />`
	`</w:pgBorders`
	
	The offsetFrom attribute specifies that the space value provides the offset of the page border from the page edge, and the value of the space attribute specifies that the page offset must be 24 points. *end example*]
	
	The possible values for this attribute are defined by the ST_PointMeasure simple type (§17.18.68).

sz (Border Width)	Specifies the width of the current border.
	
	If the border style (val attribute) specifies a line border, the width of this border is specified in measurements of eighths of a point, with a minimum value of two (one-fourth of a point) and a maximum value of 96 (twelve points). Any values outside this range can be reassigned to a more appropriate value.
	
	If the border style (val attribute) specifies an art border, the width of this border is specified in measurements of points, with a minimum value of one and a maximum value of 31. Any values outside this range can be reassigned to a more appropriate value.
	
	[*Example*: Consider a document with a three point wide dashed line border on all sides, resulting in the following WordprocessingML markup:
	
	`<w:top w:val="dashed" w:sz="24" …/>`
	`<w:left w:val="dashed" w:sz="24" …/>`
	`<w:bottom w:val="dashed" w:sz="24" …/>`
	`<w:right w:val="dashed" w:sz="24" …/>`
	
	The border style is specified using the val attribute, and because that border style is a line border (`dashed`), the sz attribute specifies the size in eighths of a point (24 eighths of a point = 3 points). *end example*]
	
	The possible values for this attribute are defined by the ST_EighthPointMeasure simple

Attributes	Description
	type (§17.18.23).

themeColor (Border Theme Color)	Specifies the base theme color used to generate the border color. The border color is the RGB value associated with themeColor as further transformed by themeTint or themeShade (if one is present), else the background color is the RGB value associated with themeColor.
	The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part (§14.2.7 and §20.1.6.9), which allows color information to be set centrally in the document.
	To determine the color to display, the following actions are performed:
	• Using the mapping specified in the ST_ThemeColor simple type (§17.18.97), the appropriate attribute on the clrSchemeMapping element (§17.15.1.20) is read.
	• Using that value and the mapping specified in the ST_ColorSchemeIndex simple type (§17.18.103), the appropriate element in the document's Theme part is read to get the base theme color.
	• The specified color is modified based on the presence of the themeTint or themeShade attribute.
	[*Example*: Consider a set of borders configured to use the `accent2` theme color, resulting in the following WordprocessingML markup:
	`<w:top … w:themeColor="accent2" w:themeTint="99" />`
	`<w:bottom … w:themeColor="accent2" w:themeTint="99" />`
	`<w:left … w:themeColor="accent2" w:themeTint="99" />`
	`<w:right … w:themeColor="accent2" w:themeTint="99" />`
	If the Settings part contained the following markup:
	`<w:clrSchemeMapping … w:accent2="accent2"/>`
	and the Theme part contained the following XML markup:
	`<a:accent2>`
	`  <a:srgbClr val="4F81BD"/>`
	`</a:accent2>`
	the resulting border color would be `95B3D7` (the result of a 60% tint applied to the original theme color; see the calculations in themeTint below for details). *end example*]
	The possible values for this attribute are defined by the ST_ThemeColor simple type (§17.18.97).

themeShade (Border Theme Color Shade)	Specifies the shade value applied to the supplied theme color (if any) for this border instance. If the themeColor attribute is not present, then this attribute shall not be used. If the themeShade is supplied, then it is applied to the RGB value of the theme color

Attributes	Description

	(from the theme part) to determine the final color applied to this border.
	The themeShade value is stored as a hex encoding of the shade value (from 0–255) applied to the current border.
	[*Example*: Consider a shade of 40% applied to a border in a document. This shade is calculated as follows:
	$\begin{aligned} S_{xml} &= 0.4 * 255 \\ &= 102 \\ &= 66(hex) \end{aligned}$
	The resulting themeShade value in the file format would be `66`. *end example*]
	Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows:
	• Convert the color to the HSL color format (values from 0 to 1)
	• Modify the luminance factor as follows:
	$\mathrm{L}' = \mathrm{L} * \mathrm{Shade}_{\mathrm{percentage}}$
	• Convert the resultant HSL color to RGB
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `C0504D`.
	The equivalent HSL color value would be $\left(\frac{1}{360}, 0.48,0.53\right)$.
	Applying the shade formula with a shade percentage of 75% to the luminance, we get:
	$\begin{aligned} L' &= 0.53 * 0.75 \\ &= 0.39698 \end{aligned}$
	Taking the resulting HSL color value of $\left(\frac{1}{360}, 0.48,0.39698\right)$and converting back to RGB, we get `943634`.
	This transformed value can be seen in the resulting background's color attribute:
	`<w:top w:val="single" w:sz="4" w:space="24"`
	`  w:color="943634" w:themeColor="accent2"`
	`  w:themeShade="BF"/>`
	end example]
	The possible values for this attribute are defined by the ST_UcharHexNumber simple

Attributes	Description
	type (§17.18.98).

themeTint (Border Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this border instance. If the themeColor attribute is not present, then this attribute shall not be used.
	If the themeTint is supplied, then it is applied to the RGB value of the theme color (from the theme part) to determine the final color applied to this border.
	The themeTint value is stored as a hex encoding of the tint value (from 0–255) applied to the current border.
	[*Example*: Consider a tint of 60% applied to a border in a document. This tint is calculated as follows:
	$T_{xml} = 0.6 * 255$
	$= 153$
	$= 99(hex)$
	The resulting themeTint value in the file format would be 99. *end example*]
	Given an RGB color defined as three hex values in RRGGBB format, the shade is applied as follows:
	- Convert the color to the HSL color format (values from 0 to 1)
	- Modify the luminance factor as follows:
	$\mathrm{L}' = \mathrm{L} * \mathrm{Tint}_{\mathrm{pct}} + \left(1 - \mathrm{Tint}_{\mathrm{pct}}\right)$
	- Convert the resultant HSL color to RGB
	[*Example*: Consider a document with a background using the `accent2` theme color, whose RGB value (in RRGGBB hex format) is `4F81BD`.
	The equivalent HSL color value would be $\left(\frac{213}{360}, 0.45,0.53\right)$.
	Applying the tint formula with a tint percentage of 60% to the luminance, we get:
	$L' = 0.53 * 0.6 + (1 - .6)$
	$= 0.71$
	Taking the resulting HSL color value of $\left(\frac{213}{360}, 0.45,0.71\right)$and converting back to RGB, we get `95B3D7`.
	This transformed value can be seen in the resulting background's color attribute:
	`<w:top w:val="single" w:sz="4" w:space="24"`
	`  w:color="95B3D7" w:themeColor="accent2"`

Attributes	Description

	`w:themeTint="99"/>`
	end example]
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

topLeft (Custom Defined Top Left Border Relationship Reference) Namespace: …/officeDocument/2006/relationships	Specifies the relationship ID for the relationship which contains the custom top left border image for the parent element. This custom border image is contained in a separate part within the WordprocessingML package. The relationship explicitly targeted by this attribute shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/image or the document shall be considered non-conformant. If this attribute is omitted, then no custom top left border shall be used. [*Example*: Consider the following WordprocessingML markup for a custom top left border in a document: `<w:top w:val="custom" r:topLeft="rIdCustomTopLeftBorder" …/>` The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rIdCustomTopLeftBorder` must contain the custom top left border image for the document. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
topRight (Custom Defined Top Right Border Relationship Reference) Namespace: …/officeDocument/2006/relationships	Specifies the relationship ID for the relationship which contains the custom top right border image for the parent element. This custom border image is contained in a separate part within the WordprocessingML package. The relationship explicitly targeted by this attribute shall be of type http://schemas.openxmlformats.org/officeDocument/2006/relationships/image or the document shall be considered non-conformant. If this attribute is omitted, then no custom top left border shall be used when the parent element is instantiated. [*Example*: Consider the following WordprocessingML markup for a custom top right border in a document: `<w:top w:val="custom" r:topRight="rIdCustomTopRightBorder" … />` The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rIdCustomTopRightBorder` must contain the custom top right

Attributes	Description
	border image for the document. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
val (Border Style)	Specifies the style of border used on this object. This border can either be an art border (a repeated image along the borders - shall only be used for page borders) or a line border (a line format repeated along the borders) - see the simple type definition for a description of each border style. [*Example*: Consider a left border resulting in the following WordprocessingML: `<w:left w:val="single" …/>` This border's val is `single`, indicating that the border style is a single line. *end example*] The possible values for this attribute are defined by the ST_Border simple type (§17.18.2).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TopPageBorder) is located in §A.1. *end note*]

17.6.22 type (Section Type)

This element specifies the section type of the current section. The section type specifies how the contents of the current section shall be placed relative to the previous section.

WordprocessingML supports five distinct types of section breaks:

- *Next page section breaks* (the default if type is not specified), which begin the new section on the following page.
- *Odd page section breaks*, which begin the new section on the next odd-numbered page.
- *Even page section breaks*, which begin the new section on the next even-numbered page.
- *Continuous section breaks*, which begin the new section on the following paragraph. This means that continuous section breaks might not specify certain page-level section properties, since they shall be inherited from the following section. These breaks, however, can specify other section properties, such as line numbering and footnote/endnote settings.
- *Column section breaks*, which begin the new section on the next column on the page.

Attributes	Description
val (Section Type Setting)	Specifies the section type of the current section. [*Example*: Consider a section that must start on the next page in the document. The

Attributes	Description

	WordprocessingML specifying this would look like:
	`<w:sectPr>`
	`  …`
	`  <w:type w:val="nextPage"/>`
	`</w:sectPr>`
	The nextPage value specifies that this section starts on the next page. *end example*]
	The possible values for this attribute are defined by the ST_SectionMark simple type (§17.18.77).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SectType) is located in §A.1. *end note*]

17.6.23 vAlign (Vertical Text Alignment on Page)

This element specifies the vertical alignment for text on pages in the current section, relative to the top and bottom margins in the main document story on each page.

[*Example*: Consider a section used as a title page on which text must be vertically centered. In order to center the text vertically on the page, the following WordprocessingML is used:

```
<w:sectPr>
  …
  <w:vAlign w:val="center" />
</w:sectPr>
```

The vAlign value of center specifies that text must be laid out in the center of the top and bottom text margins for all pages in this section. *end example*]

Attributes	Description
val (Vertical Alignment Setting)	Specifies the vertical alignment for text between the top and bottom margins of the parent container (page or table cell). [*Example*: Consider a region where the text must be vertically centered in the parent element. This would require a val value of center, in order to specify that all justification vertically must be centered relative to the parent. For a section, this setting would be specified as follows: `<w:vAlign w:val="center" />` The val attribute of center specifies that the content is centered relative to its container (in this case, the page). *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_VerticalJc simple type (§17.18.101).

[*Note*: The W3C XML Schema definition of this element's content model (CT_VerticalJc) is located in §A.1. *end note*]

17.7 Styles

Within a WordprocessingML file, *styles* are predefined sets of table, numbering, paragraph, and/or character properties which can be applied to text within the document. This allows the formatting properties to be stored and managed independently from the content, allowing the look of document content to be changed in a single location (e.g. the look of all first-level headings is changed by changing the style with styleId Heading1 rather than looking for and changing each paragraph in the document).

[*Example*: The Normal paragraph style in a word processing document can have any number of formatting properties, e.g. font face = Times New Roman; font size = 12pt; paragraph justification = left). All paragraphs which reference this paragraph style would automatically inherit these properties. *end example*]

Each style defined within a WordprocessingML document requires a *style definition*. The style definition contains all of the information needed by a consumer to store and display that style within a WordprocessingML document, and is defined using the style element. The style definition for any style in WordprocessingML can be divided into three segments The complete definition of style properties can be found on the reference for the style element (§17.7.4.17):

- General style properties
- Style types
- Type specific formatting properties

Each of these three segments are discussed in the following subclauses.

17.7.1 Style Inheritance

In order to compile the complete set of paragraph and character properties specified by any given style (as appropriate), a consumer shall follow the rule of style inheritance to determine each property in that set.

Style inheritance states that styles of any given style type can inherit from other styles of that style type, and therefore a consumer shall 'build up' the style information by following the inheritance tree. This inheritance is defined via the basedOn element, which specifies the styleId of the parent style.

[*Example*: The "Tristan Test" paragraph style can inherit properties from the "Heading 1" paragraph style, which itself can inherit properties from the "Normal" paragraph style. *end example*]

To build up the resulting style, a consumer shall trace the hierarchy (following each basedOn value) back to a style which has no basedOn element (is not based on another style). The resulting style is then constructed by following each level in the tree, applying the specified paragraph and/or character properties as appropriate. When properties conflict, they are overridden by each subsequent level (this includes turning OFF a property set at an earlier level). Properties which are not specified simply do not change those specified at earlier levels.

[*Example*: Consider a character style Green which specifies only that the text color is green, but inherits from another character style Base which defines a font face of Arial, as well as bold:

```
<w:style w:type="character" w:styleId="Green">
  <w:name w:val="Green" />
  <w:basedOn w:val="Base" />
  <w:rPr>
    <w:color w:val="22B14C" />
  </w:rPr>
</w:style>
…
  <w:style w:type="character" w:styleId="Base">
  <w:name w:val="Base" />
  <w:rPr>
    <w:rFonts w:ascii="Arial" w:hAnsi="Arial" />
    <w:b />
  </w:rPr>
</w:style>
```

The definition of the Green character style has a basedOn element which specifies the Base style. This means that any use of the Green style is defined as bold, green, Arial text. *end example*]

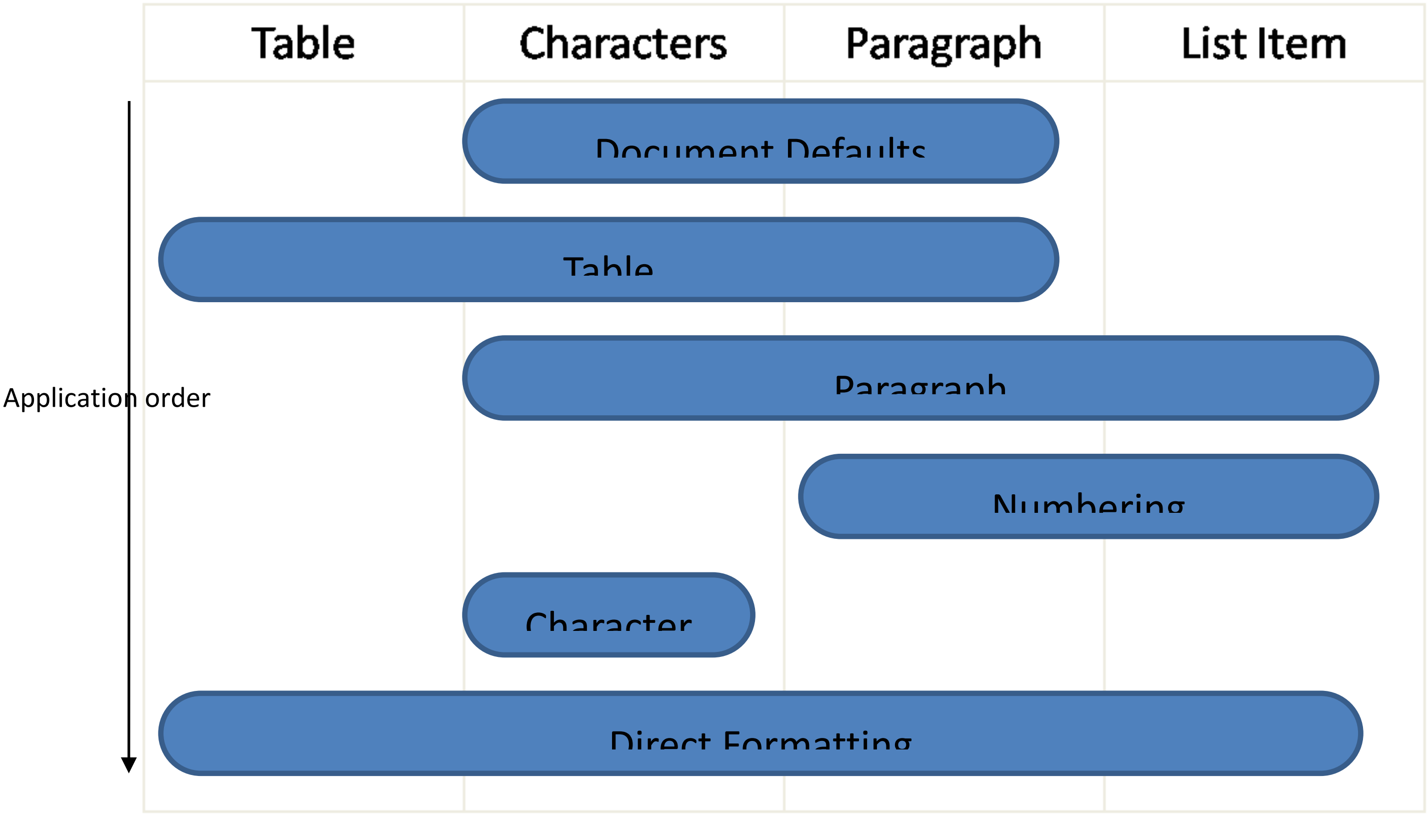
Conversely, a producer should not output any property on a style which has already been set by a previous level of the style hierarchy, as well as those which match the document defaults. This means that if the document defaults or any previous level in a style's hierarchy specify a property which is unchanged at this level, that property should not be part of the style definition in the resulting WordprocessingML.

[*Example*: If the document default font is Bauhaus 93 and the Heading 1 style also specifies the Bauhaus 93 font, then a producer should not output any rFonts element for the Heading 1 style definition, because that formatting is inherited from the document defaults. *end example*]

17.7.2 Style Hierarchy

With the various flavors of styles available (see each of the subclauses below), multiple style types can be applied to the same content within a file, which means that properties shall be applied in a specific deterministic order. As with inheritance, the resulting formatting properties set by one style type can be unchanged, removed, or altered by following style types.

The following table illustrates the order of application of these defaults, and which properties are impacted by each:



This process can be described as follows:

- First, the document defaults are applied to all runs and paragraphs in the document.
- Next, the table style properties are applied to each table in the document, following the conditional formatting inclusions and exclusions specified per table.
- Next, numbered item and paragraph properties are applied to each paragraph formatted with a numbering style.
- Next, paragraph and run properties are applied to each paragraph as defined by the paragraph style.
- Next, run properties are applied to each run with a specific character style applied.
- Finally, we apply direct formatting (paragraph or run properties not from styles). If this direct formatting includes numbering, that numbering + the associated paragraph properties are applied.

If the value of the rFonts element (§17.3.2.26) references a font which is not available, applications determine a suitable alternative font via a process called font substitution, which is defined in §17.8.2.

17.7.3 Toggle Properties

Certain character properties defined in §17.3.2 are specified as toggle properties. [*Example*: the Bold and Italics properties are toggle properties. *end example*] As indicated in the previous two sections (§17.7.1 and §17.7.2) several styles can affect the formatting applied to a given piece of content within a WordprocessingML document. When the same formatting property appears in one or more styles that affect the content applied to a run, the combined effect depends on whether or not the formatting property is a toggle property.

If the property is not a toggle property, then its values shall be applied in the order described in §17.7.1 and §17.7.2, and only its last value in that order shall be used.

If the property is a toggle property, then its values, which are limited to `true` and `false` (or the equivalent values `1` and `0`) shall be combined as follows:

- If a toggle property is explicitly set in direct formatting applied to a given piece of content, then its value in the direct formatting shall be used.
- Otherwise, the instances of that toggle property in the styles that affect the content shall be combined in the following manner:

    - If multiple instances of the toggle property appear at the same level of the style hierarchy, then the first value encountered by the following algorithm shall be used (if no value is encountered, the property takes on its default value).

        - Attempt to read the value in the style.
        - If it does not exist and the style has a basedOn element with a non-empty value, repeat step 1 using the style specified by the basedOn element.
        - [*Example*: If a paragraph style sets no value for the bold property to false and the paragraph style specified by its basedOn element specifies that it is true, the result of applying the style definition sets the value of bold to true (the first value in the hierarchy). *end example*]

    - If the value of the toggle property appears at multiple levels of the style hierarchy (§17.7.2), their effective values shall be combined as follows:

        - If the value specified by the document defaults is true, the effective value is true.
        - Otherwise, the values are combined by a Boolean XOR as follows:
          $value_{effective} = val_{table}\ XOR\ val_{paragraph}\ XOR\ val_{character}$
          i.e., the effective value to be applied to the content shall be true if its effective value is true for an odd number of levels of the style hierarchy.

The following Boolean properties are toggle properties: §17.3.2.1 (Bold), §17.3.2.2 (Complex Script Bold), §17.3.2.5 (Display All Characters as Capital Letters), §17.3.2.13 (Embossing), §17.3.2.16 (Italics), §17.3.2.17 (Complex Script Italics), §17.3.2.18 (Imprinting), §17.3.2.23 (Display Character Outline), §17.3.2.31 (Shadow), §17.3.2.33 (Small Caps), §17.3.2.37 (Single Strikethrough), §17.3.2.41 (Hidden Text).

[*Example*: Consider a table style with two styles in its basedOn chain. If the resolved value of the bold property (a toggle property) within the basedOn chain of the table style is true, that specifies that this property should be applied to the contents of the table:

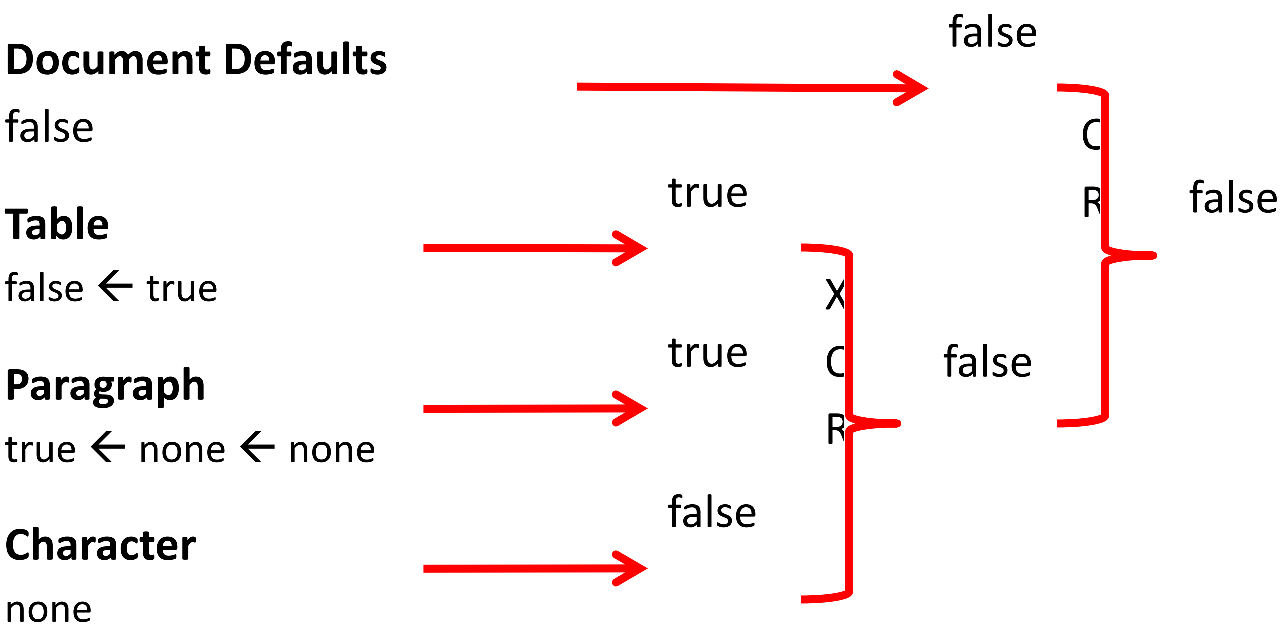
Cell 1,1	**Cell 1,2**
Cell 2,1	**Cell 2,2**

©ISO/IEC 2012 – All rights reserved

If a single paragraph within that table also has a paragraph style applied, with three styles in its basedOn chain that resolve to a value of true, the toggle property logic above would toggle the bold property, resulting in bold not being applied to its contents. Applying this to the paragraph in the first cell below, the resulting table would appear as follows:

Cell 1,1	**Cell 1,2**
Cell 2,1	**Cell 2,2**

The calculation which results in this value for the bold property is displayed below:



← symbolizes the traversal of a basedOn reference

end example]

17.7.4 General Style Properties

General style properties refer to the set of properties which can be used regardless of the type of style. [*Example*: Within a style definition the style name, additional aliases for the style, a style ID (used by the document content to refer to the style), if style is hidden, if style is locked, etc. are general style properties. *end example*]

[*Example:* Consider a style called Heading 1 in a document as follows:

```
<w:style w:type="paragraph" w:styleId="Heading1">
  <w:name w:val="Heading 1"/>
  <w:basedOn w:val="Normal"/>
  <w:next w:val="Normal"/>
  <w:link w:val="Heading1Char"/>
  <w:uiPriority w:val="1"/>
  <w:qFormat/>
  <w:rsid w:val="00F303CE"/>
  …
</w:style>
```

Above the formatting information specific to this style type are a set of general style properties which define information shared by all style types. *end example*]

17.7.4.1 aliases (Alternate Style Names)

This element specifies the set of alternative names for the parent style definition. These names can be used in an application's user interface as desired. The alternate names shall be stored in this element's val attribute, and each name shall be separated by one or more consecutive comma characters (Unicode character value 002C). All commas present shall be interpreted as separator character and never as part of an alternate style name.

If present, the alternate style names shall be used in the user interface in place of the built-in name specified in the name element (§17.7.4.9) when the appropriate value is set in the stylePaneFormatFilter element (§17.15.1.86).

If this element is omitted, then the style shall not have any alternate style names.

[*Example*: Consider a style with a primary name and two alternate names, defined using the name and aliases elements, as follows:

```
<w:style w:styleId="TestStyle" … >
  <w:name w:val="GD20Complex"/>
  <w:aliases w:val="Regional Growth,Complex Growth"/>
  …
</w:style>
```

This style specifies that it has the primary name GD20Complex using the name element (§17.7.4.9), as well as two alternate names Regional Growth and Complex Growth using the aliases element. *end example*]

Attributes	Description
val (String Value)	Specifies that its contents contain a string. The contents of this string are interpreted based on the context of the parent XML element.

Attributes	Description

	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.7.4.2 autoRedefine (Automatically Merge User Formatting Into Style Definition)

This element specifies whether an application shall automatically modify this style when the contents of an entire paragraph in the document with this style applied are modified, ensuring that although only a single instance of text with this style was modified, that change is stored on the style and therefore propagated to all locations where the style is in use.

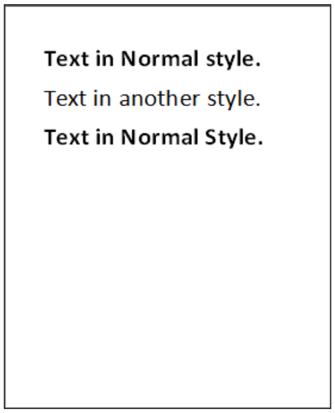
If this element is omitted, then formatting shall not automatically be merged back into the style definition.

[*Example*: Consider a style defined as follows in a WordprocessingML document:

```
<w:style w:styleId="Normal" … >
  <w:name w:val="Normal"/>
  <w:autoRedefine/>
  <w:rPr>
    <w:b/>
  </w:rPr>
  …
</w:style>
```

This style specifies via the use of the autoRedefine element that any formatting applied to text which uses this style must be merged back into the style definition (assuming, of course, that this is a paragraph style).

For example, consider a document which uses the Normal style as defined above:

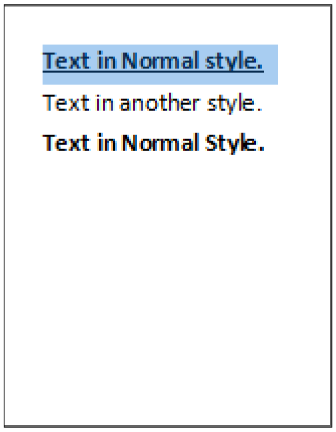


Text in Normal style.

Text in another style.

Text in Normal Style.

The first and third paragraphs use the Normal style, and hence have the bold property applied. If an application were to add the underline formatting to the entire first paragraph, as follows:



Text in Normal style.

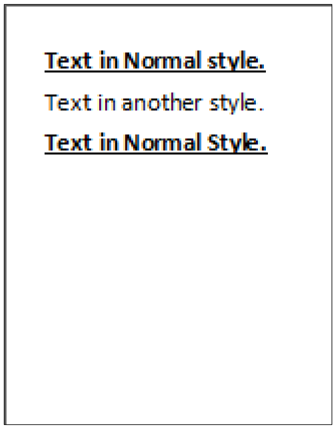
Text in another style.

Text in Normal Style.

That property, rather than being saved as direct formatting, must be used to update the associated Normal style to add this property, specified using the u element (§17.3.2.40).

```
<w:style w:styleId="Normal" … >
  <w:name w:val="Normal"/>
  <w:autoRedefine/>
  <w:rPr>
    <w:b/>
    <w:u/>
  </w:rPr>
  …
</w:style>
```

Since this property is automatically merged into the style, it would also appear on the third paragraph (note that the step above would normally be automatically modified into the state shown below, and not discrete as shown above).



Text in Normal style.

Text in another style.

Text in Normal Style.

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.7.4.3 basedOn (Parent Style ID)

This element specifies the style ID of the parent style from which this style inherits in the style inheritance. The *style inheritance* refers to a set of styles which inherit from one another to produce the resulting set of properties for a single style. The val attribute of this element specifies the styleId attribute for the parent style in the style inheritance.

If this element is omitted, then this style shall not be based on any other style in the current document (i.e. this element is the root of the style inheritance for a style). If no style in the current document specifies the styleId present in the val attribute, then this element shall be ignored (i.e. this element is the root of the style inheritance for a style).

If a style with this styleId is present, then it shall be subject to the following restrictions:

- If the current style is a table style, then the parent style shall also be a table style, or this element shall be ignored.
- If the current style is a paragraph style, then the parent style shall also be a paragraph style, or this element shall be ignored.
- If the current style is a character style, then the parent style shall also be a character style, or this element shall be ignored.
- If the current style is a numbering style, then this element shall be ignored.

[*Example*: Consider three WordprocessingML character styles defined as follows:

- A character style with a styleId value of `Strong` whose properties consist of the bold property
- A character style with a styleId value of `Underline` whose properties consist of the underline property

- A character style with a styleId value of Emphasis whose properties consist of the italics property

Each of these character styles defines a single character formatting property. If the basedOn values for each element were defined as follows:

```
<w:style w:styleId="Strong">
  <w:basedOn w:val="Underline"/>
  …
  <w:rPr>
    <w:b/>
  </w:rPr>
</w:style>
<w:style w:styleId="Underline">
  <w:basedOn w:val="Emphasis"/>
  …
  <w:rPr>
    <w:u/>
  </w:rPr>
</w:style>
<w:style w:styleId="Emphasis">
  …
  <w:rPr>
    <w:i/>
  </w:rPr>
</w:style>
```

The Strong style is based on the Underline style which is in turn based on the Emphasis style. This means that the actual definition of the Strong style would be as follows:

- Bold
- Underline (inherited from Underline)
- Italics (inherited from Emphasis)

The style chain for the Strong style would be defined as follows:

- Emphasis
- Underline
- Strong

Similarly, the style chain for the Underline style would be defined as follows:

- Emphasis
- Underline

In each case, the style chain is the list of all styles which are combined in order to produce the entire set of properties for any given style. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.7.4.4 hidden (Hide Style From User Interface)

This element specifies whether this style shall be hidden from any and all user interfaces when this document is loaded by an application. If this element is set, then this style can be used to format content (i.e. any content which references this style shall have its properties as normal), but the style shall be hidden from all user interface associated with that application. [*Note*: This setting is typically used to hide styles which are being used internally by an application which should not be used as formatting in a typical case. *end note*]

If this element is omitted, then the style shall not be required to be hidden from the user interface.

[*Example*: Consider a style with a primary name of `InternalStyle` that should not be displayed in any user interface. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="Style2">
  <w:name w:val="InternalStyle"/>
  <w:hidden/>
  …
</w:style>
```

The hidden element specifies that this style definition shall be round-tripped with the file (since it is part of the document) but should not be displayed in any user interface associated with an application which processes this document. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.7.4.5 latentStyles (Latent Style Information)

This element specifies the properties which shall be applied to a set of latent styles for this document. *Latent styles* refer to any set of style definitions known to an application which have not been included in the current document. [*Example*: Latent styles can include additional styles known by a particular hosting application. *end example*]

When a style definition is embedded in a document, it specifies two distinct groups of properties:

- Behavior properties
- Formatting properties

Obviously, embedding all the styles known to a particular application in each document which it produces would drastically increase the file size. Latent styles provide a way to store pieces of information for the first group (behavior properties) which shall be specified for all styles known to an application without requiring the storage of the second group (formatting properties).

[*Example*: Consider a WordprocessingML document which contains text specified in one of two styles: Heading1 or Normal. Based on this, the document only needs to store the formatting properties for those two styles, saving the additional overhead which would be required to save all of the styles supported by the hosting application.

However, if the documentProtection element (§17.15.1.29) specifies that the hosting application shall prevent the use of any style whose locked element (§17.7.4.7) is set to false, then the locking state of all styles known to that application become useful and necessary to maintain the current state of the document. Using latent styles, this information can be stored without storing any formatting properties for those styles.

For example, if all styles which are not stored in the document must be locked except for the style with a primary name (§17.7.4.9) of Heading 2. This requirement would be specified using latent styles as follows:

```
<w:latentStyles … w:defLockedState="true">
  <w:lsdException w:name="Heading 2" w:locked="false"/>
</w:latentStyles>
```

The latentStyles element specifies that all latent styles known to any hosting application must have a default locking state of `true` except for any style known to the hosting application with a primary name of `Heading 2`, whose latent style definition specifies that its locked state must be `false`. *end example*]

Attributes	Description

count (Latent Style Count)	Specifies the number of known styles which shall be initialized to the current latent style defaults when this document is first processed. [*Note*: This property can be used by an application as needed to ensure that only the number of styles known when this document was created are initialized with the defaults on the parent element, and that all new known styles use their default values. *end note*]
	[*Example*: Consider a WordprocessingML document in which only the first `20` latent styles must be initialized. This requirement would be specified as follows:
	`<w:latentStyles w:count="20" … >`
	`…`
	`</w:latentStyles>`
	The count attribute specifies that `20` known styles must be initialized to the default settings when the document is first opened, and any additional styles should use the defaults defined by the application. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

defLockedState (Default Style Locking Setting)	Specifies the default setting for the locked element (§17.7.4.7) which shall be applied to any style made available by the hosting application which is not explicitly defined in the current document. This setting shall be overridden for every style for which a latent style exception (§17.7.4.8) exists.
	If this element is omitted, the default locked state for all latent styles in the current document shall be `false`.
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document must be locked. This requirement would be specified using latent styles as follows:
	`<w:latentStyles … w:defLockedState="true">`
	`…`
	`</w:latentStyles>`
	The defLockedState attribute specifies that all latent styles in the current document must have a locked element setting of `true` by default. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

defQFormat (Default Primary	Specifies the default setting for the qFormat element (§17.7.4.14) which shall be applied to any style made available by the hosting application which is not explicitly defined in

Attributes	Description

Style Setting)	the current document. This setting shall be overridden for every style for which a latent style exception (§17.7.4.8) exists.
	
	If this element is omitted, the default qFormat state for all latent styles in the current document shall be `false`.
	
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document must not be marked as primary styles. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defQFormat="false">`
	`  …`
	`</w:latentStyles>`
	
	The defQFormat attribute specifies that all latent styles in the current document must have a qFormat element setting of `false` by default. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

defSemiHidden (Default Semi-Hidden Setting)	Specifies the default setting for the semiHidden element (§17.7.4.16) which shall be applied to any style made available by the hosting application which is not explicitly defined in the current document. This setting shall be overridden for every style for which a latent style exception (§17.7.4.8) exists.
	
	If this element is omitted, the default semiHidden state for all latent styles in the current document shall be `false`.
	
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document must not be marked as semi-hidden. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defSemiHidden="false">`
	`  …`
	`</w:latentStyles>`
	
	The defSemiHidden attribute specifies that all latent styles in the current document must have a semiHidden element setting of `false` by default. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

defUIPriority (Default User Interface Priority Setting)	Specifies the default setting for the uiPriority element (§17.7.4.19) which shall be applied to any style made available by the hosting application which is not explicitly defined in the current document. This setting shall be overridden for every style for which a latent style exception (§17.7.4.8) exists. If this element is omitted, the default uiPriority state for all latent styles in the current document shall be `99`.

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document must not be marked as semi-hidden. This requirement would be specified using latent styles as follows:
	`<w:latentStyles … w:defUIPriority="10">`
	`  …`
	`</w:latentStyles>`
	The defUIPriority attribute specifies that all latent styles in the current document must have a uiPriority element setting of `10` by default. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

defUnhideWhenUsed (Default Hidden Until Used Setting)	Specifies the default setting for the unhideWhenUsed element (§17.7.4.20) which shall be applied to any style made available by the hosting application which is not explicitly defined in the current document. This setting shall be overridden for every style for which a latent style exception (§17.7.4.8) exists.
	If this element is omitted, the default unhideWhenUsed state for all latent styles in the current document shall be `false`.
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document must be hidden until they are used in the document's contents. This requirement would be specified using latent styles as follows:
	`<w:latentStyles … w:defUnhideWhenUsed="true">`
	`  …`
	`</w:latentStyles>`
	The defUnhideWhenUsed attribute specifies that all latent styles in the current document must have a unhideWhenUsed element setting of `true` by default. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_LatentStyles) is located in §A.1. *end note*]

17.7.4.6 link (Linked Style Reference)

This element specifies the pairing of styles which comprise a linked style. A *linked style* is a grouping of a paragraph style and character style which is used in a user interface to allow the same set of formatting properties to be applied:

- To the contents of one or more entire paragraphs (i.e. as a paragraph style)

- To the contents of one or more runs within a paragraph (i.e. as a character style)

Each style continues to exist independently in the file format as there is both a paragraph and character style present within the styles element (§17.7.4.18), however these two styles shall be merged into one and applied appropriately based on whether they are applied to run(s) or paragraph(s), by referencing the styleId attribute of the paired linked style via this element's val attribute.

A style element without a child link element is not part of a linked style pairing. If no style in the current document specifies the styleId present in the val attribute, then this element shall be ignored.

If a style with this styleId is present, then it shall be subject to the following restrictions:

- If the parent style is a table style, then this element shall be ignored.
- If the parent style is a paragraph style, then this element's val attribute must refer to a character style, or this element shall be ignored.
- If the parent style is a character style, then this element's val attribute must refer to a paragraph style, or this element shall be ignored.
- If the parent style is a numbering style, then this element shall be ignored.

[*Example*: Consider a linked style defined as follows in a WordprocessingML document:

```
<w:style w:type="paragraph" w:styleId="TestParagraphStyle">
  <w:link w:val="TestCharacterStyle"/>
  …
</w:style>
<w:style w:type="character" w:styleId="TestCharacterStyle">
  <w:link w:val="TestParagraphStyle"/>
  …
</w:style>
```

This pairing of a paragraph style and a character style are linked via the link element, which is used to reference the styleId of the paragraph style from the character style definition and vice versa. Because this pairing is permitted based on the rules above, the resulting combination must be used as a linked style, which appears as one style in an application, but uses the character and/or paragraph style as appropriate. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`

Attributes	Description

	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.7.4.7 locked (Style Cannot Be Applied)

This element specifies whether an application shall prevent the use of this style when this document is loaded and/or modified. If this element is set, then this style can be used to format existing content (i.e. any content which references this style shall have its properties as normal), but new instances of the style shall be prevented from being applied via all mechanisms associated with that application.

If this element is omitted, then the use of the style shall not be prevented by an application processing this document.

[*Example*: Consider a style with a primary name of `Test Style` which should be locked, and prevented from being added to any content in a given document. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="TestStyle">
  <w:name w:val="Test Style"/>
  <w:locked/>
  …
</w:style>
```

The presence of the locked element specifies that new instances of the style must be prevented from being applied via all mechanisms associated with that application. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.7.4.8 lsdException (Latent Style Exception)

This element specifies the properties which shall be applied a single latent style for this document. *Latent styles* refer to any set of known style definitions which have not been included in the current document.

[*Example*: Consider a WordprocessingML document which contains text specified in one of two styles: Heading1 or Normal. Based on this, the document only needs to store the formatting properties for those two styles, saving the additional overhead which would be required to save all of the styles supported by the hosting application.

However, if the documentProtection element (§17.15.1.29) specifies that the hosting application must prevent the use of any style whose locked element (§17.7.4.7) is set to false, then the locking state of all styles known to that application become useful and necessary to maintain the current state of the document. Using latent styles, this information can be stored without storing any formatting properties for those styles.

For example, if all styles which are not stored in the document must be locked except for the style with a primary name (§17.7.4.9) of Heading 2. This requirement would be specified using latent styles as follows:

```
<w:latentStyles … w:defLockedState="true">
  <w:lsdException w:name="Heading 2" w:locked="false"/>
</w:latentStyles>
```

The lsdException element specifies that the latent style with a primary name of Heading 2 must have a locked state setting of false. *end example*]

Attributes	Description

locked (Latent Style Locking Setting)	Specifies the default setting for the locked element (§17.7.4.7) which shall be applied to the latent style with the matching style name value.
	If this element is omitted, the default locked state for this latent style shall be determined by the defLockedState attribute on the parent latentStyles element.
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document must be locked except for the TestStyle style. This requirement would be specified using latent styles as follows:
	`<w:latentStyles … w:defLockedState="true">`
	`  <w:lsdException w:name="TestStyle" w:locked="false"/>`
	`</w:latentStyles>`
	The locked attribute on the latent style exception specifies that the TestStyle style must have a locked element setting of false by default. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

name (Primary Style Name)	Specifies the primary name for the style which shall inherit this set of latent style property exceptions.

Attributes	Description

	If the current application does not know of an internal primary style with the current name, then this set of latent style exceptions can be ignored.
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document must be locked except for the `TestStyle` style. This requirement would be specified using latent styles as follows:
	`<w:latentStyles … w:defLockedState="true">`
	  `<w:lsdException w:name="TestStyle" w:locked="false"/>`
	`</w:latentStyles>`
	The name attribute on the latent style exception specifies that the `TestStyle` style must have this set of latent style properties (if the application knows of a style with this name). *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

qFormat (Latent Style Primary Style Setting)	Specifies the default setting for the qFormat element (§17.7.4.14) which shall be applied to the latent style with the matching style name value.
	If this element is omitted, the default qFormat state for this latent style shall be determined by the defQFormat attribute on the parent latentStyles element.
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document must not be primary styles except for the `TestStyle` style. This requirement would be specified using latent styles as follows:
	`<w:latentStyles … w:defQFormat="false">`
	  `<w:lsdException w:name="TestStyle" w:qFormat="true"/>`
	`</w:latentStyles>`
	The qFormat attribute on the latent style exception specifies that the `TestStyle` style must have a qFormat element setting of `true` by default. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

semiHidden (Semi hidden text override)	Specifies the default setting for the semiHidden element (§17.7.4.16) which shall be applied to the latent style with the matching style name value. If this element is omitted, the default semiHidden state for this latent style shall be determined by the defSemiHidden attribute on the parent latentStyles element. [*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document must not be semi-hidden except for the `TestStyle` style. This requirement would be specified using latent styles as follows:

Attributes	Description

	`<w:latentStyles … w:defSemiHidden="false">`
	`  <w:lsdException w:name="TestStyle" w:semiHidden="true"/>`
	`</w:latentStyles>`
	
	The semiHidden attribute on the latent style exception specifies that the `TestStyle` style must have a semiHidden element setting of `true` by default. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

uiPriority (Override default sorting order)	Specifies the default setting for the uiPriority element (§17.7.4.19) which shall be applied to the latent style with the matching style name value.
	
	If this element is omitted, the default uiPriority state for this latent style shall be determined by the defUIPriority attribute on the parent latentStyles element.
	
	[*Example*: Consider a WordprocessingML document in which all styles which are not stored in the document must have a priority value of `10` except for the `TestStyle` style. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defUIPriority="10">`
	`  <w:lsdException w:name="TestStyle" w:uiPriority="25"/>`
	`</w:latentStyles>`
	
	The uiPriority attribute on the latent style exception specifies that the `TestStyle` style must have a uiPriority element setting of `25` by default. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

unhideWhenUsed (Unhide when used)	Specifies the default setting for the unhideWhenUsed element (§17.7.4.20) which shall be applied to the latent style with the matching style name value.
	
	If this element is omitted, the default unhideWhenUsed state for this latent style shall be determined by the defUnhideWhenUsed attribute on the parent latentStyles element.
	
	[*Example*: Consider a WordprocessingML document in which all styles are to be hidden until used except for the `TestStyle` style. This requirement would be specified using latent styles as follows:
	
	`<w:latentStyles … w:defUnhideWhenUsed="true">`
	`  <w:lsdException w:name="TestStyle" w:unhideWhenUsed="false"/>`
	`</w:latentStyles>`
	
	The unhideWhenUsed attribute on the latent style exception specifies that the `TestStyle` style must have an unhideWhenUsed element setting of `false` by default. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_LsdException) is located in §A.1. *end note*]

17.7.4.9 name (Primary Style Name)

This element specifies the primary name for the current style in the document. This name can be used in an application's user interface as desired. The actual primary name for this style is stored in its val attribute.

If present, the alternate style names (§17.7.4.1) shall be used in the user interface in place of the built-in name specified when the appropriate value is set in the stylePaneFormatFilter element (§17.15.1.85).

If this element is omitted, then the style shall not have a primary style name.

[*Example*: Consider a style with a primary name and two alternate names, defined using the name and aliases elements, as follows:

```
<w:style w:styleId="TestStyle" … >
  <w:name w:val="GD20Complex"/>
  <w:aliases w:val="Regional Growth,Complex Growth"/>
  …
</w:style>
```

This style specifies that it has the primary name `GD20Complex` using the name element. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`

Attributes	Description
	`</w:sdtPr>` In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.7.4.10 next (Style For Next Paragraph)

This element specifies the style which shall automatically be applied to a new paragraph created following a paragraph with the parent paragraph style applied. [*Note*: This setting is typically used when the use of the current style is limited to one paragraph at most, and it would typically be undesirable to apply this style to following paragraphs - for example, a title style might specify that its following paragraphs must return to regular text formatting. *end note*]

If this element is specified on a style of any style type other than a paragraph style, this element shall be ignored. If no style whose styleId matches the val attribute of this element exists or that style is not a paragraph style, this element shall be ignored.

If this element is omitted, then the following paragraph shall use the same paragraph style as the current paragraph.

[*Example*: Consider a style defined as follows in a WordprocessingML document:

```
<w:style w:styleId="TestParagraphStyle" … >
  <w:name w:val="Test Paragraph Style"/>
  <w:next w:val="AnotherParagraphStyle"/>
  <w:rPr>
    <w:b/>
  </w:rPr>
  …
</w:style>
```

This style specifies via the use of the next element that the style for the next paragraph in the document must be the paragraph style whose styleId attribute value is `AnotherParagraphStyle` (if such a paragraph style exists). *end example*]

Attributes	Description
val (String Value)	Specifies that its contents contain a string.

Attributes	Description

	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.7.4.11 personal (E-Mail Message Text Style)

This element specifies that the parent style, when in use in the context of an e-mail message, was used by default to format all message text from one or more users. [*Note*: This setting does not provide any additional semantic about the style, but can be used in the context of e-mail to automatically reformat the contents of the e-mail message while ignoring any content to which styles were deliberately applied (since this style was implicitly applied to message text without user interaction). *end note*]

If this element is specified on a style of any style type other than a character style, this element shall be ignored. If no style whose styleId matches the val attribute of this element exists or that style is not a character style, this element shall be ignored.

If this element is omitted, then the current style shall not be considered a message text style in the context of e-mail messages.

[*Example*: Consider a style defined as follows in a WordprocessingML document:

```
<w:style w:styleId="EmailText" w:type="character" >
  <w:name w:val="EmailText"/>
  <w:personal w:val="true" />
  <w:rPr>
    …
  </w:rPr>
</w:style>
```

This style specifies via the use of the personal element that this style is a style used to format message text in the context of e-mail. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.7.4.12 personalCompose (E-Mail Message Composition Style)

This element specifies that the parent style, when in use in the context of an e-mail message, can be used by default to format new message text within the e-mail message. [*Note*: This setting does not provide any additional semantic about the style, but can be used in the context of e-mail to automatically format the contents of new test in the e-mail message. *end note*]

If this element is specified on a style of any style type other than a character style, this element shall be ignored. If no style whose styleId matches the val attribute of this element exists or that style is not a character style, this element shall be ignored.

If this element is omitted, then the current style shall not be considered a message composition text style in the context of e-mail messages.

[*Example*: Consider a style defined as follows in a WordprocessingML document:

```
<w:style w:styleId="EmailText" w:type="character" >
  <w:name w:val="EmailText"/>
  <w:personalCompose w:val="true" />
  <w:rPr>
    …
  </w:rPr>
</w:style>
```

This style specifies via the use of the personalCompose element that this style is a style used to format new message text in the context of e-mail. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.7.4.13 personalReply (E-Mail Message Reply Style)

This element specifies that the parent style, when in use in the context of an e-mail message, can be used by default to format existing message text within the e-mail message when a new reply is generated. [*Note*: This setting does not provide any additional semantic about the style, but can be used in the context of e-mail to automatically format the contents of existing test in the e-mail message. *end note*]

If this element is specified on a style of any style type other than a character style, this element shall be ignored. If no style whose styleId matches the val attribute of this element exists or that style is not a character style, this element shall be ignored.

If this element is omitted, then the current style shall not be considered a message reply text style in the context of e-mail messages.

[*Example*: Consider a style defined as follows in a WordprocessingML document:

```
<w:style w:styleId="EmailText" w:type="character" >
  <w:name w:val="EmailText"/>
  <w:personalReply w:val="true" />
  <w:rPr>
    …
  </w:rPr>
</w:style>
```

This style specifies via the use of the personalReply element that this style is a style used to format existing message text in the context of e-mail. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.7.4.14 qFormat (Primary Style)

This element specifies whether this style shall be treated as a primary style when this document is loaded by an application. If this element is set, then this style has been designated as being particularly important for the current document, and this information can be used by an application in any means desired. [*Note*: This setting does not imply any behavior for the style, only that the style is of particular significance for this document. *end note*]

If this element is omitted, then the style shall not be considered a primary style for this document.

[*Example*: Consider a style with a primary name of PrimaryStyleExample that should be treated as a primary style for the document. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="PStyle">
  <w:name w:val="PrimaryStyleExample"/>
  <w:qFormat/>
  …
</w:style>
```

The qFormat element specifies that this style definition must be treated as a primary style for this document. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.7.4.15 rsid (Revision Identifier for Style Definition)

This element specifies a unique four digit number which shall be used to determine the editing session in which this style definition was last modified. This value shall follow this following constraint: All document elements which specify the same rsid* values shall correspond to changes made during the same editing session. An *editing session* is defined as the period of editing which takes place between any two subsequent save actions. [*Note*: This setting does not imply any behavior for the style, only that the style was last modified during one particular editing session. This information can be interpreted by an application in any manner desired. *end note*]

If this element is omitted, then no revision identifier shall be associated with the parent style definition.

[*Example*: Consider a style with a primary name of `PrimaryStyleExample` that is defined as follows:

```
<w:style … w:styleId="PStyle">
  <w:name w:val="PrimaryStyleExample"/>
  <w:rsid w:val="3E412D01"/>
  …
</w:style>
```

The rsid element specifies that this style definition was last edited in the editing session corresponding to the value `3E412D01`. *end example*]

Attributes	Description
val (Long Hexadecimal Number Value)	Specifies a number value specified as a four digit hexadecimal number), whose contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following value for an attribute of simple type ST_LongHexNumber: `00BE2C6C`. This value is permitted, as it contains four hexadecimal digits, each an encoding of an octet of the actual decimal number value. It can therefore be interpreted as desired in the context of the parent XML element, *end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element's content model (CT_LongHexNumber) is located in §A.1. *end note*]

17.7.4.16 semiHidden (Hide Style From Main User Interface)

This element specifies whether this style shall be hidden from the main user interface when this document is loaded by an application. If this element is set, then this style can be used to format content (i.e. any content which references this style shall have its properties as normal), but the style shall be hidden from the main user interface associated with that application.

[*Note*: The interpretation of a "main" user interface must not be dictated by ISO/IEC 29500, and can be defined by an application as appropriate.

This setting is intended to define a style property which allows styles to be seen and modified in an advanced user interface, without exposing the style in a less advanced setting, for example, the style which is used to format the contents of a comment should typically not be shown in a simple user interface (as it is uncommon to want to modify it), but would be inappropriate to hide completely using the hidden element (§17.7.4.4), as very advanced users might want to change its appearance. *end note*]

If this element is omitted, then the style shall not be required to be hidden from the main user interface.

[*Example*: Consider a style with a primary name of `Comment Style` that should not be displayed in the main user interface. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="CStyle">
  <w:name w:val="Comment Style"/>
  <w:semiHidden/>
  …
</w:style>
```

The semiHidden element specifies that this style definition should not be displayed in any main user interface associated with an application which processes this document. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.7.4.17 style (Style Definition)

This element specifies the definition of a single style within a WordprocessingML document. A *style* is a predefined set of table, numbering, paragraph, and/or character properties which can be applied to regions within a document.

The style definition for any style definition can be divided into three segments:

- General style properties
- Style type
- Style type-specific properties

General style properties refers to the set of properties which can be used regardless of the style type; for example, the style name, additional aliases for the style, a style ID (used by the document content to refer to the style), if style is hidden, if style is locked, etc.

[*Example:* Consider a style called Heading 1 in a document as follows:

```
<w:style w:type="paragraph" w:styleId="Heading1">
  <w:name w:val="Heading 1"/>
  <w:basedOn w:val="Normal"/>
  <w:next w:val="Normal"/>
  <w:link w:val="Heading1Char"/>
  <w:uiPriority w:val="1"/>
  <w:qFormat/>
  <w:rsid w:val="00F303CE"/>
  …
</w:style>
```

Above the formatting information specific to this style type are a set of general style properties which define information shared by all style types. *end example*]

Style types refers to the property on a style which defines the type of style created with this style definition. WordprocessingML supports six types of style definitions by the values for the style definition's type attribute:

- Paragraph styles
- Character styles
- Linked styles (paragraph + character) [*Note*: Accomplished via the link element (§17.7.4.6). *end note*]
- Table styles
- Numbering styles
- Default paragraph + character properties

[*Example:* Consider a style called Heading 1 in a document as follows:

```
<w:style w:type="paragraph" w:styleId="Heading1">
  <w:name w:val="Heading 1"/>
  <w:basedOn w:val="Normal"/>
  <w:next w:val="Normal"/>
  <w:link w:val="Heading1Char"/>
  <w:uiPriority w:val="1"/>
  <w:qFormat/>
  <w:rsid w:val="00F303CE"/>
  …
</w:style>
```

The type attribute has a value of `paragraph`, which indicates that the following style definition is a paragraph style. *end example*]

Style type-specific properties refers to the payload of the style: its formatting information as well as any properties which apply only to that style type.

[*Example*: Consider a table style with primary name `Normal Table` defined as follows:

```
<w:style w:type="table" w:default="1" w:styleId="TableNormal">
  <w:name w:val="Normal Table"/>
  …
  <w:tblPr>
    <w:tblInd w:w="0" w:type="dxa"/>
    <w:tblCellMar>
      <w:top w:w="0" w:type="dxa"/>
      <w:start w:w="108" w:type="dxa"/>
      <w:bottom w:w="0" w:type="dxa"/>
      <w:end w:w="108" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
</w:style>
```

The tblPr element contains the formatting payload for this table style, which is only applicable to a table style. *end example*]

Attributes	Description

customStyle (User-Defined Style)	Specifies that this style is a user-defined style (i.e. it is not a style which was automatically generated by an application). This setting (specifically a value of `true` or its equivalents) shall not allow the formatting associated with the style to be changed automatically by an application, but can be used to specify that if the associated style ID is known, certain user interface behaviors can be applied to its definition. [*Example*: The style's primary name can be localized to match the current user interface language. *end example*]
	If this attribute is omitted, then the style shall be assumed to be a built-in style.
	[*Example*: Consider a paragraph style defined as follows:
	`<w:style w:type="paragraph" w:styleId="MyStyle" w:customStyle="true">`
	`  <w:name w:val="My Paragraph Style"/>`
	`  <w:rPr>`
	`    <w:b/>`
	`  </w:rPr>`
	`</w:style>`
	This paragraph style specifies that it is a user-defined style using the customStyle attribute's value of `true`. An application can therefore take action on the style if it has behaviors associated with the style ID `MyStyle`. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

default (Default Style)	Specifies that this style is the default for this style type. This property is used in conjunction with the type attribute to determine the style which is applied to objects that do not explicitly declare a style. [*Example*: The paragraph style

Attributes	Description

	with the default attribute set is the paragraph style applied to all paragraphs which do not explicitly reference a paragraph style using the pStyle element (§17.3.1.27). *end example*]
	If this attribute is not specified for any style, then no properties shall be applied to objects of the specified style type. If this attribute is specified by multiple styles, then the last instance of a style with this property shall be used.
	[*Example*: Consider a paragraph style defined as follows:
	`<w:style w:type="paragraph" w:default="1" w:styleId="MyStyle" >`
	`  <w:name w:val="My Paragraph Style"/>`
	`  <w:rPr>`
	`    <w:b/>`
	`  </w:rPr>`
	`</w:style>`
	This paragraph style specifies that it is the default paragraph style, and therefore all paragraphs which do not explicitly reference a paragraph style must have this style applied.
	For example, consider the following paragraphs from the same WordprocessingML document:
	`<w:p>`
	`  <w:pPr>`
	`    <w:pStyle w:val="Normal"/>`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	`<w:p>`
	`  …`
	`</w:p>`
	The contents of the first paragraph must have the `Normal` paragraph style applied to them, while the contents of the second paragraph must have the `MyStyle` paragraph style applied, since it does not explicitly reference a paragraph style and therefore inherits the default. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

styleId (Style ID)	Specifies a unique identifier for the parent style definition. This identifier shall be used in multiple contexts to uniquely reference this style definition within the document. [*Example*: The following are examples of elements which reference a style via its styleId attribute value: • To reference a style from content using elements like the pStyle element

Attributes	Description

	(§17.3.1.27), rStyle element (§17.3.2.29), and the tblStyle element (§17.4.62) for paragraphs, runs, and tables, respectively.
	• To link the paragraph and character versions of a style via the link element (§17.7.4.6)
	• To reference the parent style for style inheritance via the basedOn element (§17.7.4.3)
	
	end example]
	
	If multiple style definitions each declare the same value for their styleId, then the first such instance shall keep its current identifier with all other instances being reassigned in any manner desired. This reassignment shall not require references to those style definitions to be 'repaired' in the content (i.e. some content might lose its style definition information, since the document was ill-formed).
	
	If this attribute is not specified, then a style ID can be assigned in any manner desired.
	
	[*Example*: Consider a paragraph style defined as follows:
	
	`<w:style w:type="paragraph" w:styleId="MyStyle" >`
	`  <w:name w:val="My Paragraph Style"/>`
	`  <w:rPr>`
	`    <w:b/>`
	`  </w:rPr>`
	`</w:style>`
	
	This paragraph style specifies that its style identifier must be `MyStyle` using the styleId attribute.
	
	Now consider the following paragraphs from the same WordprocessingML document:
	
	`<w:p>`
	`  <w:pPr>`
	`    <w:pStyle w:val="MyStyle"/>`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	`<w:p>`
	`  …`
	`</w:p>`
	
	The contents of the first paragraph must have the bold paragraph property applied to them because their paragraph properties specify that they must inherit the paragraph style whose styleId is `MyStyle` therefore inheriting its properties using the rules of the style hierarchy. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type

Attributes	Description
	(§22.9.2.13).

type (Style Type)	Specifies the type of style definition defined by this element. WordprocessingML supports six types of style definitions:
	• Paragraph styles
	• Character styles
	• Table styles
	• Numbering styles
	• Linked styles (paragraph + character)
	• Default paragraph + character properties
	
	Each of the first four style types corresponds to a different value in this attribute, and therefore defines the style type of the current style. [*Note*: The last two style types are unique in that they are not simply a style type: a linked style is a pairing of a character and paragraph style via the link element (§17.7.4.6); and the document default properties are defined via the docDefaults element (§17.7.5.1). *end note*]
	
	If this attribute is not specified, then the default value shall be assumed to be paragraph.
	
	[*Example*: Consider a style defined as follows:
	
	`<w:style w:type="paragraph" … >`
	`  <w:name w:val="My Paragraph Style"/>`
	`  <w:rPr>`
	`    <w:b/>`
	`  </w:rPr>`
	`</w:style>`
	
	The type attribute value of paragraph specifies that this style definition creates a paragraph style. *end example*]
	
	The possible values for this attribute are defined by the ST_StyleType simple type (§17.18.83).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Style) is located in §A.1. *end note*]

17.7.4.18 styles (Style Definitions)

This element specifies all of the style information stored in the WordprocessingML document: style definitions as well as latent style information.

[*Example*: The Normal paragraph style in a word processing document can have any number of formatting properties, e.g. font face = Times New Roman; font size = 12pt; paragraph justification = left). All paragraphs which reference this paragraph style would automatically inherit these properties. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Styles) is located in §A.1. *end note*]

17.7.4.19 uiPriority (Optional User Interface Sorting Order)

This element specifies a number which can be used to sort the set of style definitions in a user interface when this document is loaded by an application and the recommended setting is specified in the stylePaneSortMethod element (§17.15.1.86). If this element is set, then this priority shall be used to sort all available styles in ascending value order.

If this element is omitted, then the style shall not have an associated priority value and shall be sorted to the end of the list of style definitions (more or less equivalent to a priority value of infinity) when the recommended sort order setting is specified.

[*Example*: Consider a style with a primary name of `Comment Style` that should have an associated priority value of ten. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="CStyle">
  <w:name w:val="Comment Style"/>
  <w:uiPriority w:val="10"/>
  …
</w:style>
```

The uiPriority element specifies that this style definition should be sorted into the list of styles using a value of `10` when the styles are listed in recommended order using the stylePaneSortMethod element (§17.15.1.86). *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.7.4.20 unhideWhenUsed (Remove Semi-Hidden Property When Style Is Used)

This element specifies whether the semiHidden property (§17.7.4.16) shall be removed when this style is used by the content of the document. If this element is set, then an application shall ensure that even if the semiHidden element is specified on a style, that this property is removed when the document is resaved if the style is referenced by any content in the document.

If this element is omitted, then the style shall not automatically lose the semi-hidden property when it is used in the document contents.

[*Example*: Consider a style with a primary name of `Test Paragraph Style` that should not be displayed in the main user interface until it is used. This requirement would be specified using the following WordprocessingML:

```
<w:style … w:styleId="TestStyle">
  <w:name w:val="Test Paragraph Style"/>
  <w:semiHidden/>
  <w:unhideWhenUsed/>
  …
</w:style>
```

The unhideWhenUsed element specifies that this style definition should not be displayed in any main user interface associated with an application which processes this document until it is referenced by document content. If a paragraph was added to the document which referenced this style:

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="TestStyle"/>
  </w:pPr>
  …
</w:p>
```

This style is now referenced by the document's contents and would have the semiHidden element removed on save. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.7.5 Document Defaults

The first formatting information which is applied to all regions of text in a WordprocessingML document when that document is displayed is the document defaults. The document defaults specify the default set of properties which shall be inherited by every paragraph and run of text within all stories of the current WordprocessingML document. If no other formatting information was referenced by that text, these properties would solely define the formatting of the resulting text.

[*Example*: Consider the following fragment from the main document part of a WordprocessingML document:

```
<w:body>
  <w:p>
    <w:r>
      <w:t>Hello, world</w:t>
    </w:r>
  </w:p>
</w:body>
```

This paragraph and run of text both specify no formatting information (i.e. the paragraph and run possess neither a pPr element nor an rPr element respectively). Therefore, the only formatting applied to this text shall be the formatting in the document defaults which is applied to all paragraphs and runs in the document.

Note that this does not imply that these properties are only applied to text with no formatting - they are rather applied to all text before all other formatting (and in this case, there is no other formatting). *end example*]

17.7.5.1 docDefaults (Document Default Paragraph and Run Properties)

This element specifies the set of default paragraph and run properties which shall be applied to every paragraph and run in the current WordprocessingML document. These properties are applied first in the style hierarchy; therefore they are superseded by any further conflicting formatting, but apply if no further formatting is present.

If this element is omitted, then the document defaults shall be application-defined by the hosting application.

[*Example*: Consider the following definition for the document defaults for a WordprocessingML document:

```
<w:docDefaults>
  <w:rPrDefault>
    <w:rPr>
      <w:b/>
    </w:rPr>
  </w:rPrDefault>
  <w:pPrDefault>
    <w:pPr>
      <w:jc w:val="center"/>
    </w:pPr>
  </w:pPrDefault>
</w:docDefaults>
```

The child elements of docDefaults specify a default paragraph property of centered text and a default run property of bold text. Applying this formatting to the following fragment from the main document part of the same document:

```
<w:body>
  <w:p>
    <w:r>
      <w:t>Hello, world</w:t>
    </w:r>
  </w:p>
</w:body>
```

This paragraph contains no formatting properties, therefore, using the style hierarchy the document default paragraph and run properties are applied as specified within the docDefaults element and the resulting paragraph is centered as specified in the jc element (§17.3.1.13) as well as bold as specified via the b element (§17.3.2.1). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocDefaults) is located in §A.1. *end note*]

17.7.5.2 pPr (Paragraph Properties)

This element specifies the set of paragraph properties which comprise the default paragraph properties for the current WordprocessingML document. [*Rationale*: The reason that a pPr element is present within the pPrDefault element is to allow for easy repurposing of any set of paragraph properties within a WordprocessingML document - since the paragraph properties are always child elements of a single pPr element, that element can simply be relocated in its entirety to the desired new location without additional modifications. *end rationale*]

If this element is omitted, then the default paragraph properties for the current document are non-existent (i.e. there are no default paragraph properties, and the defaults are therefore application-defined).

[*Example*: Consider the following definition for the document defaults for a WordprocessingML document:

```
<w:docDefaults>
  <w:pPrDefault>
    <w:pPr>
      <w:jc w:val="center"/>
    </w:pPr>
  </w:pPrDefault>
  …
</w:docDefaults>
```

The pPr element as a child of the pPrDefault element contains the set of default paragraph properties for this document - in this case, a justification value of `center`. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_PPrGeneral) is located in §A.1. *end note*]

17.7.5.3 pPrDefault (Default Paragraph Properties)

This element specifies the presence of a set of default paragraph properties for the current document. The actual paragraph properties are stored within the pPr child element of the current element.

If this element is omitted, then the default paragraph properties for the current document are non-existent (i.e. there are no default paragraph properties in the document, and the defaults are therefore application-defined).

[*Example*: Consider the following definition for the document defaults for a WordprocessingML document:

```
<w:docDefaults>
  <w:pPrDefault>
    <w:pPr>
      <w:jc w:val="center"/>
    </w:pPr>
  </w:pPrDefault>
  …
</w:docDefaults>
```

The pPrDefault element is a container for the set of default paragraph properties for this document. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_PPrDefault) is located in §A.1. *end note*]

17.7.5.4 rPr (Run Properties)

This element specifies the set of run properties which comprise the default run properties for the current WordprocessingML document. [*Rationale*: The reason that an rPr element is present within the rPrDefault element is to allow for easy repurposing of any set of run properties within a WordprocessingML document - since the run properties are always child elements of a single rPr element, that element can simply be relocated in its entirety to the desired new location without additional modifications. *end rationale*]

If this element is omitted, then the default run properties for the current document are non-existent (i.e. there are no default run properties, and the defaults are therefore application-defined).

[*Example*: Consider the following definition for the document defaults for a WordprocessingML document:

```
<w:docDefaults>
  …
  <w:rPrDefault>
    <w:rPr>
      <w:b/>
    </w:rPr>
  </w:rPrDefault>
</w:docDefaults>
```

The rPr element as a child of the rPrDefault element contains the set of default run properties for this document - in this case, bold text. *end example*]

The W3C XML Schema definition of this element’s content model (CT_RPr) is located in §A.1. Each child element from the above table shall not occur more than once. [*Note*: This restriction is not reflected in the element's content model due to limitations of W3C XML Schema language.*end note*]

17.7.5.5 rPrDefault (Default Run Properties)

This element specifies the presence of a set of default run properties for the current document. The actual run properties are stored within the rPr child element of the current element.

If this element is omitted, then the default run properties for the current document are non-existent (i.e. there are no default run properties in the document, and the defaults are therefore application-defined).

[*Example*: Consider the following definition for the document defaults for a WordprocessingML document:

```
<w:docDefaults>
  …
  <w:rPrDefault>
    <w:rPr>
      <w:b/>
    </w:rPr>
  </w:rPrDefault>
</w:docDefaults>
```

The rPrDefault element is a container for the set of default run properties for this document. *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_RPrDefault) is located in §A.1. *end note*]

17.7.6 Table Styles

Table styles are style definitions which apply to the contents of zero or more tables within a document. This definition can imply that the style can only define table properties (properties which apply to the table and its constituent rows and cells), however a table style can also define paragraph properties (properties which apply to the positioning and appearance of paragraphs) as well as character properties (properties which apply to runs) for all of the paragraphs and runs within the specified table in the document.

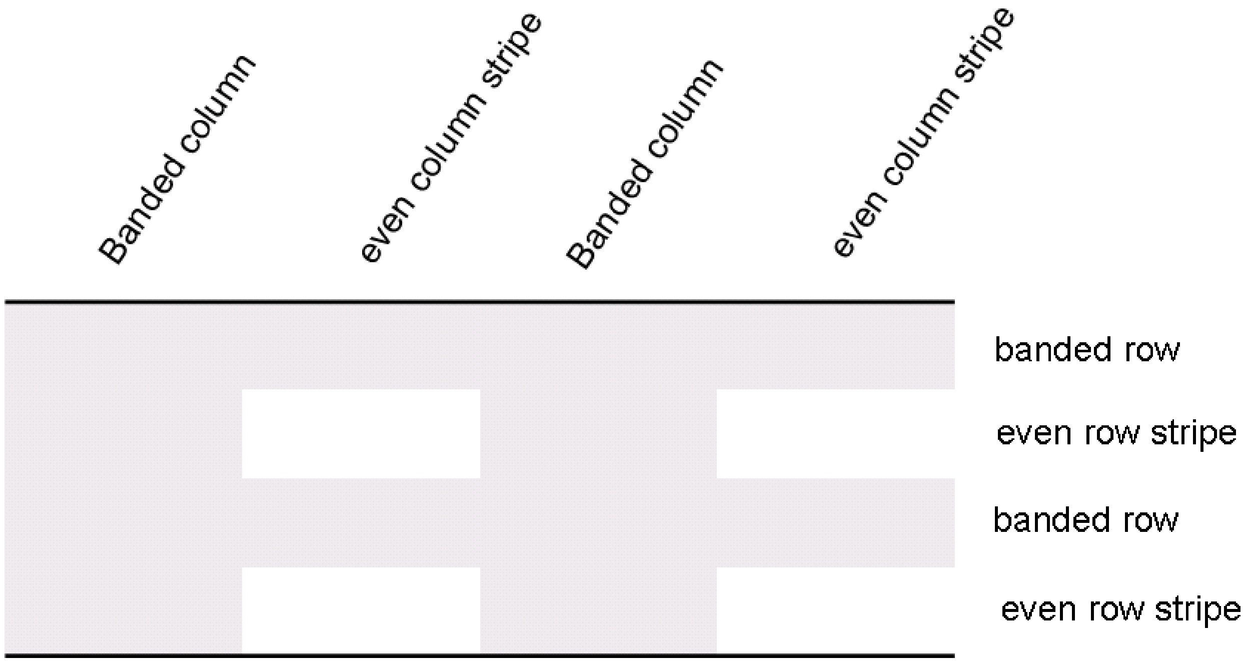
Table styles can only be referenced by tables within a document, and they shall be referenced by the tblStyle element (§17.4.62) within a table’s table properties.

As discussed above, table styles can specify all of the properties that can be applied to a table, as well as paragraph and character properties for the table’s contents. However, unlike other style definitions, table styles allow for the definition of conditional formats for different regions of the table.

These table conditional formats are applied to different regions of the table as follows:

Top left cell	Header row	Top right cell
First column	Table body	Last column
Bottom left cell	Footer row	Bottom right cell

All rows in the table can also have conditional formatting on an alternating row/column basis as well as follows:



When specified, these conditional formats shall be applied in the following order (therefore subsequent formats override properties on previous formats):

- Whole table
- Banded columns, even column banding
- Banded rows, even row banding
- First row, last row
- First column, last column

- Top left, top right, bottom left, bottom right

[*Example*: Consider a table style `Test Table Style` defined as follows:

- All cells with 1pt table borders on all sides
- 0.1" cell margins on left and right of cells
- 0" cell margins on top and bottom of cells

As well as header row specific formatting of

- Red shading
- Bold text

```
<w:style w:type="table" w:styleId="TestTableStyle">
  <w:name w:val="Test Table Style"/>
  <w:basedOn w:val="TableNormal"/>
  <w:uiPriority w:val="99"/>
  <w:rsid w:val="00340CC4"/>
  <w:tblPr>
    <w:tblBorders>
      <w:top w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:start w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:bottom w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:end w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideH w:val="single" w:sz="4" w:space="0" w:color="auto"/>
      <w:insideV w:val="single" w:sz="4" w:space="0" w:color="auto"/>
    </w:tblBorders>
    <w:tblCellMar>
      <w:top w:w="0" w:type="dxa"/>
      <w:start w:w="108" w:type="dxa"/>
      <w:bottom w:w="0" w:type="dxa"/>
      <w:end w:w="108" w:type="dxa"/>
    </w:tblCellMar>
  </w:tblPr>
  <w:tblStylePr w:type="firstRow">
    <w:rPr>
      <w:b/>
    </w:rPr>
    <w:tcPr>
      <w:shd w:val="clear" w:color="auto" w:fill="ED1C24"/>
    </w:tcPr>
  </w:tblStylePr>
</w:style>
```

The tblPr element holds the formatting which is applied to the entire table, and the tblStylePr element with a type attribute value of firstRow holds the formatting for the first table row, specifically the bold run property and red cell shading. *end example*]

An individual instance of a table defines an association with a table style using the tblStyle element in the table's properties (tblPr), as discussed above. However, individual tables can choose whether to apply the following aspects of the table's conditional formats individually:

- First row
- Last row
- First column
- Last column
- Row banding
- Column banding

The use or omission of conditional formats shall be specified using the tblLook element, which contains a a number of attributes that indicate which properties are applied and omitted.

[*Example*: Consider two tables using the table style Style2; one which specifies that it should only use the header row and footer row conditional formatting properties from the table style, and the other which specifies that it should use the header row, footer row, and banded row conditional formatting:

```
<w:tbl>
  <w:tblPr>
    <w:tblStyle w:val="Style2"/>
    <w:tblW w:w="0" w:type="auto"/>
    <w:tblLook w:firstRow="true" w:lastRow="true"
      w:noHBand="true" w:noVBand="true" />
  </w:tblPr>
  …
</w:tbl>
…
<w:tbl>
  <w:tblPr>
    <w:tblStyle w:val="Style2"/>
    <w:tblW w:w="0" w:type="auto"/>
    <w:tblLook w:firstRow="true" w:lastRow="true"
      w:noVBand="true" />
  </w:tblPr>
  …
</w:tbl>
```

The tables each specify the appropriate set of conditional formats using the tblLook element, as seen by the identical table styles in the tblStyle element, and different tblLook values. *end example*]

17.7.6.1 pPr (Table Style Conditional Formatting Paragraph Properties)

This element specifies the set of paragraph properties which shall be applied to all paragraphs within a table which match the conditional formatting type specified on the parent tblStylePr element. These properties are applied in the order specified via the style hierarchy.

[*Example*: Consider a table style which contains conditional formatting for its `firstRow`, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:pPr>
      <w:jc w:val="center"/>
    </w:pPr>
    …
  </w:tblStylePr>
</w:style>
```

The pPr element specified within the tblStylePr element specifies the set of paragraph properties which must be applied to all parts of the table which meet the criteria specified by the type value of `firstRow` - all of the header rows of the table. In this example, the single paragraph property applied is an alignment value of center via the jc element (§17.3.1.13). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_PPrGeneral) is located in §A.1. *end note*]

17.7.6.2 rPr (Table Style Conditional Formatting Run Properties)

This element specifies the set of run properties which shall be applied to all runs within a table which match the conditional formatting type specified on the parent tblStylePr element. These properties are applied in the order specified via the style hierarchy.

[*Example*: Consider a table style which contains conditional formatting for its `firstRow`, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:rPr>
      <w:i/>
    </w:rPr>
    …
  </w:tblStylePr>
</w:style>
```

The rPr element specified within the tblStylePr element specifies the set of run properties which must be applied to all parts of the table which meet the criteria specified by the type value of `firstRow` - all of the

©ISO/IEC 2012 – All rights reserved

header rows of the table. In this example, the single run property applied is italics via the i element (§17.3.2.16). *end example*]

The W3C XML Schema definition of this element's content model (CT_RPr) is located in §A.1. Each child element from the above table shall not occur more than once. [*Note*: This restriction is not reflected in the element's content model due to limitations of W3C XML Schema language.*end note*]

17.7.6.3 tblPr (Table Style Conditional Formatting Table Properties)

This element specifies the set of table properties which shall be applied to all regions within a table which match the conditional formatting type specified on the parent tblStylePr element. These properties are applied in the order specified via the style hierarchy.

If the current conditional formatting type does not consist of one or more full table rows, then table properties which cannot be applied to a single cell or column [*Example*: Table justification. *end example*] can be ignored.

[*Example*: Consider a table style which contains conditional formatting for its `firstRow`, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:tblPr>
      <w:tblCellSpacing w:w="29" w:type="dxa"/>
    </w:tblPr>
    …
  </w:tblStylePr>
</w:style>
```

The tblPr element specified within the tblStylePr element specifies the set of table properties which must be applied to all parts of the table which meet the criteria specified by the type value of `firstRow` - all of the header rows of the table. In this example, the single table property applied is a default table cell spacing value of 0.02 inches via the tblCellSpacing element (§17.4.45). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblPrBase) is located in §A.1. *end note*]

17.7.6.4 tblPr (Style Table Properties)

This element specifies the set of table properties which shall be applied to the table. These properties are not conditional and shall always be applied (although they are applied before all conditional formatting properties).

[*Example*: Consider a table style defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  <w:tblPr>
    <w:tblCellSpacing w:w="15" w:type="dxa"/>
  </w:tblPr>
  …
  <w:tblStylePr w:type="firstRow">-
    <w:tblPr>
      <w:tblCellSpacing w:w="29" w:type="dxa"/>
    </w:tblPr>
    …
  </w:tblStylePr>
</w:style>
```

The tblPr element specified within the style element specifies the set of table properties which must be applied to all parts of the table. In this example, the single table property applied is a default table cell spacing value of 0.01 inches via the tblCellSpacing element (§17.4.45). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblPrBase) is located in §A.1. *end note*]

17.7.6.5 tblStyleColBandSize (Number of Columns in Column Band)

This element specifies the number of columns which shall comprise each a table style column band for this table style. This element determines how many columns constitute each of the column bands for the current table, allowing column band formatting to be applied to groups of columns (rather than just single alternating columns) when the table is formatted.

If this element is omitted, then the default number of columns in a single column band shall be assumed to be 1.

[*Example*: Consider a table style defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  <w:tblPr>
    <w:tblStyleRowBandSize w:val="3" />
    <w:tblStyleColBandSize w:val="2" />
  </w:tblPr>
  …
</w:style>
```

The tblStyleColBandSize element specifies that the width of each column band must be 2 columns - therefore band1Vert column banding conditional formatting must be applied to columns 1 and 2, 5 and 6, etc. in the table. *end example*]

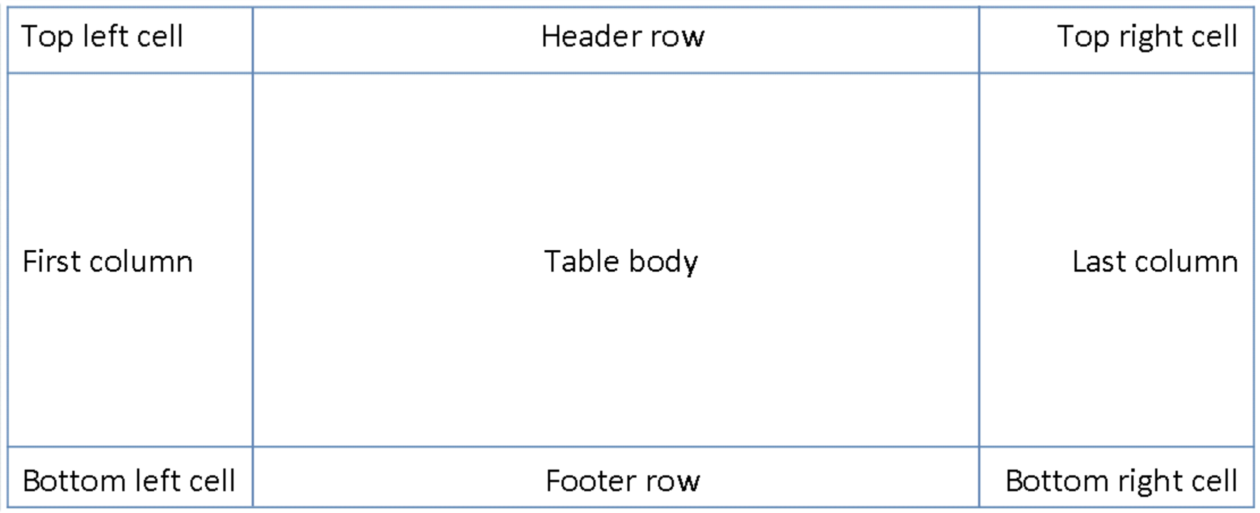
Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type

Attributes	Description
	ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

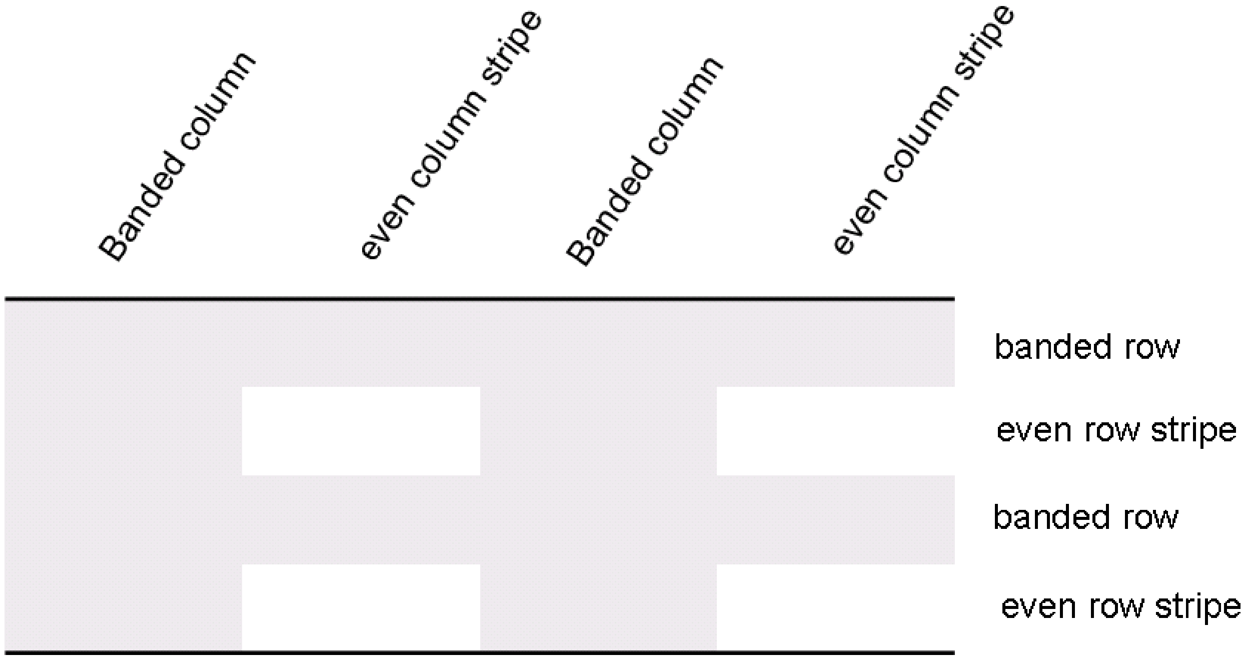
17.7.6.6 tblStylePr (Style Conditional Table Formatting Properties)

This element specifies a set of formatting properties which shall be conditionally applied to the parts of a table which match the requirement specified on the type attribute. These table conditional formats are applied to different regions of the table as follows:



Top left cell	Header row	Top right cell
First column	Table body	Last column
Bottom left cell	Footer row	Bottom right cell

All rows in the table can also have conditional formatting on an alternating row/column basis as well as follows:



When specified, these conditional formats shall be applied in the following order (therefore subsequent formats override properties on previous formats):

- Whole table
- Banded columns, even column banding
- Banded rows, even row banding
- First row, last row
- First column, last column
- Top left, top right, bottom left, bottom right

[*Example*: Consider a table style which contains conditional formatting, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:tblPr>
      <w:tblCellSpacing w:w="29" w:type="dxa"/>
    </w:tblPr>
    …
  </w:tblStylePr>
</w:style>
```

The tblStylePr element specifies a set of table properties which must be conditionally applied to all parts of the table which meet the criteria specified by the type attribute (in this case, all heading rows for the current table). *end example*]

©ISO/IEC 2012 – All rights reserved

Attributes	Description

type (Table Style Conditional Formatting Type)	Specifies the section of the table to which the current conditional formatting properties shall be applied.
	[*Example*: Consider a table style which contains conditional formatting, defined as follows:
	`<w:style w:type="table" …>`
	`  …`
	`  <w:tblStylePr w:type="lastRow">`
	`    …`
	`  </w:tblStylePr>`
	`</w:style>`
	The type attribute value of `lastRow` specifies that this set of conditional formatting properties must be applied to the last row of the table only. *end example*]
	The possible values for this attribute are defined by the ST_TblStyleOverrideType simple type (§17.18.89).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblStylePr) is located in §A.1. *end note*]

17.7.6.7 tblStyleRowBandSize (Number of Rows in Row Band)

This element specifies the number of rows which shall comprise each a table style row band for this table style. This element determines how many rows constitute each of the row bands for the current table, allowing row band formatting to be applied to groups of rows (rather than just single alternating rows) when the table is formatted.

If this element is omitted, then the default number of rows in a single row band shall be assumed to be 1.

[*Example*: Consider a table style defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  <w:tblPr>
    <w:tblStyleRowBandSize w:val="3" />
    <w:tblStyleColBandSize w:val="2" />
  </w:tblPr>
  …
</w:style>
```

The tblStyleRowBandSize element specifies that the width of each row band must be 3 columns - therefore `band1Horiz` row banding conditional formatting must be applied to row 1 through 3, 7 through 9, etc. in the table. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.7.6.8 tcPr (Table Style Conditional Formatting Table Cell Properties)

This element specifies the set of table cell properties which shall be applied to all regions within a table which match the conditional formatting type specified on the parent tblStylePr element. These properties are applied in the order specified via the style hierarchy.

[*Example*: Consider a table style which contains conditional formatting for its `firstRow`, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:tcPr>
      <w:tcBorders>
        <w:top w:val="nil" />
        <w:start w:val="nil" />
        <w:bottom w:val="nil" />
        <w:end w:val="nil" />
        <w:insideH w:val="nil" />
        <w:insideV w:val="nil" />
      </w:tcBorders>
    </w:tcPr>
    …
  </w:tblStylePr>
</w:style>
```

The tcPr element specified within the tblStylePr element specifies the set of table cell properties which must be applied to all parts of the table which meet the criteria specified by the type value of `firstRow` - all of the header rows of the table. In this example, the single table cell property applied is a set of table cell borders via the tcBorders element (§17.4.66). In this case, these cell borders simply reset any previous cell borders to `nil`. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TcPr) is located in §A.1. *end note*]

17.7.6.9 tcPr (Style Table Cell Properties)

This element specifies the set of table cell properties which shall be applied to the table. These properties are not conditional and shall always be applied (although they are applied before all conditional formatting properties).

[*Example*: Consider a table style defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  <w:tcPr>
    <w:tcFitText/>
  </w:tcPr>
</w:style>
```

The tcPr element specified within the style element specifies the set of table cell properties which must be applied to all parts of the table. In this example, the single table cell property applied is the fit text setting via the tcFitText element (§17.4.67). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TcPr) is located in §A.1. *end note*]

17.7.6.10 trPr (Table Style Conditional Formatting Table Row Properties)

This element specifies the set of table row properties which shall be applied to all rows within a table which match the conditional formatting type specified on the parent tblStylePr element. These properties are applied in the order specified via the style hierarchy.

[*Example*: Consider a table style which contains conditional formatting for its `firstRow`, defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  …
  <w:tblStylePr w:type="firstRow">
    <w:trPr>
      <w:tblHeader/>
      <w:cantSplit/>
    </w:trPr>
    …
  </w:tblStylePr>
</w:style>
```

The trPr element specified within the tblStylePr element specifies the set of table row properties which must be applied to all rows of the table which meet the criteria specified by the type value of `firstRow` - all of the header rows of the table. In this example, the table row properties applied are the fact that these rows must be repeated on each page via the tblHeader element (§17.4.49) and the fact that these rows must not be split across pages using the cantSplit element (§17.4.6). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrPr) is located in §A.1. *end note*]

17.7.6.11 trPr (Style Table Row Properties)

This element specifies the set of table row properties which shall be applied to the table. These properties are not conditional and shall always be applied (although they are applied before all conditional formatting properties).

[*Example*: Consider a table style defined as follows:

```
<w:style w:type="table" w:styleId="exampleTableStyle">
  <w:trPr>
    <w:jc w:val="center"/>
  </w:trPr>
</w:style>
```

The trPr element specified within the style element specifies the set of table row properties which must be applied to all parts of the table. In this example, the single table row property applied is the alignment setting of center via the jc element (§17.4.27). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrPr) is located in §A.1. *end note*]

17.7.7 Numbering Styles

Numbering styles are style definitions which specify common style properties for a multi-level numbering format within a document. This means that a numbering style defines only a single paragraph property: a reference to a numbering definition stored in the document's numbering part, using the numPr element.

Unlike paragraph and character styles, numbering styles are never directly referenced by content in the document – instead, an abstract numbering definition (covered in the numbering section) specifies that it is actually the underlying numbering information for a numbering style.

[*Example*: Consider a numbering style "Test Numbering Style":

```
<w:style w:type="numbering" w:styleId="TestNumberingStyle">
  <w:name w:val="Test Numbering Style" />
  <w:uiPriority w:val="99" />
  <w:rsid w:val="0045009F" />
  <w:pPr>
    <w:numPr>
      <w:numId w:val="1" />
      </w:numPr>
    </w:pPr>
  </w:style>
```

The only information specified in the numbering style definition is a reference to the numbering definition for the numbering information which is defined by this numbering style. *end example*]

17.7.8 Paragraph Styles

Paragraph styles are styles which apply to the contents of an entire paragraph as well as the paragraph mark. This definition implies that the style can define both character properties (properties which apply to text within the document) as well as paragraph properties (properties which apply to the positioning and appearance of the paragraph). Paragraph styles cannot be referenced by runs within a document; they shall be referenced by the pStyle element (§17.3.1.27) within a paragraph's paragraph properties element.

A paragraph style has three defining style type-specific characteristics:

- The type attribute on the style has a value of paragraph, which indicates that the following style definition is a paragraph style.
- The next element defines an editing behavior which supplies the paragraph style to be automatically applied to the next paragraph when ENTER is pressed at the end of a paragraph of this style.
- The style specifies both paragraph-level and character-level properties using the pPr and rPr elements, respectively. In this case, the run properties are the set of properties applied to each run in the paragraph.

The paragraph style is then applied to paragraphs by referencing the styleId attribute value for this style in the paragraph properties' pStyle element.

[*Example:* Consider a paragraph style titled "Test Paragraph Style" which defines; font = Algerian, font size = 20; font color = red; paragraph spacing = double; paragraph indent = 1" (first line only). The resulting style definition would be:

```
<w:style w:type="paragraph" w:styleId="TestParagraphStyle">
  <w:name w:val="Test Paragraph Style"/>
  <w:qFormat/>
  <w:rsid w:val="00F85845"/>
  <w:pPr>
    <w:spacing w:line="480" w:lineRule="auto"/>
    <w:ind w:firstLine="1440"/>
  </w:pPr>
  <w:rPr>
    <w:rFonts w:ascii="Algerian" w:hAnsi="Algerian"/>
    <w:color w:val="ED1C24"/>
    <w:sz w:val="40"/>
  </w:rPr>
</w:style>
```

Notice that the character properties for the style are under the rPr element, and the paragraph properties are under the pPr element.

The document content for a paragraph of this style would be:

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="TestParagraphStyle"/>
  </w:pPr>
  <w:r>
    <w:t xml:space="preserve">Here is some fancy Text</w:t>
  </w:r>
</w:p>
```

The pStyle element links the paragraph with the style definition. *end example*]

17.7.8.1 Numbering in Paragraph Styles

When a paragraph style references a numbering definition and level which shall also be applied, that reference shall be done in a way slightly different from the typical numbering reference as follows:

- When a numbering reference is created as direct formatting, that reference consists of a reference to the numbering definition instance + a numbering level

[*Example*: Consider a numbered paragraph in a WordprocessingML document whose numbering is a result of direct formatting (formatting not from a style). This numbered paragraph might be represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level one</w:t>
  </w:r>
</w:p>
```

The numPr element contains two pieces of information:

- The numId element (the numbering definition instance referenced)
- The ilvl element (the level within that numbering definition)

end example]

- When numbering is done as part of a paragraph style, that reference consists of a reference to the numbering definition only. The numbering definition then in turn has a reference to the paragraph style on the level which shall be associated with this style

[*Example*: Consider a numbered paragraph in a WordprocessingML document whose numbering is a result of a paragraph style. This numbered paragraph might be represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:pStyle w:val="TestParagraphStyle"/>
  </w:pPr>
  <w:r>
    <w:t>Level one</w:t>
  </w:r>
</w:p>
```

The paragraph references the style via its styleId attribute, which itself looks like this:

```
<w:style w:styleId="TestParagraphStyle" … >
  <w:pPr>
    <w:numPr>
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
</w:style>
```

The numPr element contains one piece of information:

- The numId element (the numbering definition instance referenced)

Obviously, this is insufficient to apply the numbering since we need to know which level to apply, so this information is specified on the appropriate level using the pStyle element:

```
<w:abstractNum w:abstractNumId="1">
  …
  <w:lvl w:ilvl="0">
    …
    <w:pStyle w:val="TestParagraphStyle" />
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="720" />
      </w:tabs>
      <w:ind w:start="720" w:hanging="360" />
    </w:pPr>
    …
  </w:lvl>
</w:abstractNum>
```

In this case, level 0 of the underlying abstract numbering definition specifies that it is associated with paragraph style TestParagraphStyle, so this level of the numbering must be applied along with the paragraph style. *end example*]

When numbering is referenced by a paragraph style, its properties shall be applied before the style's properties (the style's paragraph properties shall override the numbering level's paragraph properties).

17.7.8.2 pPr (Style Paragraph Properties)

This element specifies the set of paragraph properties which shall be applied to the paragraph.

[*Example*: Consider a paragraph style defined as follows:

```
<w:style w:type="paragraph" w:styleId="TestParaStyle">
  <w:pPr>
    <w:keepLines/>
  </w:pPr>
</w:style>
```

The pPr element specified within the style element specifies the set of paragraph properties which must be applied to the referencing paragraph. In this example, the single paragraph property applied is the fact that the paragraph must be displayed on a single page via the keepLines element (§17.3.1.14). *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_PPrGeneral) is located in §A.1. *end note*]

17.7.9 Run (Character) Styles

Character styles are styles which apply to the contents of one or more runs of text within a document's contents. This definition implies that the style can only define character properties (properties which apply to text within a paragraph) because it cannot be applied to paragraphs. Character styles can only be referenced by runs within a document, and they shall be referenced by the rStyle element within a run's run properties element.

A character style has two defining style type-specific characteristics:

- The type attribute on the style has a value of `character`, which indicates that the following style definition is a character style.
- The style specifies only character-level properties using the rPr element. In this case, the run properties are the set of properties applied to each run which is of this style.

The character style is then applied to runs by referencing the styleId attribute value for this style in the run properties' rStyle element.

[*Example:* Consider a character style titled "Test Character Style" which defines; font = Courier New, font color = yellow; underline. The resulting style definition would be:

```
<w:style w:type="character" w:styleId="TestCharacterStyle">
  <w:name w:val="Test Character Style"/>
  <w:uiPriority w:val="99"/>
  <w:qFormat/>
  <w:rsid w:val="00E77BF0"/>
  <w:rPr>
    <w:rFonts w:ascii="Courier New" w:hAnsi="Courier New"/>
    <w:color w:val="FFF200"/>
    <w:u w:val="single"/>
  </w:rPr>
</w:style>
```

Notice that the character properties applied using this style are under the rPr element. The document content for a paragraph with a run of this style would be:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">The following text is in the </w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="TestCharacterStyle"/>
    </w:rPr>
    <w:t>character style</w:t>
  </w:r>
  <w:r>
    <w:t>.</w:t>
  </w:r>
</w:p>
```

The rStyle element in the second run links that run with the style definition, inheriting the formatting properties for that run. *end example*]

17.7.9.1 rPr (Run Properties)

This element specifies the set of run properties which shall be applied to the run.

[*Example*: Consider a character style defined as follows:

```
<w:style w:type="character" w:styleId="TestCharStyle">
  <w:rPr>
    <w:dstrike/>
  </w:rPr>
</w:style>
```

The rPr element specified within the style element specifies the set of run properties which must be applied to the referencing run. In this example, the single run property applied is the fact that the paragraph must be displayed with double strikethrough via the dstrike element (§17.3.2.9). *end example*]

The W3C XML Schema definition of this element’s content model (CT_RPr) is located in §A.1. Each child element from the above table shall not occur more than once. [*Note*: This restriction is not reflected in the element's content model due to limitations of W3C XML Schema language.*end note*]

17.8 Fonts

The next component of a WordprocessingML document is storing information about the fonts used in the document. WordprocessingML stores two pieces of information about fonts:

- (optionally) Information about the font to enable font substitution. *Font substitution* is a process by which an application, when it cannot locate a specific font, attempts to locate the closest possible match to the intended appearance of the font

- (optionally) One or more embedded forms of the font for use on systems which do not have access to the font. When fonts are embedded, they are obfuscated to ensure that they are only used to view the contents of the document in which they are embedded, and for no other purpose.

[*Example*: Consider the following information stored in a document's font table part:

```
<w:fonts>
  <w:font w:name="Times New Roman">
    <w:panose1 w:val="02020603050405020304" />
    <w:charset w:val="00" />
    <w:family w:val="roman" />
    <w:pitch w:val="variable" />
    <w:sig w:usb0="20002A87" w:usb1="80000000" w:usb2="00000008"
w:usb3="00000000" w:csb0="000001FF" w:csb1="00000000" />
    <w:embedRegular r:id="rId10" w:fontKey="{302EE813-EB4A-4642-A93A-
89EF99B2457E}" />
  </w:font>
</w:fonts>
```

The font table contains information about the `Times New Roman` font; specifically, information used to locate a substitute font when it is not available and a relationship to the embedded form of the regular form of the font. *end example*]

17.8.1 Font Embedding

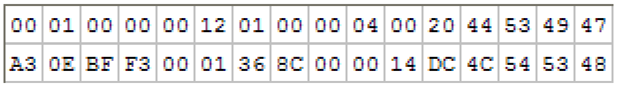
Within a WordprocessingML document, *font embedding* refers to a process in which the some or all of the fonts used in the current document are included in that document such that it can be guaranteed that they are available for use when the document is subsequently opened.

Embedded fonts are stored in an Embedded Font part within the package.

When a font is embedded within a WordprocessingML document, it shall be obfuscated to prevent it from being used outside of this document. This obfuscation shall be done using the following algorithm:

- Generate a GUID, which is used and stored as the obfuscation key
- Reverse the order of the bytes in the GUID (i.e. Big Endian ordering)
- XOR the value with the first 32 bytes of the binary: once against 0-15, once against 16-31
- Store the resulting file in the document, and store the obfuscation key in the fontKey attribute

[*Example*: Consider a font to be embedded whose first 32 bytes are as follows:

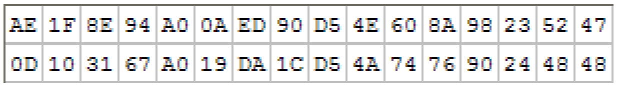


00	01	00	00	00	12	01	00	00	04	00	20	44	53	49	47
A3	0E	BF	F3	00	01	36	8C	00	00	14	DC	4C	54	53	48

To obfuscate this font for storage:

- Generate a GUID (e.g. `001B70DC-AA60-4AD5-90EC-18A0948E1EAE`)
- Reverse its order (e.g. `AE1E8E94-A018-EC90-D54A-60AADC701B00`)
- XOR the GUID with the first and second 16 bytes

The resulting 32 bytes would be:



AE	1F	8E	94	A0	0A	ED	90	D5	4E	60	8A	98	23	52	47
0D	10	31	67	A0	19	DA	1C	D5	4A	74	76	90	24	48	48

end example]

To retrieve an obfuscated font for viewing the content of this document only, repeat the procedure above to retrieve the original font.

17.8.2 Font Substitution

The rFonts element (§17.3.2.26) references the font which is applied to each run of text within a WordprocessingML document. However, based on the availability of these fonts (for example, the use of a custom font), an application might not be able to locate the specified font. The process of finding a suitable alternative font is known as *font substitution*.

The exact algorithm which is used for font substitution is highly dependent on the characteristics which are most desirable when performing the substitution: similar appearance of each glyph (to maximize visual familiarity), similar physical characteristics (to minimize changes in line height and breaking), etc. ISO/IEC 29500 recommends that applications looking for the closest match to the following pieces of information (in descending priority) in order to determine a suitable alternative font; however, applications are free to apply more sophisticated logic in its place:

- sig (§17.8.3.16)
- charset (§17.8.3.2)
- panose1 (§17.8.3.13)
- pitch (§17.8.3.14)
- family (§17.8.3.9)
- altName (§17.8.3.1)
- notTrueType (§17.8.3.12)

17.8.3 Elements

The following elements comprise the content of the font table:

17.8.3.1 altName (Alternate Names for Font)

This element specifies a set of alternative names which can be used to locate the font specified by the parent element. This set of alternative names is stored in a comma-delimited list, with all adjacent commas ignored (i.e. a value of `Name A, Name B` is equivalent to `Name A,,,,,,,,, Name B`).

When an application cannot locate a font using the primary name stored on the font attribute of the font element (§17.8.3.10), it should use each alternate name in term to attempt to locate the font, and use the first font for which is locates a match.

Font names stored using this element shall be specified in the encoding specified by the Fonts part in its XML declaration; the name of the font will be interpreted by the XML parser. [*Note:* UTF-8 is not supported for font names within OpenType fonts, and is not always supported for file names in file systems. For example, older Japanese versions of Microsoft Windows use Windows-31J for filenames. To use extant font names in the values of this attribute, they should be converted to the character encoding appropriate for this standard and copying raw byte sequences from font files should be avoided. *end note*]

If this element is omitted, then no alternate names are present for the parent font.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="SimSun">
  <w:altName w:val="Arial Unicode MS" />
  …
</w:font>
```

The altName element specifies that when no font with a name of `SimSun` (the primary font name) can be located, that applications should attempt to locate a font with the name `Arial Unicode MS` before doing substitution based on the font metrics. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest

Attributes	Description
	ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.8.3.2 charset (Character Set Supported By Font)

This element specifies the character set which is supported by the parent font. This information can be used as defined in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

If this element is not present, then the character set for this font shall be assumed to be the ISO/IEC 8859-1 character set.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="SimSun">
  <w:charset w:characterSet="GBK" />
  …
</w:font>
```

The charset element specifies via its characterSet attribute value of GBK that this font uses the GBK character set. *end example*]

Attributes	Description
characterSet (IANA Name of Character Set)	Name of the character set associated with the font. The values allowed by this attribute are defined by the names and aliases listed in the IANA registration table. The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Charset) is located in §A.1. *end note*]

17.8.3.3 embedBold (Bold Style Font Style Embedding)

This element specifies information about the embedded font storage for the bold form of a font, when it is embedded. This form is used when bold is applied to a text run.

©ISO/IEC 2012 – All rights reserved

If this element is omitted, then no bold form of the font is stored in the document. The relationship targeted by the id attribute shall be of the embedded font type, or the document shall be considered to be non-conformant.

[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:

```
<w:font w:name="Arial">
  …
  <w:embedBold r:id="rId10" />
</w:font>
```

The embedBold element specifies that the embedded font targeted with the relationship with ID rId10 can be used to retrieve the bold form of the embedded Arial font. *end example*]

Attributes	Description

fontKey (Embedded Font Obfuscation Key)	Specifies the key which was used to obfuscate this embedded font. This key can be used to retrieve the embedded font for the purposes of viewing this WordprocessingML document only, using the algorithm described in §17.8.1.
	If this attribute is omitted, then no key is provided for this font.
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:
	`<w:font w:name="Arial">`
	`  …`
	`  <w:embedRegular r:id="rId10" w:fontKey="{302EE813-EB4A-4642-A93A-89EF99B2457E}" />`
	`</w:font>`
	The fontKey attribute has a value of {302EE813-EB4A-4642-A93A-89EF99B2457E}, therefore the embedded Arial font targeted with the relationship with ID rId10 can be retrieved if needed by using this key and the algorithm above. *end example*]
	The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).

id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element

Attributes	Description
	- http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements - http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element - http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

subsetted (Embedded Font Is Subsetted)	Specifies that the embedded font targeted by the id attribute has been subsetted. *Subsetting* is a mechanism by which only the glyphs used in the contents of this WordprocessingML document are stored in an embedded font, in order to prevent the file from becoming unnecessarily large from the use of a small number of glyphs from a large embedded font.
	
	If this attribute is omitted, then the embedded font target by the id attribute shall not be handled as though it is subsetted.
	
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file after subsetting. This status would be specified using the following WordprocessingML:
	
	`<w:font w:name="Arial">`
	`  …`
	`  <w:embedRegular r:id="rId10" w:subsetted="true" />`
	`</w:font>`
	
	The subsetted attribute has a value of `true`, therefore the embedded Arial font targeted with the relationship with ID `rId10` must be treated as a subsetted font. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FontRel) is located in §A.1. *end note*]

©ISO/IEC 2012 – All rights reserved

17.8.3.4 embedBoldItalic (Bold Italic Font Style Embedding)

This element specifies information about the embedded font storage for the bold italic form of a font, when it is embedded. This form is used when bold and italics are applied to a text run.

If this element is omitted, then no bold italic form of the font is stored in the document.

[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:

```
<w:font w:name="Arial">
  …
  <w:embedBoldItalic r:id="rId11" />
</w:font>
```

The embedBoldItalic element specifies that the embedded font targeted with the relationship with ID `rId11` can be used to retrieve the bold italic form of the embedded Arial font. *end example*]

Attributes	Description

fontKey (Embedded Font Obfuscation Key)	Specifies the key which was used to obfuscate this embedded font. This key can be used to retrieve the embedded font for the purposes of viewing this WordprocessingML document only, using the algorithm described in §17.8.1.
	If this attribute is omitted, then no key is provided for this font.
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:
	`<w:font w:name="Arial">`
	`  …`
	`  <w:embedRegular r:id="rId10" w:fontKey="{302EE813-EB4A-4642-A93A-89EF99B2457E}" />`
	`</w:font>`
	The fontKey attribute has a value of `{302EE813-EB4A-4642-A93A-89EF99B2457E}`, therefore the embedded Arial font targeted with the relationship with ID `rId10` can be retrieved if needed by using this key and the algorithm above. *end example*]
	The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).

id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationshi	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/customXml • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element

Attributes	Description
ps	• http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

subsetted
(Embedded Font Is Subsetted)
	Specifies that the embedded font targeted by the id attribute has been subsetted. *Subsetting* is a mechanism by which only the glyphs used in the contents of this WordprocessingML document are stored in an embedded font, in order to prevent the file from becoming unnecessarily large from the use of a small number of glyphs from a large embedded font.
	
	If this attribute is omitted, then the embedded font target by the id attribute shall not be handled as though it is subsetted.
	
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file after subsetting. This status would be specified using the following WordprocessingML:
	
	`<w:font w:name="Arial">`
	`  …`
	`  <w:embedRegular r:id="rId10" w:subsetted="true" />`
	`</w:font>`
	
	The subsetted attribute has a value of `true`, therefore the embedded Arial font targeted with the relationship with ID `rId10` must be treated as a subsetted font. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FontRel) is located in §A.1. *end note*]

17.8.3.5 embedItalic (Italic Font Style Embedding)

This element specifies information about the embedded font storage for the italic form of a font, when it is embedded. This form is used when italics are applied to a text run.

If this element is omitted, then no italic form of the font is stored in the document.

[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:

```
<w:font w:name="Arial">
  …
  <w:embedItalic r:id="rId12" />
</w:font>
```

The embedItalic element specifies that the embedded font targeted with the relationship with ID `rId12` can be used to retrieve the italic form of the embedded Arial font. *end example*]

Attributes	Description

fontKey (Embedded Font Obfuscation Key)	Specifies the key which was used to obfuscate this embedded font. This key can be used to retrieve the embedded font for the purposes of viewing this WordprocessingML document only, using the algorithm described in §17.8.1.
	If this attribute is omitted, then no key is provided for this font.
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:
	`<w:font w:name="Arial">`
	`  …`
	`  <w:embedRegular r:id="rId10" w:fontKey="{302EE813-EB4A-4642-A93A-89EF99B2457E}" />`
	`</w:font>`
	The fontKey attribute has a value of `{302EE813-EB4A-4642-A93A-89EF99B2457E}`, therefore the embedded Arial font targeted with the relationship with ID `rId10` can be retrieved if needed by using this key and the algorithm above. *end example*]
	The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).

id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationshi	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the

Attributes	Description
ps	footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

subsetted (Embedded Font Is Subsetted)	Specifies that the embedded font targeted by the id attribute has been subsetted. *Subsetting* is a mechanism by which only the glyphs used in the contents of this WordprocessingML document are stored in an embedded font, in order to prevent the file from becoming unnecessarily large from the use of a small number of glyphs from a large embedded font.
	
	If this attribute is omitted, then the embedded font target by the id attribute shall not be handled as though it is subsetted.
	
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file after subsetting. This status would be specified using the following WordprocessingML:
	
	`<w:font w:name="Arial">`
	`  …`
	`  <w:embedRegular r:id="rId10" w:subsetted="true" />`
	`</w:font>`
	
	The subsetted attribute has a value of `true`, therefore the embedded Arial font targeted with the relationship with ID `rId10` must be treated as a subsetted font. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FontRel) is located in §A.1. *end note*]

©ISO/IEC 2012 – All rights reserved

17.8.3.6 embedRegular (Regular Font Style Embedding)

This element specifies information about the embedded font storage for the regular form of a font, when it is embedded. This form is used when neither bold nor italics is applied to a text run.

If this element is omitted, then no regular form of the font is stored in the document.

[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:

```
<w:font w:name="Arial">
  …
  <w:embedRegular r:id="rId13" />
</w:font>
```

The embedRegular element specifies that the embedded font targeted with the relationship with ID `rId13` can be used to retrieve the regular form of the embedded Arial font. *end example*]

Attributes	Description

fontKey (Embedded Font Obfuscation Key)	Specifies the key which was used to obfuscate this embedded font. This key can be used to retrieve the embedded font for the purposes of viewing this WordprocessingML document only, using the algorithm described in §17.8.1.
	If this attribute is omitted, then no key is provided for this font.
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file. This status would be specified using the following WordprocessingML:
	`<w:font w:name="Arial">`
	`…`
	`<w:embedRegular r:id="rId10" w:fontKey="{302EE813-EB4A-4642-A93A-89EF99B2457E}" />`
	`</w:font>`
	The fontKey attribute has a value of `{302EE813-EB4A-4642-A93A-89EF99B2457E}`, therefore the embedded Arial font targeted with the relationship with ID `rId10` can be retrieved if needed by using this key and the algorithm above. *end example*]
	The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).

id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationshi	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the

Attributes	Description
ps	footerReference element - http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element - http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements - http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element - http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

subsetted (Embedded Font Is Subsetted)	Specifies that the embedded font targeted by the id attribute has been subsetted. *Subsetting* is a mechanism by which only the glyphs used in the contents of this WordprocessingML document are stored in an embedded font, in order to prevent the file from becoming unnecessarily large from the use of a small number of glyphs from a large embedded font.
	
	If this attribute is omitted, then the embedded font target by the id attribute shall not be handled as though it is subsetted.
	
	[*Example*: Consider a WordprocessingML document in which the Arial font has been embedded in the file after subsetting. This status would be specified using the following WordprocessingML:
	
	`<w:font w:name="Arial">`
	`  …`
	`  <w:embedRegular r:id="rId10" w:subsetted="true" />`
	`</w:font>`
	
	The subsetted attribute has a value of `true`, therefore the embedded Arial font targeted with the relationship with ID `rId10` must be treated as a subsetted font. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FontRel) is located in §A.1. *end note*]

17.8.3.7 embedSystemFonts (Embed Common System Fonts)

This element specifies that applications shall embed common system fonts when they are in use and font embedding is enabled for this document using the embedTrueTypeFonts element (§17.8.3.8). *Common system fonts* refer to a set of fonts which are typically always present on a machine, and are not defined by ISO/IEC 29500.

If this element is omitted, then the set of fonts defined as common system fonts should not be embedded in the current document when font embedded is turned on. If the embedTrueTypeFonts element is omitted or `false`, then this setting has no effect.

[*Example*: Consider a WordprocessingML document that specifies that it must embed fonts, including common system fonts. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:embedTrueTypeFonts w:val="true" />
<w:embedSystemFonts w:val="true"/>
```

The embedSystemFonts element's val attribute has a value of `true` specifying that common system fonts should be included in this document when they are used. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.8.3.8 embedTrueTypeFonts (Embed TrueType Fonts)

This element specifies that applications shall embed the fonts in use in this document when it is saved. These fonts shall be embedded subject to the algorithm specified in §17.8.1.

If this element is omitted, then fonts in use should not be embedded in the current document.

[*Example*: Consider a WordprocessingML document that specifies that it shall embed fonts, including common system fonts. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:embedTrueTypeFonts w:val="true" />
<w:embedSystemFonts w:val="true"/>
```

The embedTrueType element's val attribute has a value of `true` specifying that fonts should be embedded in this document when they are used. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.8.3.9 family (Font Family)

This element specifies the font family of the current font. This information can be used as defined in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

If this element is omitted, then its value shall be assumed to be auto.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Calibri">
  <w:family w:val="swiss" />
  …
</w:font>
```

The family element specifies via its val attribute value of swiss that this font is part of the Swiss family. *end example*]

Attributes	Description

val (Font Family Value)	Specifies the font family for the parent font.
	[*Example*: Consider the following information stored for a single font:
	`<w:font w:name="Times New Roman">`
	`  <w:family w:val="roman" />`
	`  …`
	`</w:font>`
	The val attribute value of swiss that this font is part of the Roman family. *end example*]
	The possible values for this attribute are defined by the ST_FontFamily simple type (§17.18.30).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FontFamily) is located in §A.1. *end note*]

17.8.3.10 font (Properties for a Single Font)

This element specifies the properties for one of the fonts used in this document. A font element shall be written out for each font face used in the document, and includes:

- The name of the font as used in the document's stories
- (optionally) Font metrics allowing other applications to locate appropriate substitute fonts as needed
- (optionally) Embedded forms of the font

[*Example*: Consider the following information stored for a single font:

©ISO/IEC 2012 – All rights reserved

```
<w:font w:name="Times New Roman">
  <w:panose1 w:val="02020603050405020304" />
  <w:charset w:val="00" />
  <w:family w:val="roman" />
  <w:pitch w:val="variable" />
  <w:sig w:usb0="20002A87" w:usb1="80000000" w:usb2="00000008" w:usb3="00000000"
w:csb0="000001FF" w:csb1="00000000" />
</w:font>
```

The font element contains information about the `Times New Roman` font; specifically, information used to locate a substitute font if it is not available. *end example*]

Attributes	Description

name (Primary Font Name)	Specifies the primary name of the current font. This name shall be used to link the information stored in this element with uses of this value in the rFonts element (§17.3.2.26) in document content.
	[*Example*: Consider the following information stored for a single font:
	`<w:font w:name="Times New Roman">`
	`  …`
	`</w:font>`
	The name attribute specifies that the information contained in this element must be used to look up information about all uses of the `Times New Roman` font in the document contents. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Font) is located in §A.1. *end note*]

17.8.3.11 fonts (Font Table Root Element)

This element specifies the root element for a font table part within a WordprocessingML document, and specifies information about the fonts used in this document, each contained within a child font element.

[*Example*: Consider the following information stored in a font table part:

```
<w:fonts>
  <w:font w:name="Times New Roman">
    …
  </w:font>
  <w:font w:name="Arial">
    …
  </w:font>
</w:fonts>
```

The fonts element contains information about all fonts used in the document - in this example, the `Times New Roman` and `Arial` fonts. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_FontsList) is located in §A.1. *end note*]

17.8.3.12 notTrueType (Not a TrueType outline Font)

This element specifies that this font is not a font including TrueType outline in a format conforming to ISO/IEC 14496-22:2007. This information can be used in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

If this element is omitted, then the font shall be assumed to be a font including TrueType outline in a format conforming to ISO/IEC 14496-22:2007.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="JonsFont">
  <w:notTrueType w:val="true" />
  …
</w:font>
```

The notTrueType element specifies via its val attribute value of `true` that this font is not a font including TrueType outline. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.8.3.13 panose1 (Panose-1 Typeface Classification Number)

This element specifies the Panose-1 classification number shown in §5.2.7.17 of ISO/IEC 14496-22. This information can be used as defined in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

If this element is omitted, then no Panose-1 information is available.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Times New Roman">
  <w:panose1 w:val="02020603050405020304" />
  …
</w:font>
```

The panose1 element specifies its Panose-1 number via its val attribute value of 02020603050405020304. *end example*]

Attributes	Description
val (Value)	Specifies the Panose-1 classification number for the font, stored as a series of two digit hexadecimal encodings of each digits of the Panose number. [*Example*: Consider the following information stored for a single font: `<w:panose1 w:val="020F0603050405020304" />` The val attribute specifies that the digits in the Panose-1 number are: 2,15,6,3,5,2,3,4. *end example*] The possible values for this attribute are defined by the ST_Panose simple type (§22.9.2.8).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Panose) is located in §A.1. *end note*]

17.8.3.14 pitch (Font Pitch)

This element specifies the font pitch of the current font. This information can be used as defined in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

If this element is omitted, then its value shall be assumed to be default.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Courier New">
  <w:pitch w:val="fixed" />
  …
</w:font>
```

The pitch element specifies via its val attribute value of fixed that this is a fixed width font. *end example*]

Attributes	Description
val (Value)	Specifies the font pitch for the font. [*Example*: Consider the following information stored for a single font:

Attributes	Description
	`<w:pitch w:val="variable" />` The val attribute value of `variable` specifies that this is a variable width font. *end example*] The possible values for this attribute are defined by the ST_Pitch simple type (§17.18.66).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Pitch) is located in §A.1. *end note*]

17.8.3.15 saveSubsetFonts (Subset Fonts When Embedding)

This element specifies that applications shall subset fonts when font embedding is enabled for this document using the embedTrueTypeFonts element (§17.8.3.8). *Subsetting* is a mechanism by which only the glyphs used in the contents of this WordprocessingML document are stored in an embedded font, in order to prevent the file from becoming unnecessarily large from the use of a small number of glyphs from a large embedded font.

If this element is omitted, then the set of fonts should not be subsetted in the current document when font embedded is turned on. If the embedTrueTypeFonts element is omitted or `false`, then this setting has no effect.

[*Example*: Consider a WordprocessingML document that specifies that it must subset embedded fonts. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:embedTrueTypeFonts w:val="true" />
<w:saveSubsetFonts w:val="true"/>
```

The embedSystemFonts element's val attribute has a value of `true` specifying fonts should be subsetted in this document when they are embedded. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.8.3.16 sig (Supported Unicode Subranges and Code Pages)

This element specifies information identifying the code pages and Unicode subranges for which the parent font provides glyphs using the mechanism defined in §5.2.7.18 and §5.2.7.28 of ISO/IEC 14496-22. This information can be used as defined in font substitution logic to locate an appropriate substitute font when this font is not available. This information is determined by querying the font when present and shall not be modified when the font is not available.

When storing Unicode subrange information, the appropriate bit in the bitfield shall only be set if the entire subrange is supported by that font.

If this element is omitted, then no supported code page/Unicode subrange information is available.

©ISO/IEC 2012 – All rights reserved

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Times New Roman">
  <w:sig w:usb0="20002A87" w:usb1="80000000" w:usb2="00000008" w:usb3="00000000"
w:csb0="000001FF" w:csb1="00000000" />
  …
</w:font>
```

The sig element specifies the supported code pages and Unicode sub ranges via its attributes. For example, the code pages supported are:

- Latin 1
- Latin 2: Eastern Europe
- Cyrillic
- Greek
- Turkish
- Baltic

end example]

Attributes	Description

csb0 (Lower 32 Bits of Code Page Bit Field)	Specifies a four digit hexadecimal encoding of the first 32 bits of the 64-bit code-page bit field that identifies which specific character sets or code pages are supported by the parent font using the format defined by ulCodePageRange1 in §5.2.7.28 of ISO/IEC 14496-22.
	[*Example*: Consider font information specified as follows:
	`<w:font w:name="Lucida Console">`
	`  <w:sig w:csb0="0000001F" … />`
	`  …`
	`</w:font>`
	The csb0 attribute value of `0000001F` specifies that the following code pages are supported by this font:
	• Latin 1
	• Latin 2: Eastern Europe
	• Cyrillic
	• Greek
	• Turkish
	end example]
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

csb1 (Upper 32 Bits	Specifies a four digit hexadecimal encoding of the upper 32 bits of the 64-bit code-page

Attributes	Description

of Code Page Bit Field)	bit field that identifies which specific character sets or code pages are supported by the parent font using the format defined by ulCodePageRange2 in §5.2.7.28 of ISO/IEC 14496-22.
	[*Example*: Consider font information specified as follows:
	`<w:font w:name="Lucida Console">`
	`  <w:sig w:csb1="00000000" … />`
	`  …`
	`</w:font>`
	The csb1 attribute value of `00000000` specifies that none of the specified code pages are supported by this font. *end example*]
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

usb0 (First 32 Bits of Unicode Subset Bitfield)	Specifies the first 32 bits of the 128-bit Unicode subset bit field (USB) as defined by ulUnicodeRange1 of §5.2.7.18 of ISO/IEC 14496-22.
	[*Example*: Consider font information specified as follows:
	`<w:font w:name="Times New Roman">`
	`  <w:sig w:usb0="20002A87" … />`
	`  …`
	`</w:font>`
	The usb0 attribute value of `20002A87` specifies that the first 32 bits of the bitfield are 00100000000000000010101010000111, which corresponds to:
	• Basic Latin
	• Latin-1 Supplement
	• Latin Extended-A
	• Basic Greek
	• Cyrillic
	• Basic Hebrew
	• Basic Arabic
	• Latin Extended Additional
	end example]
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

usb1 (Second 32 Bits of Unicode Subset Bitfield)	Specifies the second 32 bits of the 128-bit Unicode subset bit field (USB) as defined by ulUnicodeRange2 of §5.2.7.18 of ISO/IEC 14496-22. [*Example*: Consider font information specified as follows:

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	`<w:font w:name="Times New Roman">`
	`  <w:sig w:usb1="80000000" … />`
	`  …`
	`</w:font>`
	
	The usb0 attribute value of `80000000` specifies that the first 32 bits of the bitfield are 10000000000000000000000000000000, which corresponds to:
	- Arabic Presentation Forms-A
	
	end example]
	
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

usb2 (Third 32 Bits of Unicode Subset Bitfield)	Specifies the third 32 bits of the 128-bit Unicode subset bit field (USB) as defined by ulUnicodeRange3 of §5.2.7.18 of ISO/IEC 14496-22.
	
	[*Example*: Consider font information specified as follows:
	
	`<w:font w:name="Times New Roman">`
	`  <w:sig w:usb2="00000008" … />`
	`  …`
	`</w:font>`
	
	The usb0 attribute value of `80000000` specifies that the first 32 bits of the bitfield are 00000000000000000000000000001000, which corresponds to:
	- Arabic Presentation Forms-B
	
	end example]
	
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

usb3 (Fourth 32 Bits of Unicode Subset Bitfield)	Specifies the fourth 32 bits of the 128-bit Unicode subset bit field (USB) as defined by ulUnicodeRange4 of §5.2.7.18 of ISO/IEC 14496-22.
	
	[*Example*: Consider font information specified as follows:
	
	`<w:font w:name="Times New Roman">`
	`  <w:sig w:usb3="00000000" … />`
	`  …`
	`</w:font>`
	
	The usb3 attribute value of `00000000` specifies that the first 32 bits of the bitfield are 00000000000000000000000000000000, which corresponds to no subranges. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FontSig) is located in §A.1. *end note*]

17.9 Numbering

Numbering refers to symbols - Arabic numerals, Roman numerals, symbol characters ("bullets"), text strings, etc. - in WordprocessingML that are used to label individual paragraphs of text.

[*Example*: The following two paragraphs each contain numbering as defined by WordprocessingML: the first uses an Arabic numeral, the second a symbol character:

7. This is a paragraph with numbering information.

- This is also a paragraph with numbering information.

end example]

The basis for all numbering in WordprocessingML is specified via two structures:

- abstract numbering definitions
- numbering definition instances

Abstract numbering definitions define the appearance and behavior of a specific set of numbered paragraphs in a document. Because this construct is abstract, they are not be directly referenced by document content, but rather they shall be inherited by a *numbering definition instance*, which itself is referenced by document content.

[*Example*: Consider the following example of an abstract numbering definition in a WordprocessingML document:

```
<w:abstractNum w:abstractNumId="4">
  <w:nsid w:val="FFFFFF7F" />
  <w:multiLevelType w:val="singleLevel" />
  <w:lvl w:ilvl="0">
    <w:start w:val="1" />
    <w:lvlText w:val="%1." />
    <w:lvlJc w:val="start" />
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="720" />
      </w:tabs>
      <w:ind w:start="720" w:hanging="360" />
    </w:pPr>
  </w:lvl>
</w:abstractNum>
```

This abstractNum element defines an abstract numbering definition which defines a set of numbering properties. It is inherited by any numbering definition instance which inherits from an abstractNumId equal to 4:

```
<w:num w:numId="2">
    <w:abstractNumId w:val="4" />
</w:num>
```

This num element defines an numbering definition instance which can define overrides to the abstract numbering definition (in this case it does not), and is used by any paragraphs with a numId equal to 2:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="2" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level one</w:t>
  </w:r>
</w:p>
```

The resulting paragraph inherits the properties of level 0 in the numbering definition instance of 2 which is simply a instance of the abstract numbering definition of 4. *end example*]

17.9.1 abstractNum (Abstract Numbering Definition)

This element specifies a set of properties which shall dictate the appearance and behavior of a set of numbered paragraphs in a WordprocessingML document. These properties are collectively called an *abstract numbering definition*, and are the basis for all numbering information in a WordprocessingML document.

Although an abstract numbering definition contains a complete set of numbering, it shall not be directly referenced by content (hence the use of *abstract*). Instead, these properties shall be inherited by a numbering definition instance using the num element (§17.9.15), which can then itself be referenced by content.

[*Example*: Consider the following example of an abstractNum in a WordprocessingML document:

```
<w:abstractNum w:abstractNumId="4">
  <w:nsid w:val="FFFFFF7F" />
  <w:multiLevelType w:val="singleLevel" />
  <w:lvl w:ilvl="0">
    <w:start w:val="1" />
    <w:lvlText w:val="%1." />
    <w:lvlJc w:val="start" />
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="720" />
      </w:tabs>
      <w:ind w:start="720" w:hanging="360" />
    </w:pPr>
  </w:lvl>
</w:abstractNum>
```

This abstractNum element defines an abstract numbering definition which must be inherited by any numbering definition instance which inherits from abstract numbering definition with an abstractNumId equal to 4. *end example*]

Attributes	Description

abstractNumId (Abstract Numbering Definition ID)	Specifies a unique number which shall be used as the identifier for this abstract numbering definition. This unique number shall be referenced by any numbering definition instance in order to inherit the properties specified by this abstract numbering definition.
	[*Example*: Consider the WordprocessingML for an abstract numbering definition with an abstractNumId attribute of 4:
	`<w:abstractNum w:abstractNumId="4">`
	`  <w:nsid w:val="FFFFFF7F" />`
	`  <w:multiLevelType w:val="singleLevel" />`
	`  <w:lvl w:ilvl="0">`
	`    <w:start w:val="1" />`

Attributes	Description

	`      <w:lvlText w:val="%1." />`
	`      <w:lvlJc w:val="start" />`
	`      <w:pPr>`
	`        <w:tabs>`
	`          <w:tab w:val="num" w:pos="720" />`
	`        </w:tabs>`
	`        <w:ind w:left="720"/>`
	`      </w:pPr>`
	`    </w:lvl>`
	`  </w:abstractNum>`
	
	The abstractNumId attribute serves as a unique identifier for the abstract numbering definition, allowing numbering definition instances (§17.9.15) with a abstractNumId element with a matching attribute value to inherit the abstract numbering definition properties, for example:
	
	`  <w:numbering>`
	`    …`
	`    <w:num w:numId="2">`
	`      <w:abstractNumId w:val="0" />`
	`    </w:num>`
	`    <w:num w:numId="3">`
	`      <w:abstractNumId w:val="1" />`
	`    </w:num>`
	`    <w:num w:numId="4">`
	`      <w:abstractNumId w:val="4" />`
	`    </w:num>`
	`    <w:num w:numId="5">`
	`      <w:abstractNumId w:val="4" />`
	`    </w:num>`
	`  </w:numbering>`
	
	In this case, the final two numbering definition instances both inherit from the abstract numbering definition with a abstractNumId of 4. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_AbstractNum) is located in §A.1. *end note*]

17.9.2 abstractNumId (Abstract Numbering Definition Reference)

This element specifies the abstract numbering definition information whose properties shall be inherited by the parent numbering definition instance.

[*Example*: Consider the WordprocessingML for a document with two numbering definition instances, each referencing a different abstract numbering definition:

```
<w:numbering>
  <w:abstractNum w:abstractNumId="0">
    …
  </w:abstractNum>
  <w:abstractNum w:abstractNumId="1">
    …
  </w:abstractNum>
  …
  <w:num w:numId="1">
    <w:abstractNumId w:val="0" />
  </w:num>
  <w:num w:numId="2">
    <w:abstractNumId w:val="1" />
  </w:num>
  …
</w:numbering>
```

The two numbering definition instances reference the abstract numbering definitions with abstractNumId attribute values of 0 and 1 respectively, via their abstractNumId elements. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

©ISO/IEC 2012 – All rights reserved

17.9.3 ilvl (Numbering Level Reference)

This element specifies the numbering level of the numbering definition instance which shall be applied to the parent paragraph.

This numbering level is specified on either the abstract numbering definition's lvl element (§17.9.6), and can be overridden by a numbering definition instance level override's lvl element (§17.9.5).

[*Example*: Consider the following numbered paragraphs in a WordprocessingML document:

1. Level one

    a. Level two

These numbered paragraphs might be represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level one</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="1" />
      <w:numId w:val="5" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>Level two</w:t>
  </w:r>
</w:p>
```

The WordprocessingML above specifies that the first numbered paragraph references the numbering level of 0, within the numbering definition of the num element (§17.9.15) with a numId attribute equal to 5.

The second numbered paragraph references the numbering of 1, within the same numbering definition instance. The WordprocessingML referenced by the ilvl elements above is given below:

```
<w:num w:numId="5">
  <w:abstractNumId w:val="0" />
</w:num>
…
<w:abstractNum w:abstractNumId="0">
  <w:nsid w:val="FFFFFF7F" />
  <w:multiLevelType w:val="singleLevel" />
  <w:lvl w:ilvl="0">
    …
  </w:lvl>
  <w:lvl w:ilvl="1">
    …
  </w:lvl>
</w:abstractNum>
```

In this case, the resulting paragraphs would inherit the properties of the abstract numbering definition levels with ilvl attributes of `0` and `1`, respectively. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.9.4 isLgl (Display All Levels Using Arabic Numerals)

This element specifies whether or not all levels displayed for a given numbering level's text shall be displayed using the `decimal` number format, regardless of the actual number format of that level in the list. [*Note*: This numbering style is often referred to as the legal numbering style. *end note*]

If this element is present, then all numbering levels present in the lvlTxt element (§17.9.11) shall be converted to their decimal equivalents when they are displayed in this level in the numbering format. If this element is omitted, then each level is displayed using the numFmt (§17.9.17) of that level.

[*Example*: Consider the numbering set below. In this set of blank numbered paragraphs, three numbering levels have been used and the third has the isLgl property applied, resulting in the following:

```
A
   A.a
   A.b
         1.2.1
         1.2.2
B
   B.a
         2.1.1
```

As shown above, each number in the third level in the list has been converted to its decimal equivalent. The WordprocessingML necessary to turn on the legal numbering rule for the third numbering level is given below:

```
<w:lvl w:ilvl="2">
  …
  <w:isLgl />
  …
</w:lvl>
```

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.9.5 lvl (Numbering Level Override Definition)

This element specifies the appearance and behavior of a specific numbering level within a given numbering level definition override defined using the lvlOverride element (§17.9.8).

A numbering level override definition is identical to a numbering level definition, except for the fact that it is defined as part of a numbering definition instance using the num element (§17.9.15) rather than as part of an abstract numbering definition using the abstractNum element (§17.9.1).

[*Example*: Consider a numbering definition instance which inherits its information from the abstract numbering definition with abstractNumId of 4, but should use a different set of properties for level 0 of the numbering definition. The resulting WordprocessingML would look like:

```
<w:num w:numId="6">
  <w:abstractNumId w:val="4" />
  <w:lvlOverride w:ilvl="0">
    <w:lvl w:ilvl="0">
      <w:start w:val="4" />
      <w:lvlText w:val="%1)" />
      <w:lvlJc w:val="start" />
      <w:pPr>
        <w:ind w:start="360" w:hanging="360" />
      </w:pPr>
    </w:lvl>
  </w:lvlOverride>
</w:num>
```

This numbering definition instance overrides level `0` of the list with the specified numbering level override definition, replacing those in the abstract numbering level definition. *end example*]

[*Note*: The ability to set level overrides optimizes use of numbering in WordprocessingML as it prevents writing out redundant abstract numbering definitions if numbering sets only slightly differ.

Consider using WordprocessingML to create two numbered sets that only differ only in the appearance and style of the first numbering level. Both could use the same abstract numbering definition as long as each references a different numbering definition instance with one of the numbering definition instances leveraging a level override for the first numbering level. Below is WordprocessingML that illustrates this:

```
<w:num w:numId="5">
  <w:abstractNumId w:val="4" />
</w:num>
<w:num w:numId="6">
  <w:abstractNumId w:val="4" />
  <w:lvlOverride w:ilvl="0">
    <w:lvl w:ilvl="0">
      <w:start w:val="4" />
      <w:lvlText w:val="%1)" />
      <w:lvlJc w:val="start" />
      <w:pPr>
        <w:ind w:start="360" w:hanging="360" />
      </w:pPr>
    </w:lvl>
  </w:lvlOverride>
</w:num>
```

end note]

Attributes	Description

ilvl (Numbering Level)	Specifies the numbering level definition that is to be defined by this set of numbering properties.
	
	This override is a zero-based index of the number of list levels in the document. [*Example*: A value of `2` is the 3rd list level in the document. *end example*]
	
	[*Example*: Consider the following WordprocessingML for a numbering definition instance:
	
	`<w:num w:numId="6">`
	`  <w:abstractNumId w:val="4" />`
	`  <w:lvlOverride w:ilvl="0">`
	`    …`
	`  </w:lvlOverride>`
	`</w:num>`
	
	In this example, the first numbering level definition (with an ilvl of `0`) within the referenced abstract numbering definition is overridden. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

tentative (Tentative Numbering)	Specifies that a given numbering level was been saved by a producer but was not used in the parent document. This means that this numbering level can be redefined by a future consumer without changing the actual content of the document.
	
	A value of `1` or `true` for this attribute value specifies that the numbering level is not used in the current document's contents.
	
	A value of `0` or `false` for this attribute value specifies that the numbering level is used in the parent document and cannot be redefined without changing its contents. This is the default value for this attribute, and is implied when this attribute is omitted.
	
	[*Example*: Consider the following WordprocessingML numbering level:
	
	`<w:lvl w:ilvl="0" w:tentative="true" >`
	`  …`
	`</w:lvl>`
	
	This level has the tentative attribute set to `true`, therefore the contents of this numbering level have not been used in the document and can be redefined by a consumer as desired. *end example*]
	
	If this attribute is equal to `1` or `true`, the WordprocessingML for a given document contains the numbering level information associated with this numbering level, but the 'tentative' numbering level(s) shall not be represented in any of the hosting application's user interface pertaining to numbering levels.

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

tplc (Template Code)	Specifies a unique hexadecimal value which can be used to specify a location within an application's user interface in which this numbering level shall be displayed. The method by which this value is interpreted shall be application-defined.
	
	If this attribute is omitted, then this numbering can be displayed in any location chosen by the consumer.
	
	[*Example*: Consider the following abstract numbering definition:
	
	`<w:abstractNum w:abstractNumId="1" >`
	`  …`
	`</w:abstractNum>`
	
	In this example the abstractNum element with attribute abstractNumId equal to 1, would appear in the area within a consumer's application user interface specified by the template code `04090019`. *end example*]
	
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Lvl) is located in §A.1. *end note*]

17.9.6 lvl (Numbering Level Definition)

This element specifies the appearance and behavior of a numbering level within a given abstract numbering definition. A numbering level contains a set of properties for the display of the numbering for a given numbering level within an abstract numbering definition.

A numbering level definition is identical to a numbering level override definition, except for the fact that it is defined as part of a numbering definition instance using the abstractNum element (§17.9.1) rather than as part of an abstract numbering definition using the num element (§17.9.15).

[*Example*: Consider the WordprocessingML below:

```
<w:abstractNum w:abstractNumId="4">
  <w:nsid w:val="1DE04504" />
  <w:multiLevelType w:val="hybridMultilevel" />
  <w:lvl w:ilvl="0" w:tplc="0409000F">
    …
  </w:lvl>
  <w:lvl w:ilvl="1" w:tplc="04090019">
    …
  </w:lvl>
  <w:lvl w:ilvl="2" w:tplc="04090019">
    …
  </w:lvl>
  <w:lvl w:ilvl="3" w:tplc="0409000F">
    …
  </w:lvl>
  …
</w:abstractNum>
```

This example shows that any paragraph whose numbering properties use the ilvl elements with the attribute val set equal to 0, 1, 2, or 3 has the appearance and behavior of their first four numbered levels specified by the lvl elements given above (assuming that no level overrides have been specified). *end example*]

Attributes	Description

ilvl (Numbering Level)	Specifies the numbering level definition that is to be defined by this set of numbering properties.
	This override is a zero-based index of the number of list levels in the document. [*Example*: A value of 2 is the 3rd list level in the document. *end example*]
	[*Example*: Consider the following WordprocessingML for a numbering definition instance:
	`<w:num w:numId="6">`
	`  <w:abstractNumId w:val="4" />`
	`  <w:lvlOverride w:ilvl="0">`
	`    …`
	`  </w:lvlOverride>`
	`</w:num>`
	In this example, the first numbering level definition (with an ilvl of 0) within the referenced abstract numbering definition is overridden. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

tentative (Tentative Numbering)	Specifies that a given numbering level was been saved by a producer but was not used in the parent document. This means that this numbering level can be redefined by a future

Attributes	Description

	consumer without changing the actual content of the document.
	A value of `1` or `true` for this attribute value specifies that the numbering level is not used in the current document's contents.
	A value of `0` or `false` for this attribute value specifies that the numbering level is used in the parent document and cannot be redefined without changing its contents. This is the default value for this attribute, and is implied when this attribute is omitted.
	[*Example*: Consider the following WordprocessingML numbering level:
	`<w:lvl w:ilvl="0" w:tentative="true" >`
	`…`
	`</w:lvl>`
	This level has the tentative attribute set to `true`, therefore the contents of this numbering level have not been used in the document and can be redefined by a consumer as desired. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

tplc (Template Code)	Specifies a unique hexadecimal value which can be used to specify a location within an application's user interface in which this numbering level shall be displayed. The method by which this value is interpreted shall be application-defined.
	If this attribute is omitted, then this numbering can be displayed in any location chosen by the consumer.
	[*Example*: Consider the following abstract numbering definition:
	`<w:abstractNum w:abstractNumId="1" >`
	`…`
	`</w:abstractNum>`
	In this example the abstractNum element with attribute abstractNumId equal to `1`, would appear in the area within a consumer's application user interface specified by the template code `04090019.`*end example*]
	The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Lvl) is located in §A.1. *end note*]

17.9.7 lvlJc (Justification)

This element specifies the type of justification used on a numbering level's text within a given numbering level. This justification is applied relative to the text margin of the parent numbered paragraph in the document.

If omitted, the paragraph shall have left justification relative to the text margin in left-to-right paragraphs, and right justification relative to the text margin in right-to-left paragraphs.

[*Example*: Consider the numbering level defined below:

```
<w:lvl w:ilvl="8" w:tplc="756C1446" w:tentative="1">
  <w:start w:val="1" />
  <w:numFmt w:val="bullet" />
  <w:lvlText w:val="•" />
  <w:lvlJc w:val="start" />
  …
</w:lvl>
```

In this numbering level, the given numbering symbol is `left` justified with respect to the text margin, therefore the numbering extends left from the text margin towards the text (assuming a left-to-right paragraph). *end example*]

A numbering level's text is the numeral, symbol, character, graphic, etc. used to create a numbered paragraph as defined by the lvlText element (§17.9.11).

[*Example:* Consider the numbered paragraphs below:

1) Example one

a. Example two

- Example three

The numbering symbol in these three numbered paragraphs are "1", "a", and "•", respectively. *end example*]

Attributes	Description
val (Alignment Type)	Specifies the justification which should be applied to the parent object within a document. The possible values (see below) for this attribute are always specified with `left` specifying justification relative to the leading edge of the paragraph, and therefore change semantic between right-to-left and left-to-right documents. [*Example*: Consider the following WordprocessingML fragment for a paragraph in a document: `<w:pPr>`

Attributes	Description

	`<w:jc w:val="end" />`
	`</w:pPr>`
	This paragraph is now right justified on the page for a left-to-right paragraph, left justfied for a right-to-left paragraph. *end example*]
	The possible values for this attribute are defined by the ST_Jc simple type (§17.18.44).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Jc) is located in §A.1. *end note*]

17.9.8 lvlOverride (Numbering Level Definition Override)

This element specifies an optional override which shall be applied in place of zero or more levels from the abstract numbering definition for a given numbering definition instance. Each instance of this element is used to override the appearance and behavior of a given numbering level definition within the given abstract numbering definition.

[*Example*: Consider a numbering definition instance which inherits its information from the abstract numbering definition with abstractNumId of 4, but wishes to use a different set of properties for level `0` and level `1` of the numbering definition. The resulting WordprocessingML would look like:

```
<w:num w:numId="6">
  <w:abstractNumId w:val="4" />
  <w:lvlOverride w:ilvl="0">
    <w:lvl w:ilvl="0">
      <w:start w:val="4" />
      <w:lvlText w:val="%1)" />
      <w:lvlJc w:val="start" />
      <w:pPr>
        <w:ind w:start="360" w:hanging="360" />
      </w:pPr>
    </w:lvl>
  </w:lvlOverride>
```

```
    <w:lvlOverride w:ilvl="1">
      <w:lvl w:ilvl="1">
        <w:start w:val="5" />
        <w:lvlText w:val="%Test)" />
        <w:lvlJc w:val="start" />
        <w:pPr>
          <w:ind w:start="360" w:hanging="360" />
        </w:pPr>
      </w:lvl>
    </w:lvlOverride>
  </w:num>
```

end example]

[*Note*: The ability to set level overrides optimizes use of numbering in WordprocessingML as it prevents writing out redundant abstract numbering definitions if numbering sets only slightly differ.

Consider using WordprocessingML to create two numbered sets that only differ only in the appearance and style of the first numbering level. Both could use the same abstract numbering definition as long as each references a different numbering definition instance with one of the numbering definition instances leveraging a level override for the first numbering level. Below is WordprocessingML that illustrates this:

```
  <w:num w:numId="5">
    <w:abstractNumId w:val="4" />
  </w:num>
  <w:num w:numId="6">
    <w:abstractNumId w:val="4" />
    <w:lvlOverride w:ilvl="0">
      <w:lvl w:ilvl="0">
        <w:start w:val="4" />
        <w:lvlText w:val="%1)" />
        <w:lvlJc w:val="start" />
        <w:pPr>
          <w:ind w:start="360" w:hanging="360" />
        </w:pPr>
      </w:lvl>
    </w:lvlOverride>
  </w:num>
```

end note]

Attributes	Description
ilvl (Numbering Level ID)	Specifies the numbering level of a given abstract numbering definition to be overridden. If this number conflicts with the ilvl of the child lvl element, then the latter shall be

Attributes	Description

	ignored.
	[*Example*: Consider a numbering definition instance which inherits its information from the abstract numbering definition with abstractNumId of 4, but wishes to use a different set of properties for level 0 of the numbering definition. The resulting WordprocessingML would look like:
	`<w:num w:numId="6">`
	`  <w:abstractNumId w:val="4" />`
	`  <w:lvlOverride w:ilvl="0">`
	`    <w:lvl w:ilvl="0">`
	`      <w:start w:val="4" />`
	`      <w:lvlText w:val="%1)" />`
	`      <w:lvlJc w:val="start" />`
	`      <w:pPr>`
	`        <w:ind w:left="360" />`
	`      </w:pPr>`
	`    </w:lvl>`
	`  </w:lvlOverride>`
	`</w:num>`
	This level overrides level `0` of the abstract numbering definition's level properties with the specified set of numbering properties, replacing those in the abstract numbering definition. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_NumLvl) is located in §A.1. *end note*]

17.9.9 lvlPicBulletId (Picture Numbering Symbol Definition Reference)

This element specifies a picture which shall be used as a numbering symbol for a given numbering level by referring to a picture numbering symbol definition's numPictBullet element (§17.9.20). This reference is made through this element's val attribute.

The picture shall be added to the numbering level by replacing each character in the lvlText with one instance of this image.

[*Example*: Consider the WordprocessingML below illustrating how the lvlPicBulletId references a picture numbering symbol definition though its val attribute:

```
<w:numPicBullet w:numPicBulletId="1">
  <w:drawing>
    …
    </w:drawing>
  </w:numPicBullet>
  …
  <w:abstractNum w:abstractNumId="7">
    <w:nsid w:val="71A06359" />
    <w:multiLevelType w:val="hybridMultilevel" />
    <w:tmpl w:val="10643FE6" />
    <w:lvl w:ilvl="0" w:tplc="B7663E56">
      <w:start w:val="1" />
      <w:numFmt w:val="bullet" />
      <w:lvlText w:val="AA" />
      <w:lvlPicBulletId w:val="1" />
    </w:lvl>
  </w:abstractNum>
```

The resulting numbering must consist of two instances of the image specified using the numPicBullet element. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.9.10 lvlRestart (Restart Numbering Level Symbol)

This element specifies a one-based index which determines when a numbering level should restart to its start value (§17.9.25). A numbering level restarts when an instance of the specified numbering level, which shall be higher (earlier than this level) or any earlier level is used in the given document's contents. [*Example:* If this value is 2, then both level two and level one reset this value. *end example*]

If this element is omitted, the numbering level shall restart each time the previous numbering level or any earlier level is used. If the specified level is higher than the current level, then this element shall be ignored. As well, a value of `0` shall specify that this level shall never restart.

[*Example*: Consider a set of numbered paragraphs in a WordprocessingML document where numbering level with ilvl of `2` is set to never restart:

```
<w:lvl w:ilvl="0">
  <w:start w:val="1" />
  <w:lvlText w:val="%1)" />
  <w:lvlJc w:val="start" />
   <w:pPr>
    <w:ind w:start="360" w:hanging="360" />
  </w:pPr>
   <w:rPr>
    <w:rFonts w:hint="default" />
  </w:rPr>
</w:lvl>
 <w:lvl w:ilvl="1">
  <w:start w:val="1" />
  <w:numFmt w:val="upperLetter" />
  <w:lvlText w:val="%2)" />
  <w:lvlJc w:val="start" />
   <w:pPr>
    <w:ind w:start="720" w:hanging="360" />
  </w:pPr>
   <w:rPr>
    <w:rFonts w:hint="default" />
  </w:rPr>
</w:lvl>
<w:lvl w:ilvl="2">
  <w:start w:val="1" />
  <w:numFmt w:val="lowerRoman" />
  <w:lvlRestart w:val="0">
  <w:lvlText w:val="%3)" />
  <w:lvlJc w:val="start" />
   <w:pPr>
    <w:ind w:start="1080" w:hanging="360" />
  </w:pPr>
   <w:rPr>
    <w:rFonts w:hint="default" />
  </w:rPr>
</w:lvl>
```

Since the lvlRestart element is omitted in numbering level 1 (a,b,…), the numbering level restarts after numbering level 0 (1,2,…) is used. Numbering level two (i, ii, iii …) never restarts as lvlRestart has a val equal to 0. An example of the resulting content would be as follows:

1) Level one

   a) Level two

      i) Level three

      ii) Level three

2) Level one

   a) Level two

      iii) Level three

      iv) Level four

The resulting set of paragraphs has level two restarting to its start value after each level one (after 2), the next level two is again a)), but level three never restarts and continues at iii) even after the use of a level two and one. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.9.11 lvlText (Numbering Level Text)

This element specifies the textual content which shall be displayed when displaying a paragraph with the given numbering level.

All text in this element's val attribute shall be taken as literal text to be repeated in each instance of this numbering level, except for any use of the percent symbol (%) followed by a number, which shall be used to indicate the one-based index of the number to be used at this level. Any number of a level higher than this level shall be ignored.

When the % syntax is used, the number shall be incremented for each subsequent paragraph of that level (sequential or not), until the restart level is seen between two subsequent paragraphs of this level.

[*Example*: Consider the following WordprocessingML for a numbering level:

```
<w:lvl w:ilvl="1">
  …
  <w:lvlText w:val="StringA %2 StringB %1 StringC %3"/>
  …
</w:lvl>
```

This specifies that three strings (StringA, StringB, StringC) must be used as string literals in the numbering for level two (ilvl of 1) along with the numbering symbol used for level one and level zero. Although level two is also referenced here, it is ignored as it is a higher level than the current numbering level.

Therefore, assuming the numbering symbol used by numbering level zero is an Arabic numeral, and the numbering symbol used by numbering level one is a Roman numeral, a set of numbered paragraphs using this WordprocessingML numbering set must be output as:

```
1
  StringA I StringB 1 StringC
  StringA II StringB 1 StringC
  StringA III StringB 1 StringC
2
  StringA I StringB 2 StringC
  StringA II StringB 2 StringC
```

with the %1 and %2 values corresponding to the current numbering symbol value for numbering level zero and one, respectively. *end example*]

Attributes	Description
null (Level Text Is Null Character)	Specifies that a null character shall be used as the numbering symbol for a given numbering level. If the val attribute contains any content, then this attribute shall be ignored.

Attributes	Description

	If this attribute is omitted, then the null string shall not be used in place of the empty string. [*Note*: A null character is different from an empty string. *end note*]
	[*Example*: Consider the WordprocessingML below:
	`<w:lvl w:ilvl="1">`
	`  …`
	`  <w:lvlText w:null="on" />`
	`  …`
	`</w:lvl>`
	This level text consists of a single null character, and not the empty string, as the null attribute is set. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

val (Level Text)	Specifies the actual text to be used for the numbering level when it is referenced in the document's content.
	If this attribute is not specified, then the empty string shall be used as the level's text.
	[*Example*: Consider the WordprocessingML below:
	`<w:lvl w:ilvl="1">`
	`  …`
	`  <w:lvlText w:val="test" />`
	`  …`
	`</w:lvl>`
	Here the val attribute specifies that the literal string `test` is to be surfaced as the text for the given numbering level, regardless of its position. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_LevelText) is located in §A.1. *end note*]

17.9.12 multiLevelType (Abstract Numbering Definition Type)

This element specifies the type of numbering defined by a given abstract numbering type. This information shall only be used by a consumer to determine user interface behaviors for this numbering definition, and shall not be used to limit the behavior of the list (i.e. a list with multiple levels marked as `singleLevel` shall not be prevented from using levels 2 through 9).

If this element is omitted, then the list shall be assumed to be of any numbering type desired by the consumer.

[*Example*: Consider the WordprocessingML below:

```
<w:abstractNum w:abstractNumId="8">
  …
  <w:multiLevelType w:val="singleLevel" />
  …
</w:abstractNum>
```

This abstract numbering definition is specified to be of the `singleLevel` numbering type by the multiLevelType element. *end example*]

Attributes	Description

val (Abstract Numbering Definition Type)	Specifies the specific type of numbering enabled by a given abstract numbering definition.
	[*Example*: Consider the WordprocessingML below:
	`<w:abstractNum w:abstractNumId="8">`
	`  …`
	`  <w:multiLevelType w:val="multilevel" />`
	`  …`
	`</w:abstractNum>`
	This abstract numbering definition is specified to be of the `multilevel` numbering type, which can be used by consumers to place this numbering correctly within a user interface. *end example*]
	The possible values for this attribute are defined by the ST_MultiLevelType simple type (§17.18.58).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MultiLevelType) is located in §A.1. *end note*]

17.9.13 name (Abstract Numbering Definition Name)

This element specifies the name of a given abstract numbering definition. This name can be surfaced in order to provide a user friendly alias for a given numbering definition, but shall not influence the behavior of the list - two identical definitions with different name elements shall behave identically.

If this element is omitted, then this abstract numbering definition shall have no name.

[*Example*: Consider the WordprocessingML below:

```
<w:abstractNum w:abstractNumId="4">
  <w:nsid w:val="5C294B5B" />
  <w:multiLevelType w:val="multilevel" />
```

```
    <w:tmpl w:val="6F8A81B0" />
    <w:name w:val="Example Name" />
    …
  </w:abstractNum>
```

In this example, the given abstract numbering definition is named `Example Name` by use of the name element. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_String) is located in §A.1. *end note*]

17.9.14 nsid (Abstract Numbering Definition Identifier)

This element associates a unique hexadecimal ID to the parent abstract numbering definition. This number shall be identical for two abstract numbering definitions that are based from the same initial numbering definition - if a document is repurposed and the underlying numbering definition is changed, it shall maintain its original nsid.

If this element is omitted, then the list shall have no nsid and one can be added by a producer arbitrarily.

©ISO/IEC 2012 – All rights reserved

[*Note*: This element can be used to determine the abstract numbering definition to be applied to a numbered paragraph copied from one document and pasted into another. Consider a case in which a given numbered paragraph associated with a abstract numbering definition with nsid FFFFFF23, is pasted among numbered paragraphs associated with a completely different appearance and an abstract numbering definition with an nsid of FFFFFF23. Here, because of the distinction enabled by the identical nsid values, the hosting application would not have to arbitrarily keep the pasted numbered paragraph associated with its original abstract numbering definition, as it might use the information provided by the abstract numbering definition's identical nsid values to know that those two numbering sets are identical, and merge the paragraphs into the target numbering format. *end note*]

[*Example*: Consider the WordprocessingML for an abstract numbering definition below:

```
<w:abstractNum w:abstractNumId="3">
  <w:nsid w:val="FFFFFF89" />
  <w:multiLevelType w:val="singleLevel" />
  <w:tmpl w:val="D9842532" />
  …
</w:abstractNum>
```

In this example, the given abstract numbering definition is associated with the unique hexadecimal ID FFFFFF89. *end example*]

Attributes	Description
val (Long Hexadecimal Number Value)	Specifies a number value specified as a four digit hexadecimal number), whose contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following value for an attribute of simple type ST_LongHexNumber: 00BE2C6C. This value is permitted, as it contains four hexadecimal digits, each an encoding of an octet of the actual decimal number value. It can therefore be interpreted as desired in the context of the parent XML element, *end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element's content model (CT_LongHexNumber) is located in §A.1. *end note*]

17.9.15 num (Numbering Definition Instance)

This element specifies a unique instance of numbering information that can be referenced by zero or more paragraphs within the parent WordprocessingML document.

This instance requires the referencing of a base abstract numbering definition through the abstractNumId child element (§17.9.2). This element also can be used to specify a set of optional overrides applied to zero or more levels from the abstract numbering definition inherited by this instance second though the optional lvlOverride child elements (§17.9.8).

[*Example*: Consider the WordprocessingML for a document with four numbering definition instances, two of which reference the same underlying abstract numbering definition:

```
<w:numbering>
  …
  <w:num w:numId="2">
    <w:abstractNumId w:val="0" />
  </w:num>
  <w:num w:numId="3">
    <w:abstractNumId w:val="1" />
  </w:num>
  <w:num w:numId="4">
    <w:abstractNumId w:val="4" />
  </w:num>
  <w:num w:numId="5">
    <w:abstractNumId w:val="4" />
  </w:num>
</w:numbering>
```

As shown above, the first two numbering definition instances reference abstractNumId values of 0 and 1 respectively, and the last two both reference the abstract numbering definition with an abstractNumId of 4. *end example*]

[*Example*: Consider a numbering definition instance which inherits its information from the abstract numbering definition with abstractNumId of 4, but wishes to use a different set of properties for level 0 of the numbering definition. The resulting WordprocessingML would look like:

```
<w:num w:numId="6">
  <w:abstractNumId w:val="4" />
  <w:lvlOverride w:ilvl="0">
    <w:lvl w:ilvl="0">
      <w:start w:val="4" />
      <w:lvlText w:val="%1)" />
      <w:lvlJc w:val="start" />
      <w:pPr>
        <w:ind w:start="360" w:hanging="360" />
      </w:pPr>
    </w:lvl>
  </w:lvlOverride>
</w:num>
```

The lvlOverride element specifies an override for level 0 of the abstract numbering definition. *end example*]

Attributes	Description

numId (Numbering Definition Instance ID)	Specifies a unique ID which any numbered paragraph which wishes to inherit these numbering properties shall reference using the numPr element (§17.3.1.19).
	[*Example*: Consider the WordprocessingML below for an example numbered paragraph:
	`<w:p>`
	`  <w:pPr>`
	`    <w:numPr>`
	`      <w:ilvl w:val="0" />`
	`      <w:numId w:val="5" />`
	`    </w:numPr>`
	`  </w:pPr>`
	`  …`
	`</w:p>`
	This paragraph references a numbering definition instance with a numId attribute of 5:
	`<w:num w:numId="5">`
	`  <w:abstractNumId w:val="4" />`
	`</w:num>`
	The numbering definition instance with a numId attribute of 5 correlates with the numbered paragraph with the numbering definition instance referent element with a val of 5, so the numbered paragraph inherits its properties. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Num) is located in §A.1. *end note*]

©ISO/IEC 2012 – All rights reserved

17.9.16 numbering (Numbering Definitions)

This element specifies the formatting, display, and functionality of numbering - Arabic numerals, Roman numerals, symbol characters ("bullets"), text strings, etc. - in WordprocessingML documents, which are used to label individual paragraphs of text.

[*Example*: The following two paragraphs each contain numbering as defined by WordprocessingML: the first uses an Arabic numeral, the second a symbol character:

8. This is a paragraph with numbering information.

- This is also a paragraph with numbering information.

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Numbering) is located in §A.1. *end note*]

17.9.17 numFmt (Numbering Format)

This element specifies the number format that shall be used to display all numbering at this level in the numbering definition. This information is used to replace the level text string %x, where x is a particular one-based level index, with the appropriate value unless the numFmt value is bullet, in which case the literal text of the level text string is used. This value shall be calculated by counting the number of paragraphs at this level since the last restart using the numbering system defined in the val attribute.

When a document has a custom number format specified by the format attribute, it shall use the referenced number format. If the referenced number format cannot be resolved as a number format the consumer shall use the number format specified by the value of the val attribute. If the corresponding value of the val attribute is custom, the result is implementation-defined.

If this element is omitted, the level shall be assumed to be of level type decimal.

[*Example*: Consider the following WordprocessingML fragment for a numbering level in a numbering definition:

```
<w:lvl w:ilvl="2">
  <w:start w:val="1" />
  <w:numFmt w:val="lowerRoman" />
  <w:lvlRestart w:val="0" />
  <w:lvlText w:val="%3)" />
  <w:lvlJc w:val="start" />
  <w:pPr>
    <w:ind w:start="1080" w:hanging="360" />
  </w:pPr>
  <w:rPr>
    <w:rFonts w:hint="default" />
  </w:rPr>
  </w:lvl>
```

A numFmt value of `lowerLetter` indicates that a consumer must use lowercase letters for all numbering of this level: a,b,c… *end example*]

Attributes	Description
format (Custom Defined Number Format)	Specifies a custom number format using the syntax defined by the XSLT format attribute. This format shall be used for all numbering in the parent object. [*Example*: A value of `&#x30A2;` indicates that a consumer must use Katakana numbering. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
val (Numbering Format Type)	Specifies the number format that shall be used for all numbering in the parent object. [*Example*: A value of `lowerLetter` indicates that a consumer must use lowercase letters for each number in this grouping: a,b,c… *end example*] The possible values for this attribute are defined by the ST_NumberFormat simple type (§17.18.59).

[*Note*: The W3C XML Schema definition of this element's content model (CT_NumFmt) is located in §A.1. *end note*]

17.9.18 numId (Numbering Definition Instance Reference)

This element specifies the numbering definition instance which shall be used for the given parent numbered paragraph in the WordprocessingML document.

A value of `0` for the val attribute shall never be used to point to a numbering definition instance, and shall instead only be used to designate the removal of numbering properties at a particular level in the style hierarchy (typically via direct formatting).

[*Example*: Consider the WordprocessingML below for an example numbered paragraph:

```
  <w:p>
    <w:pPr>
      <w:numPr>
        <w:ilvl w:val="0" />
        <w:numId w:val="5" />
      </w:numPr>
    </w:pPr>
    …
  </w:p>
```

©ISO/IEC 2012 – All rights reserved

This paragraph references a numbering definition instance with a numId attribute of 5, as follows:

```
<w:num w:numId="5">
  <w:abstractNumId w:val="4" />
</w:num>
```

The numbering definition instance reference specifies the given numbering definition instance to be applied to the given paragraph, which itself inherits its properties from abstract numbering definition with abstractNumId of 4. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.9.19 numIdMacAtCleanup (Last Reviewed Abstract Numbering Definition)

This element specifies to a consumer the progress in the last attempt made by the application to remove unused abstract numbering definitions from a given document. If a legacy document is opened by a consumer, it can choose to remove abstract numbering definition which are 'orphaned' (have no associated numbering definition instances). This element is used by those consumers to indicate their progress (if not complete) in reviewing existing abstract numbering definitions. [*Note*: Removing unused abstract numbering definition from a document reduces the file size, but is not required. *end note*]

If omitted, then all abstract numbering definitions shall be considered reviewed.

[*Example*: Consider a document with 32 abstract numbering definitions, with abstractNumId values ranging from 0 to 85. If an application has only reviewed those abstract numbering definitions with abstractNumId values lower than 25 at save time, it would indicate that state as follows:

```
<w:numIdMacAtCleanup w:val="25"/>
```

This value specifies that all abstract numbering definitions with an abstractNumId value higher than 25 have not yet been reviewed. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.9.20 numPicBullet (Picture Numbering Symbol Definition)

This element specifies the appearance and behavior of a specific picture to be used as the numbering symbol within a numbering level definition in a document, and is the basis for all picture numbering symbol information in a WordprocessingML document.

This element is not used directly within abstract numbering definitions but rather is referenced through its numPicBulletId attribute by the lvlPicBulletId element (§17.9.9) used within numbering level definitions.

[*Example*: Consider the WordprocessingML fragment below which illustrates how a numPicBullet definition is referenced by a picture numbering symbol definition reference through its numPicBulletId attribute:

```
<w:numPicBullet w:numPicBulletId="1">
  <w:drawing>
    …
  </w:drawing>
</w:numPicBullet>
…
<w:abstractNum w:abstractNumId="7">
  <w:nsid w:val="71A06359" />
  <w:multiLevelType w:val="hybridMultilevel" />
  <w:tmpl w:val="10643FE6" />
```

```
    <w:lvl w:ilvl="0" w:tplc="B7663E56">
      <w:start w:val="1" />
      <w:numFmt w:val="bullet" />
      <w:lvlText w:val="" />
      <w:lvlPicBulletId w:val="1" />
    </w:lvl>
  </w:abstractNum>
```

The lvlPicBulletId element references a numPicBullet element, which defines the size and appearance of all picture bullets of this picture bullet type within the document. It is important to note that this picture bullet can be referenced by multiple levels of various numbering definitions. *end example*]

Attributes	Description

numPicBulletId (Picture Numbering Symbol ID)	Specifies a unique ID for this picture bullet definition which shall be used to reference this picture bullet from a numbering level definition.
	[*Example*: Consider the WordprocessingML fragment below which illustrates how a numPicBullet definition is referenced by a picture numbering symbol definition reference through its numPicBulletId attribute:
	`<w:numPicBullet w:numPicBulletId="1">`
	`  …`
	`</w:numPicBullet>`
	`…`
	`<w:abstractNum w:abstractNumId="7">`
	`  <w:lvl w:ilvl="0" w:tplc="B7663E56">`
	`    …`
	`    <w:lvlPicBulletId w:val="1" />`
	`  </w:lvl>`
	`</w:abstractNum>`
	The lvlPicBulletId element references the ID in the numPicBulletId attribute directly. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_NumPicBullet) is located in §A.1. *end note*]

17.9.21 numStyleLink (Numbering Style Reference)

This element specifies an abstract numbering that does not contain the actual numbering properties for its numbering type, but rather serves as a reference to a numbering style stored in the document, which shall be applied when this abstract numbering definition is referenced, and itself points at the actual underlying abstract numbering definition to be used.

The numbering style that is to be applied when this abstract numbering definition is referenced is identified by the string contained in numStyleLink's val attribute.

[*Example*: Consider the abstract numbering definition below:

```
<w:abstractNum w:abstractNumId="0">
  <w:nsid w:val="38901FA4" />
  <w:multiLevelType w:val="multilevel" />
  <w:numStyleLink w:val="TestNumberingStyle" />
</w:abstractNum>
```

This abstract numbering definition references the numbering style with a styleId attribute equal to `TestNumberingStyle`, as follows below:

```
<w:style w:type="numbering" w:styleId="TestNumberingStyle">
  …
</w:style>
```

Therefore, this numbering style must be applied whenever the base abstract numbering definition is inherited by a numbered paragraph. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type

Attributes	Description
	(§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.9.22 pPr (Numbering Level Associated Paragraph Properties)

This element specifies the paragraph properties which shall be applied as part of a given numbering level within the parent numbering definition. These paragraph properties are applied to any numbered paragraph that references the given numbering definition and numbering level.

Paragraph properties specified on the numbered paragraph itself override the paragraph properties specified by pPr elements within a numbering lvl element (§17.9.5, §17.9.6).

[*Example:* Consider the WordprocessingML below which specifies numbering level paragraph properties:

```
<w:abstractNum w:abstractNumId="1">
  …
  <w:lvl w:ilvl="0">
    …
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="720" />
      </w:tabs>
      <w:ind w:start="720" w:hanging="360" />
    </w:pPr>
  </w:lvl>
</w:abstractNum>
```

Each of the paragraph properties specified inside the pPr element are applied to any numbered paragraph which inherits this numbering level definition as part of the numbering properties in the order defined by the style hierarchy. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_PPrGeneral) is located in §A.1. *end note*]

17.9.23 pStyle (Paragraph Style's Associated Numbering Level)

This element specifies the name of a paragraph style which shall automatically this numbering level when applied to the contents of the document. When a paragraph style is defined to include a numbering definition, any numbering level defined by the numPr element (§17.3.1.19) shall be ignored, and instead this element shall specify the numbering level associated with that paragraph style.

If this element references a style which does not exist, or is not a paragraph style, then it can be ignored.

[*Example*: Consider the WordprocessingML below which specifies that the paragraph style with styleId example, when applied to paragraphs in the document, must also apply the first numbering level of the abstract numbering definition with an abstractNumId equal to 1, as follows:

```
<w:abstractNum w:abstractNumId="1">
  …
  <w:lvl w:ilvl="0">
    …
    <w:pStyle w:val="example" />
    <w:pPr>
      <w:tabs>
        <w:tab w:val="num" w:pos="720" />
      </w:tabs>
      <w:ind w:start="720" w:hanging="360" />
    </w:pPr>
    …
  </w:lvl>
</w:abstractNum>
```

The style definition for the paragraph style would only include the numId of the numbering definition instance, and not its level:

```
<w:style w:styleId="example" w:type="paragraph">
  …
  <w:pPr>
    <w:numPr>
      <w:numId w:val="0" />
    </w:numPr>
  </w:pPr>
</w:style>
```

end example]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`

Attributes	Description

	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.9.24 rPr (Numbering Symbol Run Properties)

This element specifies the run properties which shall be applied to the numbering level's text specified in the lvlText element (§17.9.11) when it is applied to paragraphs in this document.

These run properties are applied to all numbering level text used by a given abstract numbering definition and numbering level. It should be noted that run properties specified on a numbered paragraph itself, or on text runs within a numbered paragraph, are separate from the run properties specified by rPr elements within a numbering level, as the latter affects only the numbering text itself, not the remainder of runs in the numbered paragraph.

[*Example*: Consider the WordprocessingML below which uses the rPr element to specify that the numbering symbol used within a given numbering level should be bold and of a 16 point font size:

```
<w:lvl w:ilvl="1">
  …
  <w:rPr>
    <w:b />
    <w:sz w:val="32" />
  </w:rPr>
</w:lvl>
```

The resulting paragraph uses its regular paragraph formatting, but the numbering level text itself must be specifically formatted as bold in 16 point font. *end example*]

The W3C XML Schema definition of this element's content model (CT_RPr) is located in §A.1. Each child element from the above table shall not occur more than once. [*Note*: This restriction is not reflected in the element's content model due to limitations of W3C XML Schema language. *end note*]

17.9.25 start (Starting Value)

This element specifies the starting value for the numbering used by the parent numbering level within a given numbering level definition. This value is used when this level initially starts in a document, as well as whenever it is restarted via the properties set in the lvlRestart element (§17.9.10).

If this element is omitted, then the starting value shall be zero (0).

[*Example*: Consider the following WordprocessingML fragment for an abstract numbering definition:

```
<w:abstractNum w:abstractNumId="1">
  …
  <w:lvl w:ilvl="0">
    <w:start w:val="2" />
    <w:numFmt w:val="upperLetter"/>
    …
  </w:lvl>
</w:abstractNum>
```

In this example, since upper case Western letters (upperLetter) are being used as numbering symbols for this numbering level, the first instance of a numbering paragraph associated with this abstract numbering definition and numbering level would have the numbering symbol B, the second letter in the number format.

Subsequent numbered paragraphs with this abstract numbering definition and at this level would have their numbering symbols incremented from B (the starting value for this numbering level). *end example*]

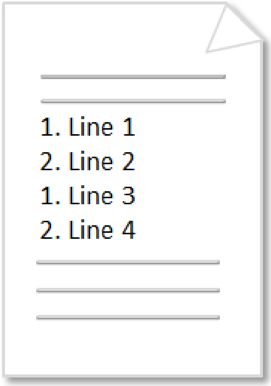
Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.9.26 startOverride (Numbering Level Starting Value Override)

This element specifies the number that the specified level override shall begin with. This value is used to reset the numbering in a given level.

[*Example*: Consider using WordprocessingML to override the numbering of a particular list item and make the list look as follows:



1. Line 1
2. Line 2
1. Line 3
2. Line 4

Firstly, create two w:num instances that reference the same abstract numbering definition. One of the w:num instance overrides the numbering value for the first level. Below is the Wordprocessing ML that illustrates this:

```
<w:num w:numId="5">
  <w:abstractNumId w:val="4"/>
</w:num>

<w:num w:numId="6">
  <w:abstractNumId w:val="4"/>
  <w:lvlOverride w:ilvl="0">
    <w:startOverride w:val = "1"/>
  </w:lvlOverride>
</w:num>
```

Then, in the main document, number the first and the second paragraph as "1" and "2" respectively by using numId 5, and reset the numbering of the third paragraph to "1" by using numId 6. In turn, the numbering of the fourth paragraph becomes "2", because it is incremented based on the new reset value. Below is the WordprocessingML that illustrates this:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0"/>
      <w:numId w:val="5"/>
    </w:numPr>
  </w:pPr>
  <w:r>
```

```
        <w:t> Line 1 </w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:pPr>
        <w:numPr>
          <w:ilvl w:val="0"/>
          <w:numId w:val="5"/>
        </w:numPr>
      </w:pPr>
      <w:r>
        <w:t> Line 2 </w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:pPr>
        <w:numPr>
          <w:ilvl w:val="0"/>
          <w:numId w:val="6"/>
        </w:numPr>
      </w:pPr>
      <w:r>
        <w:t> Line 3 </w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:pPr>
        <w:numPr>
          <w:ilvl w:val="0"/>
          <w:numId w:val="5"/>
        </w:numPr>
      </w:pPr>
      <w:r>
        <w:t> Line 4 </w:t>
      </w:r>
    </w:p>
```

end example]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.9.27 styleLink (Numbering Style Definition)

This element specifies that the parent abstract numbering definition is the base numbering definition for the specified numbering style referenced in its val attribute.

If this element is omitted, or it references a style which does not exist, then this numbering definition shall not be the underlying properties for a numbering style.

[*Note*: Numbering styles are never directly referenced by paragraphs or runs in the document – instead, an abstract numbering definition specifies that it contains the underlying numbering information for a numbering style, and one or more numbering definition instances reference a numbering definition which inherits from it. The numbering style itself is just a friendly name on an abstract numbering definition. *end note*]

[*Example*: Consider the WordprocessingML fragment below, representing an abstract numbering definition which defines the properties for a numbering style:

```
<w:numbering>
  …
  <w:abstractNum w:abstractNumId="5">
    …
    <w:styleLink w:val="ExampleNumberingStyle" />
    …
  </w:abstractNum>
</w:numbering>
…
<w:styles>
  …
  <w:style w:type="numbering" w:styleId="ExampleNumberingStyle">
    <w:name w:val="ExampleNumberingStyle" />
    …
    <w:pPr>
      <w:numPr>
        <w:numId w:val="6" />
      </w:numPr>
    </w:pPr>
  </w:style>
  …
  </w:styles>
```

The styleLink element specifies that the abstract numbering definition defines the properties for a numbering style whose styleId matches its val attribute, and is defined in the styles element of the WordprocessingML. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	
	The contents of this string are interpreted based on the context of the parent XML element.
	
	[*Example*: Consider the following WordprocessingML fragment:
	
	`<w:pPr>`
	  `<w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	  `<w:alias w:val="SDT Title Example" />`
	  …
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

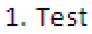
[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.9.28 suff (Content Between Numbering Symbol and Paragraph Text)

This element specifies the content which shall be added between a given numbering level's text and the text of every numbered paragraph which references that numbering level.

If this element is omitted, then its value shall be assumed to be `tab`.

[*Example:* Consider the numbered paragraph below:



1. Test

In this example, a space exists between the numbering symbol `1.` and the numbered paragraph text `Test`. The space would be specified in WordprocessingML as follows:

```
<w:lvl w:ilvl="0">
  …
  <w:suff w:val="space" />
  …
</w:lvl>
```

The suff element with an attribute value of `space` specifies that the character between the numbering's level text and the paragraph text must be a space. *end example*]

Attributes	Description

val (Character Type Between Numbering and Text)	Specifies the character which shall follow the list number.
	[*Example:* Consider a numbered for which a tab exists between the numbering symbol and the numbered paragraph's text. The tab would be specified in WordprocessingML as follows:
	`<w:lvl w:ilvl="0">`
	`  …`
	`  <w:suff w:val="tab" />`
	`  …`
	`</w:lvl>`
	The val attribute with a value of `tab` specifies that the character between the numbering's level text and the paragraph text must be a tab. This tab follows normal tab stop rules. *end example*]
	The possible values for this attribute are defined by the ST_LevelSuffix simple type (§17.18.46).

[*Note*: The W3C XML Schema definition of this element's content model (CT_LevelSuffix) is located in §A.1. *end note*]

17.9.29 tmpl (Numbering Template Code)

This element specifies a unique hexadecimal code which can be used to determine a location within application user interface in which this abstract numbering definition shall be displayed.

If this element is omitted, then this abstract numbering definition can be displayed in any location chosen by the consumer.

[*Example*: Consider the following abstract numbering definition:

```
<w:abstractNum w:abstractNumId="1">
```

```
    …
    <w:tmpl w:val="CA48B6BA" />
    …
  </w:abstractNum>
```

In this example the abstractNum element with attribute abstractNumId equal to 1, would appear in the area within a consumer's application user interface specified by the template code CA48B6BA. *end example*]

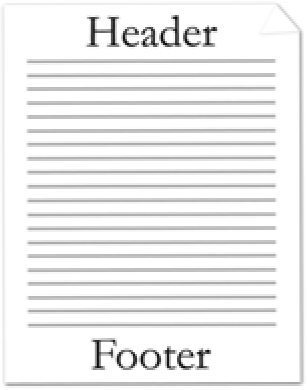
Attributes	Description
val (Long Hexadecimal Number Value)	Specifies a number value specified as a four digit hexadecimal number), whose contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following value for an attribute of simple type ST_LongHexNumber: 00BE2C6C. This value is permitted, as it contains four hexadecimal digits, each an encoding of an octet of the actual decimal number value. It can therefore be interpreted as desired in the context of the parent XML element, *end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element's content model (CT_LongHexNumber) is located in §A.1. *end note*]

17.10 Headers and Footers

Headers and footers refer to text, graphics or data (such as page number, date, document title, and so on) that can appear at the top or bottom of each page in a WordprocessingML document.

A header appears in the top margin (above the main document content on the page), while a footer appears in the bottom margin of a document page (below the main document content on the page).



©ISO/IEC 2012 – All rights reserved

Since WordprocessingML is a flow-based format, headers and footers are applied by specifying the headers and footers for all pages in a particular section of a document.

Within each section of a document there can be up to three different types of headers and footers:

- First page header/footer
- Odd page header/footer
- Even page header/footer

First page headers and footers specify a unique header or footer which shall appear on the first page of a section. Odd page headers and footers specify a unique header and footer which shall appear on all odd numbered pages for a given section. Even page headers and footers specify a unique header and footer which shall appear on all even numbered pages in a given section.

17.10.1 evenAndOddHeaders (Different Even/Odd Page Headers and Footers)

This element specifies whether sections in this document shall have different headers and footers for even and odd pages (an odd page header/footer and an even page header/footer).

If the val attribute is set to `true`, then each section in the document shall use an odd page header for all odd numbered pages in the section, and an even page header for all even numbered pages in the section (counting from the starting value of page numbering for the parent section to determine if the first page is even or odd, as specified with the start attribute on the pgNumType element). If the val attribute is set to `false`, then all pages in a section shall use the odd page header.

This setting does not affect the presence of a first page header on each section, which is specified using the titlePg element (§17.10.6). If a first page header is specified, then all subsequent pages shall have this setting applied, including the first page in the odd/even page count.

If this element is set to `false` and an even page header/footer is specified , then it shall be ignored and only the odd page header/footer shall be displayed. Conversely, if this element is set to true and either header/footer type is omitted for a given section, the appropriate header/footer should be inherited from the previous section; if this is the first section in the document, then a blank header/footer shall be created as needed (another header/footer type shall not be used in its place).

If this element is omitted, then its value shall be assumed to be `false`.

[*Example*: Consider a document which must have a different even and odd page header for each section in its contents. This requirement must be specified using the following WordprocessingML:

```
<w:settings>
  …
  <w:evenAndOddHeaders />
  …
</w:settings>
```

Since the evenAndOddHeaders property is set (and its default value is true), this document now has different headers and footers for even and odd pages. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.10.2 footerReference (Footer Reference)

This element specifies a single footer which shall be associated with the current section in the document. This footer shall be referenced via the id attribute, which specifies an explicit relationship to the appropriate Footer part in the WordprocessingML package.

If the relationship type of the relationship specified by this element is not http://purl.oclc.org/ooxml/officeDocument/relationships/footer, is not present, or does not have a TargetMode attribute value of Internal, then the document shall be considered non-conformant.

Within each section of a document there can be up to three different types of footers:

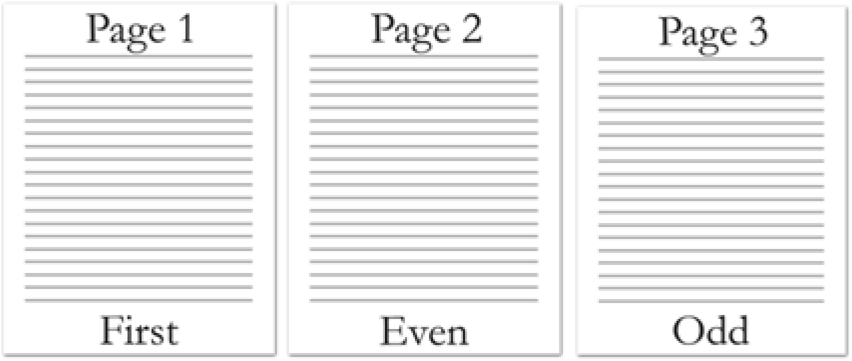
- First page footer
- Odd page footer
- Even page footer

The footer type specified by the current footerReference is specified via the type attribute.

If any type of footer is omitted for a given section, then the following rules shall apply.

- If no footerReference for the first page footer is specified and the titlePg element is specified, then the first page footer shall be inherited from the previous section or, if this is the first section in the document, a new blank footer shall be created. If the titlePg element is not specified, then no first page footer shall be shown, and the odd page footer shall be used in its place.
- If no footerReference for the even page footer is specified and the evenAndOddHeaders element is specified, then the even page footer shall be inherited from the previous section or, if this is the first section in the document, a new blank footer shall be created. If the evenAndOddHeaders element is not specified, then no even page footer shall be shown. and the odd page footer shall be used in its place.
- If no footerReference for the odd page footer is specified then the odd page footer shall be inherited from the previous section or, if this is the first section in the document, a new blank footer shall be created.

[*Example*: Consider a three page document with different first, odd, and even page footers defined as follows:



This document defines three footers, each of have a relationship from the document part with a unique relationship ID, as shown in the following packaging markup:

```
<Relationships
xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  …
  <Relationship Id="rId6"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/footer" Target="footer1.xml"
/>
  <Relationship Id="rId7"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/footer" Target="footer2.xml"
/>
  <Relationship Id="rId10"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/footer" Target="footer3.xml"
/>
  …
</Relationships>
```

These relationships are then referenced in the section's properties using the following WordprocessingML:

```
<w:sectPr>
  …
  <w:footerReference r:id="rId6" w:type="first" />
  <w:footerReference r:id="rId7" w:type="default" />
  <w:footerReference r:id="rId10" w:type="even" />
  …
</w:sectPr>
```

The resulting section must use the footer part with relationship id `rId6` for the first page, the footer part with relationship id `rId10` for all subsequent even pages, and the footer part with relationship id `rId7` for all subsequent odd pages. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

type (Header or Footer Type)	Specifies the type of header or footer specified by the target relationship ID. This header/footer type determines the page(s) on which the current header or footer shall be displayed.
	If any section contains more than a single header or footer of each type, then the document shall be considered non-conformant.
	[*Example*: Consider a document with the following WordprocessingML:
	`<w:sectPr>`
	`  …`
	`  <w:footerReference r:id="rId6" w:type="first" />`
	`  <w:footerReference r:id="rId7" w:type="first" />`
	`  <w:footerReference r:id="rId10" w:type="even" />`
	`  …`
	`</w:sectPr>`
	The resulting section has two footers of type `first`, and therefore is non-conformant. *end example*]

Attributes	Description
	[*Example*: Consider a WordprocessingML section which specifies the following header reference: `<w:headerReference r:id="rId10" w:type="first" />` The resulting section must use the specified header part for the `first` page. *end example*] The possible values for this attribute are defined by the ST_HdrFtr simple type (§17.18.36).

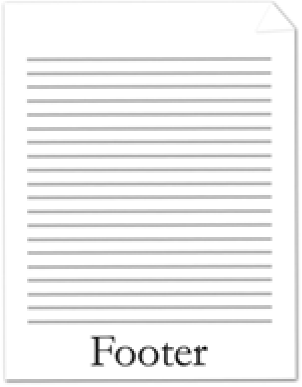
[*Note*: The W3C XML Schema definition of this element's content model (CT_HdrFtrRef) is located in §A.1. *end note*]

17.10.3 ftr (Footer)

This element specifies the content for a single footer for use within one or more sections of a WordprocessingML document.

Within the ftr element, the content of the element is similar to the content of the body (§17.2.2) element, and contains what is referred to as *block-level markup* - markup which can exist as a sibling element to paragraphs in a WordprocessingML document.

[*Example*: Consider the following simple one page document with one footer:

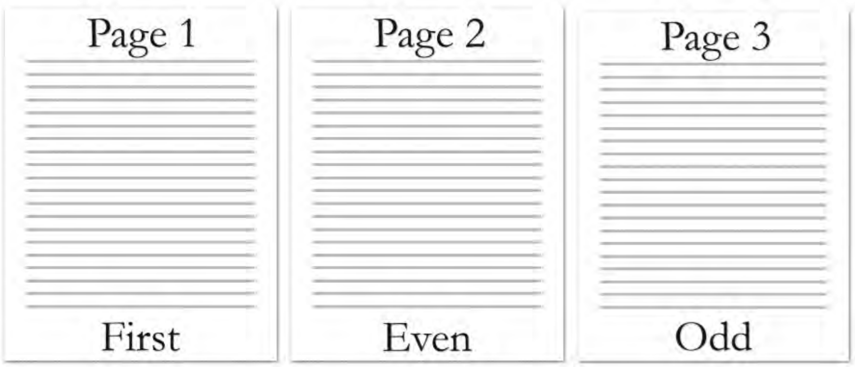


This document defines one footer with the text `footer`. The footer contents are stored in a unique footer part. The resulting footer is represented by the following WordprocessingML:

```
<w:ftr>
  <w:p>
    <w:r>
      <w:t>Footer</w:t>
    </w:r>
  </w:p>
</w:ftr>
```

Since footers are containers of block level contents, all block level elements can be used within them. In this particular example, the content is a single paragraph. *end example*]

[*Example*: Consider a more complex three page document with different first, odd, and even page footers defined:



This document defines three footers stored in three different footer parts. The resulting footers are represented by the following WordprocessingML:

First page footer part:

```
<w:ftr>
  <w:p>
    <w:r>
      <w:t>First</w:t>
    </w:r>
  </w:p>
</w:ftr>
```

Even page footer part:

```
<w:ftr>
  <w:p>
    <w:r>
      <w:t>Even</w:t>
    </w:r>
  </w:p>
</w:ftr>
```

Odd page footer part:

```
<w:ftr>
  <w:p>
    <w:r>
      <w:t>Odd</w:t>
    </w:r>
  </w:p>
</w:ftr>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_HdrFtr) is located in §A.1. *end note*]

17.10.4 hdr (Header)

This element specifies the content for a single header for use within one or more sections of a WordprocessingML document.

Within the hdr element, the content of the element is similar to the content of the body (§17.2.2) element, and contains what is referred to as *block-level markup* - markup which can exist as a sibling element to paragraphs in a WordprocessingML document.

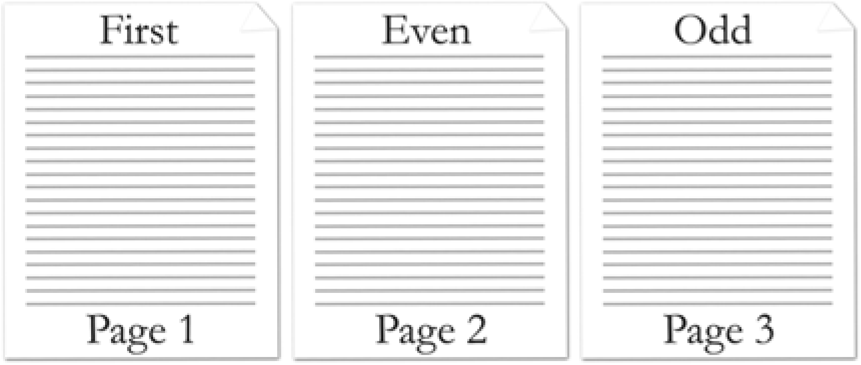
[*Example*: Consider the following simple one page document with one header:

This document defines one header with the text Header. The header's contents is stored in a unique Header part. The resulting header is represented by the following WordprocessingML:

```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>Header</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Since headers are containers of block level contents, all block level elements can be used within them. In this particular example, the content is a single paragraph. *end example*]

[*Example*: Consider a more complex three page document with different first, odd, and even page headers defined:



This document defines three headers stored in three different header parts. The resulting headers are represented by the following WordprocessingML:

First page header part:

```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>First</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Even page header part:

```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>Even</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

Odd page header part:

```
<w:hdr>
  <w:p>
    <w:r>
      <w:t>Odd</w:t>
    </w:r>
  </w:p>
</w:hdr>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_HdrFtr) is located in §A.1. *end note*]

17.10.5 headerReference (Header Reference)

This element specifies a single header which shall be associated with the current section in the document. This header shall be referenced via the id attribute, which specifies an explicit relationship to the appropriate Header part in the WordprocessingML package.

If the relationship type of the relationship specified by this element is not http://purl.oclc.org/ooxml/officeDocument/relationships/header, is not present, or does not have a TargetMode attribute value of `Internal`, then the document shall be considered non-conformant.

Within each section of a document there can be up to three different types of headers:

- First page header
- Odd page header
- Even page header

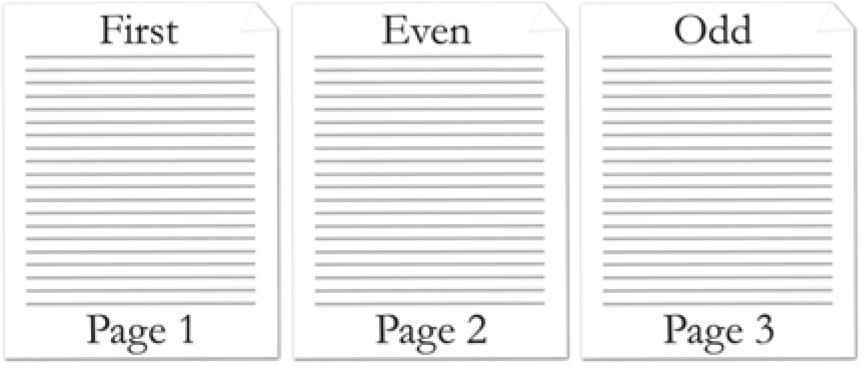
The header type specified by the current headerReference is specified via the type attribute.

If any type of header is omitted for a given section, then the following rules shall apply.

- If no headerReference for the first page header is specified and the titlePg element is specified, then the first page header shall be inherited from the previous section or, if this is the first section in the document, a new blank header shall be created. If the titlePg element is not specified, then no first page header shall be shown, and the odd page header shall be used in its place.

- If no headerReference for the even page header is specified and the evenAndOddHeaders element is specified, then the even page header shall be inherited from the previous section or, if this is the first section in the document, a new blank header shall be created. If the evenAndOddHeaders element is not specified, then no even page header shall be shown, and the odd page header shall be used in its place.
- If no headerReference for the odd page header is specified then the odd page header shall be inherited from the previous section or, if this is the first section in the document, a new blank header shall be created.

[*Example*: Consider a three page document with different first, odd, and even page header defined as follows:



This document defines three headers, each of have a relationship from the document part with a unique relationship ID, as shown in the following packaging markup:

```
<Relationships
xmlns="http://schemas.openxmlformats.org/package/2006/relationships">
  …
  <Relationship Id="rId2"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/header"
Target="header1.xml" />
  <Relationship Id="rId3"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/header"
Target="header2.xml" />
  <Relationship Id="rId5"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/header"
Target="header3.xml" />
  …
</Relationships>
```

These relationships are then referenced in the section's properties using the following WordprocessingML:

```
<w:sectPr>
  …
  <w:headerReference r:id="rId3" w:type="first" />
```

```
    <w:headerReference r:id="rId5" w:type="default" />
    <w:headerReference r:id="rId2" w:type="even" />
    …
  </w:sectPr>
```

The resulting section must use the header part with relationship id `rId3` for the first page, the header part with relationship id `rId2` for all subsequent even pages, and the header part with relationship id `rId5` for all subsequent odd pages. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
type (Header or Footer Type)	Specifies the type of header or footer specified by the target relationship ID. This header/footer type determines the page(s) on which the current header or footer shall be displayed. If any section contains more than a single header or footer of each type, then the document shall be considered non-conformant. [*Example*: Consider a document with the following WordprocessingML: `<w:sectPr>`

Attributes	Description

	`    …`
	`    <w:footerReference r:id="rId6" w:type="first" />`
	`    <w:footerReference r:id="rId7" w:type="first" />`
	`    <w:footerReference r:id="rId10" w:type="even" />`
	`    …`
	`  </w:sectPr>`
	
	The resulting section has two footers of type `first`, and therefore is non-conformant. *end example*]
	
	[*Example*: Consider a WordprocessingML section which specifies the following header reference:
	
	`  <w:headerReference r:id="rId10" w:type="first" />`
	
	The resulting section must use the specified header part for the `first` page. *end example*]
	
	The possible values for this attribute are defined by the ST_HdrFtr simple type (§17.18.36).

[*Note*: The W3C XML Schema definition of this element's content model (CT_HdrFtrRef) is located in §A.1. *end note*]

17.10.6 titlePg (Different First Page Headers and Footers)

This element specifies whether the parent section in this document shall have a different header and footer for its first page.

If the val attribute is set to `true`, then the parent section in the document shall use a first page header for the first page in the section. If the val attribute is set to `false`, then the first page in the parent section shall use the starting value of page numbering for the parent section to determine if the first page should use the even or odd header/footer, as specified with the start attribute on the pgNumType element.

This setting does not affect the presence of even and odd page header on all sections, which is specified using the evenAndOddHeaders element (§17.10.1).

If this element is set to `false` and a first page header/footer is specified , then it shall be ignored and only the odd page header/footer shall be displayed. Conversely, if this element is set to `true` and the first page header/footer type is omitted for the given section, the first page header/footer should be inherited from the previous section; if this is the first section in the document, then a blank header/footer shall be created (another header/footer type shall not be used in its place).

If this element is omitted, then its value shall be assumed to be `false`.

[*Example*: Consider a section which must have a different first page header. This requirement is specified using the following WordprocessingML:

```
<w:sectPr>
  …
  <w:titlePg />
  …
</w:sectPr>
```

Since the titlePg property is present (and its default attribute value is `true`), this document now has a different header and footer for its first page. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.11 Footnotes and Endnotes

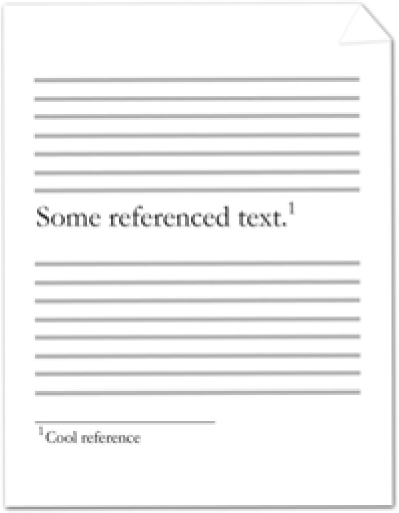
Footnotes and *endnotes* are separate text stories used in documents and books to show the source of borrowed material or to enter explanatory or supplementary information which does not interrupt the normal reading flow of the document.

Footnotes are typically located at the bottom of a page or beneath text being referenced, and *endnotes* are typically placed at the end of a document or at the end of a section. If document has been divided up into one or more sections, each section of a document can contain endnotes.

Both footnotes and endnotes consist of two parts:

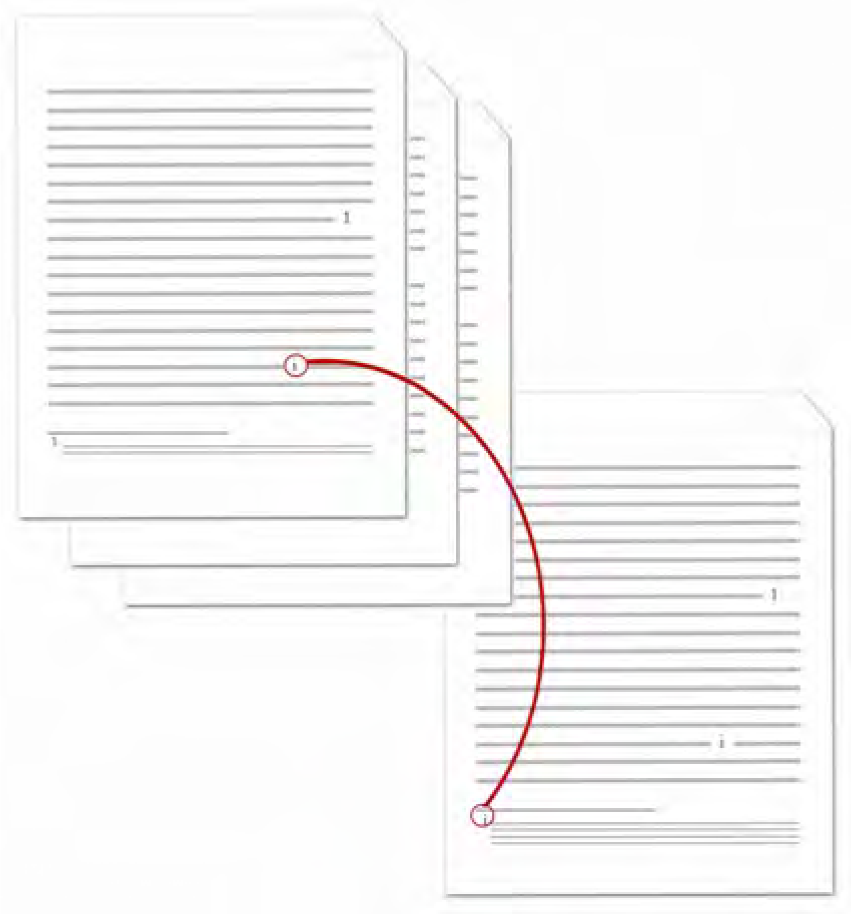
- A note reference mark in the body text to indicate that additional information is in a footnote or endnote, with a numbering system used for each to tell readers whether to look for the note at the end of the page or the end of the document or section.
- The actual footnote or endnote story content.

[*Example:* Example of a footnote applied to text in a document:



The note reference mark follows the noted text and specifies that there is associated footnote information, and the footnote itself is at the bottom of the current page. *end example*]

[*Example:* Consider the following example of an endnote applied to text in a document:



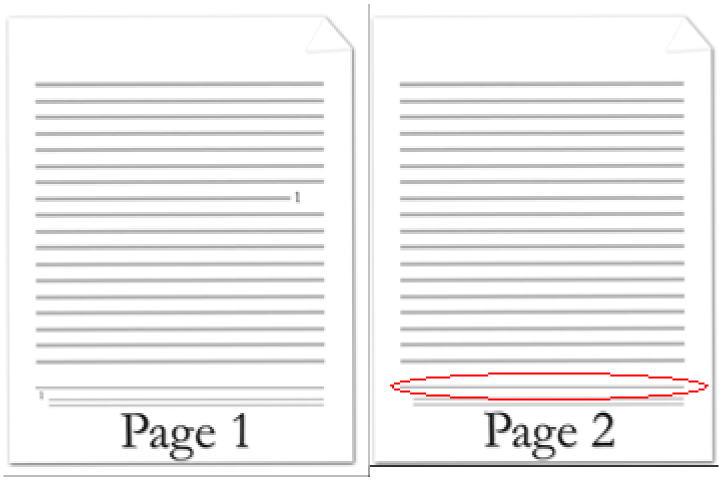
The note reference mark follows the noted text and specifies that there is associated endnote information, and the endnote itself is at the end of the current section. *end example*]

17.11.1 continuationSeparator (Continuation Separator Mark)

This element specifies the presence of a continuation separator mark within the current run. A continuation separator mark is a horizontal line which spans the width of the main story's text extents.

[*Note*: The continuation separator mark is typically used within the context of continuation separator footnotes or endnotes. These footnote and endnote types define the footnote/endnote used to separate the contents of the main document story from continuation of footnotes or endnotes which began on a previous page. *end note*]

[*Example*: Consider the following two pages in a document, where some text is referenced by a footnote that extends to the next page (with the continuation separator circled in red):



The line separating the document text from the footnote that is continued on the next page is represented by the following WordprocessingML:

```
<w:footnote w:type="continuationSeparator" w:id="1">
  <w:p>
    <w:r>
      <w:continuationSeparator />
    </w:r>
  </w:p>
</w:footnote>
```

In this example, the footnote has a content which consists of a single continuationSeparator, which is displayed as a horizontal line across the text extents. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.11.2 endnote (Endnote Content)

This element specifies the content of a single endnote within a WordprocessingML document. Each endnote shall be represented by a single endnote element, which can contain any *block-level content*.

[*Example*: Consider a document with a single endnote, identified by a endnote element, defined in the endnotes part:

```
<w:endnotes>
  <w:endnote w:id="2">
    <w:p>
      <w:pPr>
        <w:pStyle w:val="EndnoteText" />
      </w:pPr>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="EndnoteReference" />
        </w:rPr>
        <w:endnoteRef />
      </w:r>
      <w:r>
        <w:t xml:space="preserve">This is an endnote</w:t>
      </w:r>
    </w:p>
  </w:endnote>
</w:endnotes>
```

This endnote contains an endnote reference mark, as well as the endnote text `This is an endnote`. *end example*]

Attributes	Description

id (Footnote/Endnote ID)	Specifies a unique ID which shall be used to match the contents of a footnote or endnote to the associated footnote/endnote reference mark in the document using the footnoteRef or endnoteRef element, as appropriate.
	
	If this attribute is omitted, then this footnote or endnote shall have no ID. If more than one footnote shares the same ID, then this document shall be considered non-conformant. If more than one endnote shares the same ID, then this document shall be considered non-conformant.
	
	[*Example*: Consider the following footnote as defined in the footnotes part:
	
	`<w:footnotes>`
	`  <w:footnote w:type="normal" w:id="0">`
	`    …`
	`  </w:footnote>`

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	`  …`
	`</w:footnotes>`
	
	The contents of this footnote are associated with the footnoteReference with a matching ID, as follows:
	
	`<w:p>`
	`  <w:r>`
	`    <w:footnoteReference w:id="0" />`
	`  </w:r>`
	`</w:p>`
	
	The resulting paragraph has a footnote reference mark which references the footnote number value of the footnote with an id of `0`. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

type (Footnote/Endnote Type)	Specifies the type of footnote or endnote contained within the current footnote or endnote content definition.
	
	If this attribute is omitted, then it shall be considered to be of style `normal`. If a footnote or endnote is not of style `normal`, then it shall not be referenced by a footnoteReference or endnoteReference element within the main document story.
	
	[*Example*: Consider the following example of a footnote defined in a WordprocessingML document as follows:
	
	`<w:footnote w:type="continuationSeparator" w:id="1">`
	`  <w:p>`
	`    <w:r>`
	`      <w:continuationSeparator />`
	`    </w:r>`
	`  </w:p>`
	`</w:footnote>`
	
	In this example, the footnote is of style `continuationSeparator` and must be used by a consumer to separate continued footnotes from the main document contents (see simple type for full details). *end example*]
	
	The possible values for this attribute are defined by the ST_FtnEdn simple type (§17.18.33).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FtnEdn) is located in §A.1. *end note*]

17.11.3 endnote (Special Endnote List)

This element specifies the ID for all endnotes which are located in the current document that are not of style normal. Each other type of endnote shall be referenced in this list, or it shall not be loaded. If an endnote is not listed beneath this element, and it is required by the document content, then the document shall be considered non-conformant.

[*Example*: Consider a document that has three endnotes represented by the following WordprocessingML:

```
<w:endnotes …>
  <w:endnote w:type="separator" w:id="0">
    …
  </w:endnote>
  <w:endnote w:type="continuationSeparator" w:id="1">
    …
  </w:endnote>
  <w:endnote w:id="2">
    …
  </w:endnote>
</w:endnotes>
```

Each of the endnotes which are not of style normal must be specified in the endnotePr element, as follows:

```
<w:endnotePr>
  <w:endnote w:id="0" />
  <w:endnote w:id="1" />
</w:endnotePr>
```

This indicates to the consumer that the endnotes with an id attribute value of 0 and 1 are special endnotes, and should be treated accordingly. *end example*]

Attributes	Description

id (Footnote/Endnote ID)	Specifies a unique ID that shall be used to match the contents of a footnote or endnote to the associated footnote/endnote reference mark in the document using the footnoteReference or endnoteReference element, as appropriate.
	If more than one footnote shares the same ID, then this document shall be considered non-conformant. If more than one endnote shares the same ID, then this document shall be considered non-conformant.
	[*Example*: Consider the following footnote as defined in the footnotes part:
	`<w:footnotes>`
	  `<w:footnote w:type="normal" w:id="0">`
	    …
	  `</w:footnote>`

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	`  …`
	`</w:footnotes>`
	The contents of this footnote are associated with the footnoteReference with a matching ID, as follows:
	`<w:p>`
	`  <w:r>`
	`    <w:footnoteReference w:id="0" />`
	`  </w:r>`
	`</w:p>`
	The resulting paragraph has a footnote reference mark which references the footnote number value of the footnote with an id of `0`. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

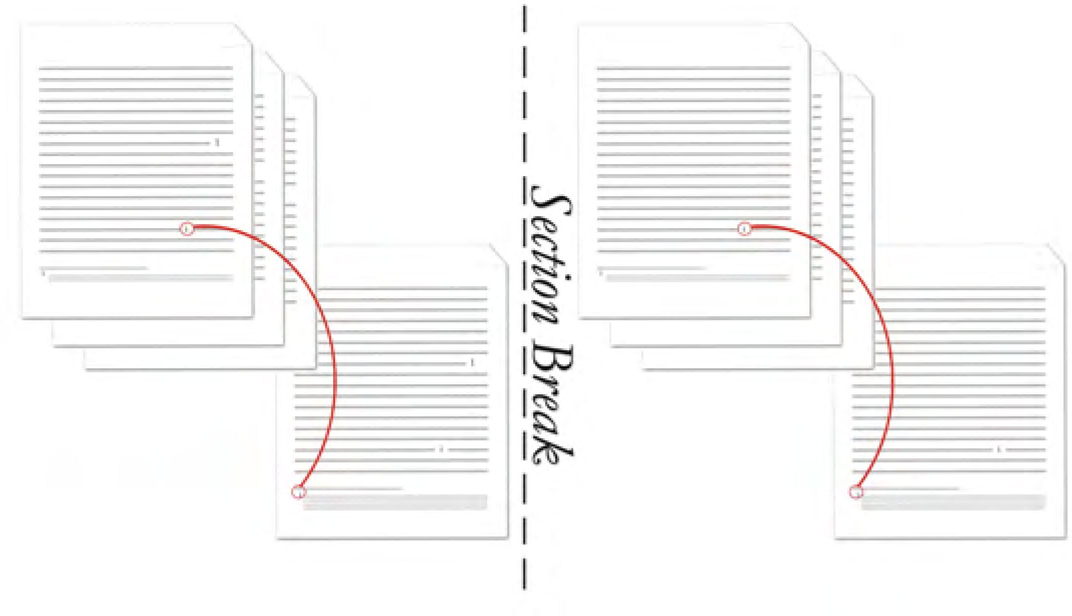
[*Note*: The W3C XML Schema definition of this element's content model (CT_FtnEdnSepRef) is located in §A.1. *end note*]

17.11.4 endnotePr (Document-Wide Endnote Properties)

This element specifies the endnote properties for the current document. Each of these properties are stored as a child element within the endnotePr element.

These properties can be overridden for a specific section via the section-wide endnotePr element (§17.11.5).

[*Example*: Consider the following document with two sections, where the endnotes for each section appears at the end of that section and use lower case roman numerals:



Since both sections are identical, the endnote properties are specified as document-wide level properties (this is not necessary but is most efficient) as follows:

```
<w:settings>
  …
  <w:endnotePr>
    <w:pos w:val="sectEnd"/>
    <w:numFmt w:val="lowerRoman" />
  </w:endnotePr>
  …
</w:settings>
```

Note that the pos element could have been omitted since it is using its default value. *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_EdnDocProps) is located in §A.1. *end note*]

17.11.5 endnotePr (Section-Wide Endnote Properties)

This element specifies the endnote properties for the current section. Each of these properties are an override of the document-wide endnote properties (§17.11.4) and are stored as a child element within the endnotePr element.

If this element is omitted for a given section, then that section shall use the endnote properties defined at the document-wide level.

[*Example*: Consider a document consisting of three sections, which has endnotes in the first section which use lowercase roman numerals, and endnotes in the third section which use the Chicago Manual of Style format. The WordprocessingML for each section would be specified as follows:

```
<w:sectPr>
  <w:endnotePr>
    <w:numFmt w:val="lowerRoman" />
  </w:endnotePr>
</w:sectPr>
…
<w:sectPr>
  …
</w:sectPr>
…
<w:sectPr>
  …
</w:sectPr>
```

This assumes that the document-wide endnote settings are specified to use the Chicago Manual of Style format, as follows:

```
<w:settings>
  <w:endnotePr>
    <w:numFmt w:val="chicago" />
  </w:endnotePr>
</w:settings>
```

The resulting document would override the endnote numbering format for the first section to `lowerRoman`, but would use the `chicago` endnote numbering format for section three (and would also use it for section two if that section had endnotes. *end example*]

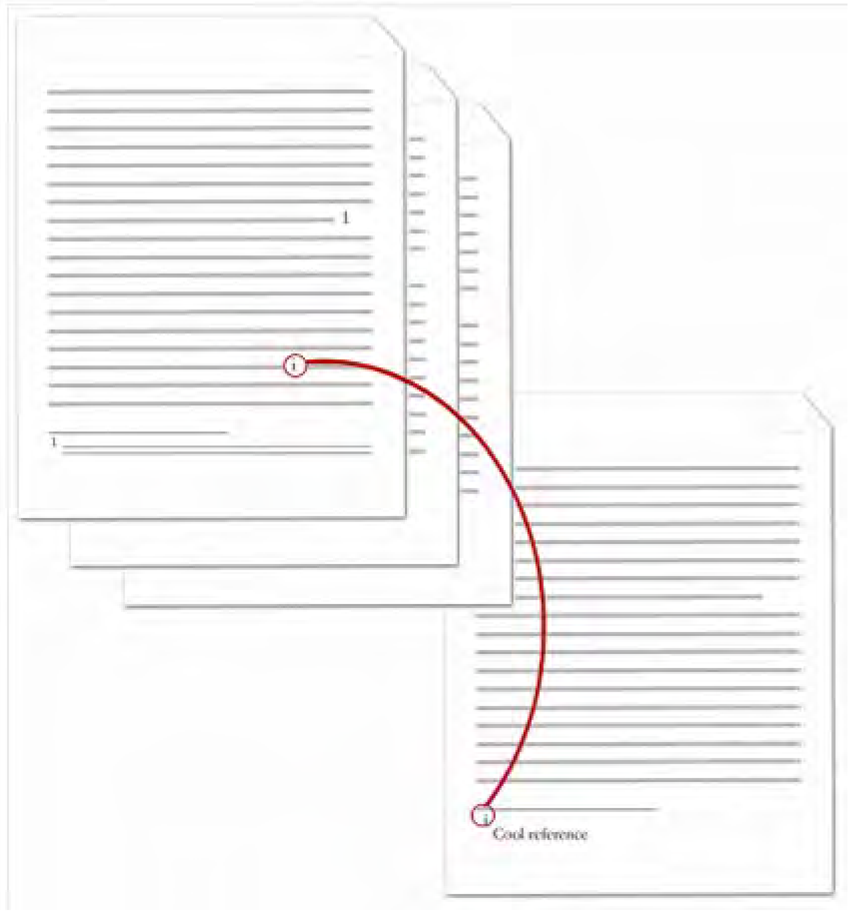
[*Note*: The W3C XML Schema definition of this element's content model (CT_EdnProps) is located in §A.1. *end note*]

17.11.6 endnoteRef (Endnote Reference Mark)

This element specifies the presence of an endnote reference mark. An *endnote reference mark* is a run of automatically numbered text which follows the numbering format set forth via the numFmt element (§17.11.17).

If an endnote reference mark is specified within a run which is not part of an endnote, then that endnote reference mark can be ignored.

[*Example*: Consider the following document where some text is referenced by an endnote at the end of the document:



The endnote reference mark is the lower case roman numeral within the actual endnote itself in the diagram above. The contents of the endnote (including the endnote reference mark) are represented by the following WordprocessingML:

```
<w:endnote w:id="2">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="EndnoteText" />
    </w:pPr>
    <w:r>
      <w:rPr>
        <w:rStyle w:val="EndnoteReference" />
      </w:rPr>
      <w:endnoteRef />
    </w:r>
    <w:r>
      <w:t>Cool reference</w:t>
    </w:r>
  </w:p>
</w:endnote>
```

The resulting endnote contains the literal endnote content of `Cool reference`, preceding by an automatically numbered endnote reference mark. Since this is the first endnote in the document, that automatically numbered reference mark uses the lower case roman numeral `i`. *end example*]

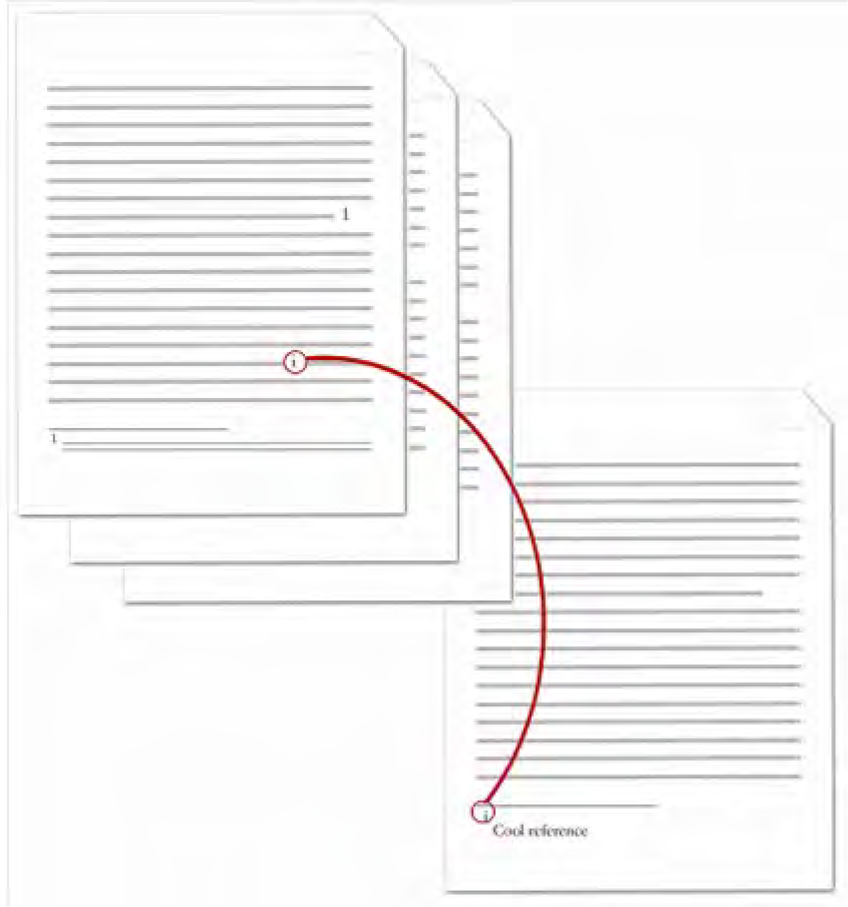
[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.11.7 endnoteReference (Endnote Reference)

This element specifies the presence of an endnote reference. An *endnote reference* is a run of automatically numbered text which references a particular endnote within the parent document, and inherits the endnote reference mark's numbering.

If an endnote reference is specified within a footnote or endnote, then the document shall be considered non-conformant.

[*Example*: Consider the following document where some text is referenced by an endnote at the end of the document:



The endnote reference is the lower case roman numeral within the document content in the diagram above. The contents of the paragraph which contains the endnote reference are represented by the following WordprocessingML:

```
<w:p>
  <w:r>
    <w:t>This text is followed by an endnote</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="EndnoteReference" />
    </w:rPr>
    <w:endnoteReference w:id="2" />
  </w:r>
  <w:r>
    <w:t>.</w:t>
  </w:r>
</w:p>
```

The resulting paragraph contains the literal text content of `This text is followed by an endnote`, followed by an automatically numbered endnote reference. Since this is the first endnote in the document, that automatically numbered reference inherits the lower case roman numeral `i` from the endnote reference mark. *end example*]

Attributes	Description

customMarkFollows (Suppress Footnote/Endnote Reference Mark)	Specifies that the current footnote or endnote shall not have an associated footnote or endnote reference mark, as appropriate.
	
	This attribute shall be used to specify that a particular footnote or endnote shall not increment the numbering for its associated footnote/endnote numbering format, so that the use of a footnote with a custom footnote mark does not cause a missing value in the footnote/endnote values. The display of the mark is specified via the footnoteRef/endnoteRef elements, as appropriate.
	
	If this attribute is omitted, then the footnote or endnote reference mark shall not be skipped when incrementing over this footnote or endnote.
	
	[*Example*: Consider a footnote with an id value of 1 that uses a custom footnote mark:
	
	`<w:footnotes>`
	  `<w:footnote w:id="0">`
	    `…`
	  `</w:footnote>`
	  `<w:footnote w:id="2">`
	    `…`
	  `</w:footnote>`
	  `<w:footnote w:id="2">`
	    `…`
	  `</w:footnote>`
	`</w:footnotes>`

Attributes	Description
	If the numbering format for footnotes in this document is `upperRoman`, then the first footnote must be `I`, the second is suppressed, and the third is `II`, noticing that the second does not increment the numbering sequence. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

id (Footnote/Endnote ID Reference)	Specifies the footnote or endnote which is being referenced by the current footnote or endnote reference in the document.
	
	If the resulting footnote or endnote ID is not present in the footnotes or endnote part (as appropriate), then this document shall be considered non-conformant.
	
	[*Example*: Consider a paragraph with an endnote reference, represented by the following WordprocessingML:
	
	`<w:p>`
	`  <w:r>`
	`    <w:t>This text is followed by an endnote</w:t>`
	`  </w:r>`
	`  <w:r>`
	`    <w:endnoteReference w:id="2" />`
	`  </w:r>`
	`  <w:r>`
	`    <w:t>.</w:t>`
	`  </w:r>`
	`</w:p>`
	
	This text references the endnote in the document's endnotes part which has an id value of 2. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FtnEdnRef) is located in §A.1. *end note*]

17.11.8 endnotes (Document Endnotes)

This element specifies the set of all endnotes in the document, including endnote separators and continuation notices. This element is the root node for the Endnotes part.

[*Example*: Consider the following example of the contents of the endnotes part:

```
<w:endnotes>
  <w:endnote w:type="separator" w:id="0">
    …
  </w:endnote>
  <w:endnote w:type="continuationSeparator" w:id="1">
    …
  </w:endnote>
  <w:endnote w:id="2">
    …
  </w:endnote>
</w:endnotes>
```

The endnotes part contains the definition for one normal endnote, as well as the separator and continuation separator endnote for this document. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Endnotes) is located in §A.1. *end note*]

17.11.9 footnote (Special Footnote List)

This element specifies the ID for all footnotes which are located in the current document that are not of style `normal`. Each other type of footnote shall be referenced in this list, or it shall not be loaded. This means that if a special footnote is not listed beneath this element, and it is required by the document content, then the document shall be considered non-conformant.

[*Example*: Consider a document that has three footnotes represented by the following WordprocessingML:

```
<w:footnotes>
  <w:footnote w:type="separator" w:id="0">
    …
  </w:footnote>
  <w:footnote w:type="continuationSeparator" w:id="1">
    …
  </w:footnote >
  <w:footnote w:id="2">
    …
  </w:footnote>
</w:footnotes>
```

Each of the footnotes which are not of style `normal` must be specified in the footnotePr element, as follows:

```
<w:footnotePr>
  <w:footnote w:id="0" />
  <w:footnote w:id="1" />
</w:footnotePr>
```

This indicates to the consumer that the footnotes with an id attribute value of 0 and 1 are special footnotes, and should be treated accordingly. *end example*]

Attributes	Description

id (Footnote/Endnote ID)	Specifies a unique ID which shall be used to match the contents of a footnote or endnote to the associated footnote/endnote reference mark in the document using the footnoteRef or endnoteRef element, as appropriate.
	If more than one footnote shares the same ID, then this document shall be considered non-conformant. If more than one endnote shares the same ID, then this document shall be considered non-conformant.
	[*Example*: Consider the following footnote as defined in the footnotes part:
	`<w:footnotes>`
	`  <w:footnote w:type="normal" w:id="0">`
	`    …`
	`  </w:footnote>`
	`  …`
	`</w:footnotes>`
	The contents of this footnote are associated with the footnoteReference with a matching ID, as follows:
	`<w:p>`
	`  <w:r>`
	`    <w:footnoteReference w:id="0" />`
	`  </w:r>`
	`</w:p>`
	The resulting paragraph has a footnote reference mark which references the footnote number value of the footnote with an id of 0. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FtnEdnSepRef) is located in §A.1. *end note*]

17.11.10 footnote (Footnote Content)

This element specifies the content of a single footnote within a WordprocessingML document. Each footnote shall be represented by a single footnote element, which can contain any *block-level content*.

[*Example*: Consider a document with a single footnote, identified by a footnote element, defined in the footnotes part as follows:

```
<w:footnotes>
  <w:footnote w:id="2">
    <w:p>
      <w:pPr>
        <w:pStyle w:val="FootnoteText" />
      </w:pPr>
      <w:r>
        <w:rPr>
          <w:rStyle w:val="FootnoteReference" />
        </w:rPr>
        <w:footnoteRef />
      </w:r>
      <w:r>
        <w:t xml:space="preserve">This is a sample footnote</w:t>
      </w:r>
    </w:p>
  </w:footnote>
</w:footnotes>
```

This footnote contains an footnote reference mark, as well as the endnote text `This is a sample footnote`. *end example*]

Attributes	Description

id (Footnote/Endnote ID)	Specifies a unique ID that shall be used to match the contents of a footnote or endnote to the associated footnote/endnote reference mark in the document using the footnoteReference or endnoteReference element, as appropriate.
	
	If this attribute is omitted, then this footnote or endnote shall have no ID. If more than one footnote shares the same ID, then this document shall be considered non-conformant. If more than one endnote shares the same ID, then this document shall be considered non-conformant.
	
	[*Example*: Consider the following footnote as defined in the footnotes part:
	
	`<w:footnotes>`
	`  <w:footnote w:type="normal" w:id="0">`
	`    …`
	`  </w:footnote>`
	`  …`
	`</w:footnotes>`
	
	The contents of this footnote are associated with the footnoteReference with a matching ID, as follows:
	
	`<w:p>`
	`  <w:r>`

Attributes	Description

	`<w:footnoteReference w:id="0" />`
	`</w:r>`
	`</w:p>`
	
	The resulting paragraph has a footnote reference mark which references the footnote number value of the footnote with an id of `0`. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

type (Footnote/Endnote Type)	Specifies the type of footnote or endnote contained within the current footnote or endnote content definition.
	
	If this attribute is omitted, then it shall be considered to be of style `normal`. If a footnote or endnote is not of style `normal`, then it shall not be referenced by a footnoteReference or endnoteReference element within the main document story.
	
	[*Example*: Consider the following example of a footnote defined in a WordprocessingML document as follows:
	
	`<w:footnote w:type="continuationSeparator" w:id="1">`
	`<w:p>`
	`<w:r>`
	`<w:continuationSeparator />`
	`</w:r>`
	`</w:p>`
	`</w:footnote>`
	
	In this example, the footnote is of style `continuationSeparator` and must be used by a consumer to separate continued footnotes from the main document contents (see simple type for full details). *end example*]
	
	The possible values for this attribute are defined by the ST_FtnEdn simple type (§17.18.33).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FtnEdn) is located in §A.1. *end note*]

17.11.11 footnotePr (Section-Wide Footnote Properties)

This element specifies the footnote properties for the current section. Each of these properties are an override of the document-wide footnote properties (§17.11.12) and are stored as a child element within the footnotePr element.

If this element is omitted for a given section, then that section shall use the footnote properties defined at the document-wide level.

[*Example*: Consider a document consisting of three sections, which has footnotes in the first section which appear below text, and footnotes in the third section which appear at the bottom of the page. The WordprocessingML for each section would be specified as follows:

```
<w:sectPr>
  <w:footnotePr>
    <w:pos w:val="beneathText" />
  </w:footnotePr>
</w:sectPr>
…
<w:sectPr>
  …
</w:sectPr>
…
<w:sectPr>
  …
</w:sectPr>
```

This assumes that the document-wide footnote settings are specified as the default positioning at the bottom of the page by omitting the pos element (§17.11.21), as follows:

```
<w:settings>
  <w:footnotePr>
    …
  </w:footnotePr>
</w:settings>
```

The resulting document would override the footnote positioning for the first section to `beneathText`, but would use the `pageBottom` footnote positioning for section three (and would also use it for section two if that section had footnotes. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_FtnProps) is located in §A.1. *end note*]

17.11.12 footnotePr (Document-Wide Footnote Properties)

This element specifies the footnote properties for this document. Each property is stored as a unique element within the footnotePr element.

These properties can be overridden for a specific section via the section-wide footnotePr element (§17.11.11).

[*Example*: Consider the following one page document, where the footnote appears beneath the text it references:

Some referenced text.[1]
[1]Cool reference

Since the document consists of a single footnote, the footnote properties can be stored in either the section-wide or document-wide footnote properties. Assuming that they are stored in the latter, the footnote properties are represented by the following WordprocessingML:

```
<w:settings>
  …
  <w:footnotePr>
    <w:pos w:val="beneathText" />
  </w:footnotePr>
  …
</w:settings>
```

The footnote properties specify that footnotes appear below the noted text *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_FtnDocProps) is located in §A.1. *end note*]

17.11.13 footnoteRef (Footnote Reference Mark)

This element specifies the presence of a footnote reference mark. A *footnote reference mark* is a run of automatically numbered text which follows the numbering format set forth via the footnote numFmt element (§17.11.18).

If a footnote reference mark is specified within a run which is not part of a footnote, then that footnote reference mark can be ignored.

[*Example*: Consider the following document where some text is referenced by a footnote at the end of the page:

Some referenced text.1
1Cool reference

The footnote reference mark is the decimal number within the actual footnote itself in the image above. The contents of the footnote (including the footnote reference mark) are represented by the following WordprocessingML:

```
<w:footnote w:id="2">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="FootnoteText" />
    </w:pPr>
    <w:r>
      <w:rPr>
        <w:rStyle w:val="FootnoteReference" />
      </w:rPr>
      <w:footnoteRef />
    </w:r>
    <w:r>
      <w:t>Cool reference</w:t>
    </w:r>
  </w:p>
</w:footnote>
```

The resulting footnote contains the literal endnote content of `Cool reference`, preceding by an automatically numbered footnote reference mark. Since this is the first footnote in the document, that automatically numbered reference mark uses the first decimal number `1`. It is also important to note that the use of styles `FootnoteText` and `FootnoteReference` is not required, these can simply be added by a particular producer automatically to give the footnote contents are particular style (just like any other use of styles). *end example*]

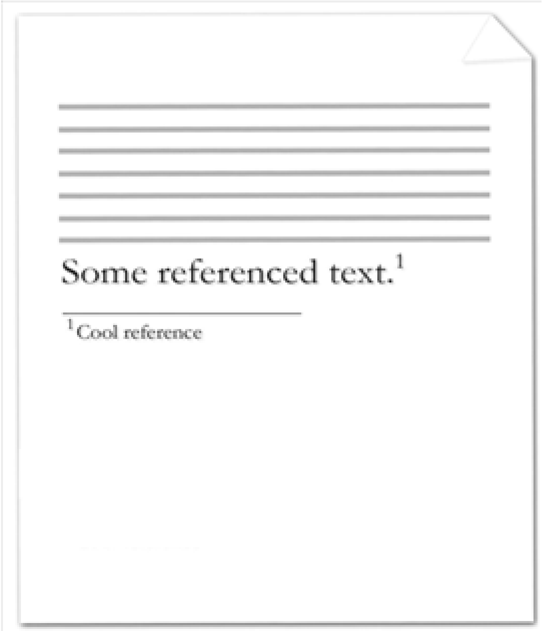
[*Note*: The W3C XML Schema definition of this element’s content model (CT_Empty) is located in §A.1. *end note*]

17.11.14 footnoteReference (Footnote Reference)

This element specifies the presence of a footnote reference. A *footnote reference* is a run of automatically numbered text which references a particular footnote within the parent document, and inherits the footnote reference mark's numbering.

If an footnote reference is specified within a footnote or endnote, then the document shall be considered non-conformant.

[*Example*: Consider the following document where some text is referenced by a footnote at the bottom of the page:



The footnote reference is the superscript decimal number within the document content in the diagram above. The contents of the paragraph which contains the footnote reference are represented by the following WordprocessingML:

```
<w:p>
  <w:r>
    <w:t>Some referenced text.</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="FootnoteReference" />
    </w:rPr>
    <w:footnoteReference w:id="2" />
  </w:r>
</w:p>
```

The resulting paragraph contains the literal text content of `Some referenced text.`, followed by an automatically numbered footnote reference. Since this is the first footnote in the document, that automatically numbered reference inherits the decimal number 1 from the footnote reference mark. *end example*]

Attributes	Description

customMarkFollows (Suppress Footnote/Endnote Reference Mark)	Specifies that the current footnote or endnote shall not have an associated footnote or endnote reference mark, as appropriate.
	
	This attribute shall be used to specify that a particular footnote or endnote shall not increment the numbering for its associated footnote/endnote numbering format, so that the use of a footnote with a custom footnote mark does not cause a missing value in the footnote/endnote values. The display of the mark is specified via the footnoteRef/endnoteRef elements, as appropriate.
	
	If this attribute is omitted, then the footnote or endnote reference mark shall not be skipped when incrementing over this footnote or endnote.
	
	[*Example*: Consider a footnote with an id value of 1 that uses a custom footnote mark:
	
	`<w:footnotes>`
	`  <w:footnote w:id="0">`
	`    …`
	`  </w:footnote>`
	`  <w:footnote w:id="2">`
	`    …`
	`  </w:footnote>`
	`  <w:footnote w:id="2">`
	`    …`
	`  </w:footnote>`
	`</w:footnotes>`
	
	If the numbering format for footnotes in this document is `upperRoman`, then the first footnote must be `I`, the second is suppressed, and the third is `II`, noticing that the second does not increment the numbering sequence. *end example*]

<table>
<thead>
<tr><th>Attributes</th><th>Description</th></tr>
</thead>
<tbody>
<tr><td></td><td>The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).</td></tr>
<tr><td>id (Footnote/Endnote ID Reference)</td><td>Specifies the footnote or endnote which is being referenced by the current footnote or endnote reference in the document.

If the resulting footnote or endnote ID is not present in the footnotes or endnote part (as appropriate), then this document shall be considered non-conformant.

[<em>Example</em>: Consider a paragraph with an endnote reference, represented by the following WordprocessingML:
<pre>
<w:p>
  <w:r>
    <w:t>This text is followed by an endnote</w:t>
  </w:r>
  <w:r>
    <w:endnoteReference w:id="2" />
  </w:r>
  <w:r>
    <w:t>.</w:t>
  </w:r>
</w:p>
</pre>
This text references the endnote in the document's endnotes part which has an id value of 2. <em>end example</em>]

The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).</td></tr>
</tbody>
</table>

[*Note*: The W3C XML Schema definition of this element’s content model (CT_FtnEdnRef) is located in §A.1. *end note*]

17.11.15 footnotes (Document Footnotes)

This element specifies the set of all footnotes in the document, including footnote separators and continuation notices. This element is the root node for the Footnotes part.

[*Example*: Consider the following example of the contents of the footnotes part:

```
<w:footnotes>
  <w:footnote w:type="separator" w:id="0">
    …
  </w:footnote >
  <w:footnote w:type="continuationSeparator" w:id="1">
    …
  </w:footnote>
  <w:footnote w:id="2">
    …
  </w:footnote>
</w:footnotes>
```

The footnotes part contains the definition for one normal footnote, as well as the separator and continuation separator footnotes for this document. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Footnotes) is located in §A.1. *end note*]

17.11.16 noEndnote (Suppress Endnotes In Document)

This element specifies that all endnotes in this document shall not be displayed or printed. If this element is placed on any section break other than the first section break in the document, it shall be ignored.

If this element is omitted, endnotes shall not be suppressed in the current document.

[*Example*: Consider a document in which in the first section endnotes are marked to be hidden:

```
<w:sectPr>
  <w:noEndnote />
</w:sectPr>
```

In this example, this document does not display endnotes. *end example*]

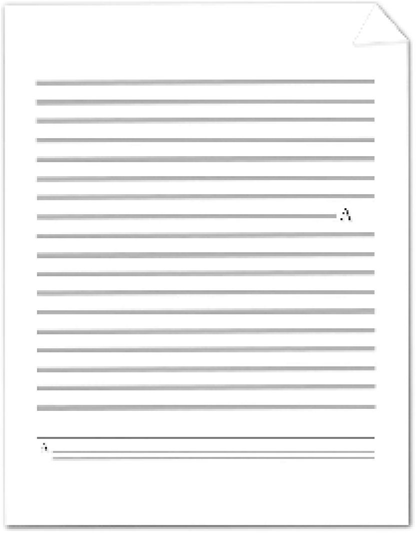
This element's content model is defined by the common boolean property definition in §17.17.4.

17.11.17 numFmt (Endnote Numbering Format)

This element specifies the numbering format that shall be used to determine the endnote reference mark value for all automatically numbered endnote reference marks (those without the suppressRef attribute set).

If this element is omitted, then the numbering format shall be assume to be `decimal`.

[*Example*: Consider the following footnote reference with the number format set to upper case letters:



This footnote numbering format is specified by the following WordprocessingML:

```
<w:footnotePr>
  <w:numFmt w:val="upperLetter" />
</w:footnotePr>
```

end example]

Attributes	Description
format (Custom Defined Number Format)	Specifies a custom number format using the syntax defined by the XSLT format attribute. This format shall be used for all numbering in the parent object. [*Example*: A value of `&#x30A2;` indicates that a consumer must use Katakana numbering. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
val (Numbering Format Type)	Specifies the number format that shall be used for all numbering in the parent object. [*Example*: A value of `lowerLetter` indicates that a consumer must use lowercase letters for each number in this grouping: a,b,c… *end example*] The possible values for this attribute are defined by the ST_NumberFormat simple type (§17.18.59).

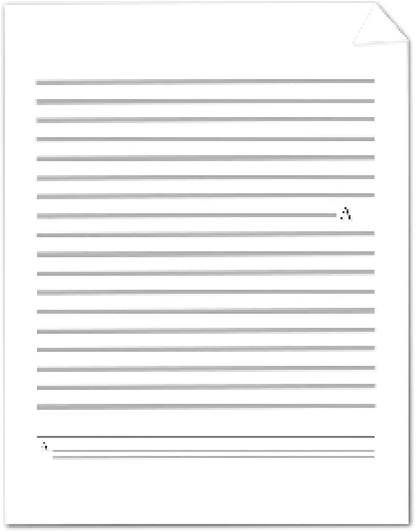
[*Note*: The W3C XML Schema definition of this element's content model (CT_NumFmt) is located in §A.1. *end note*]

17.11.18 numFmt (Footnote Numbering Format)

This element specifies the numbering format that shall be used to determine the footnote reference mark value for all automatically numbered footnote reference marks (those without the suppressRef attribute set).

If this element is omitted, then the numbering format shall be assume to be `decimal`.

[*Example*: Consider the following footnote reference with the number format set to upper case letters:



This footnote numbering format is specified by the following WordprocessingML:

```
<w:footnotePr>
  <w:numFmt w:val="upperLetter" />
</w:footnotePr>
```

end example]

Attributes	Description
format (Custom Defined Number Format)	Specifies a custom number format using the syntax defined by the XSLT format attribute. This format shall be used for all numbering in the parent object. [*Example*: A value of `&#x30A2;` indicates that a consumer must use Katakana numbering. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
val (Numbering Format Type)	Specifies the number format that shall be used for all numbering in the parent object. [*Example*: A value of `lowerLetter` indicates that a consumer must use lowercase letters for each number in this grouping: a,b,c... *end example*]

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	The possible values for this attribute are defined by the ST_NumberFormat simple type (§17.18.59).

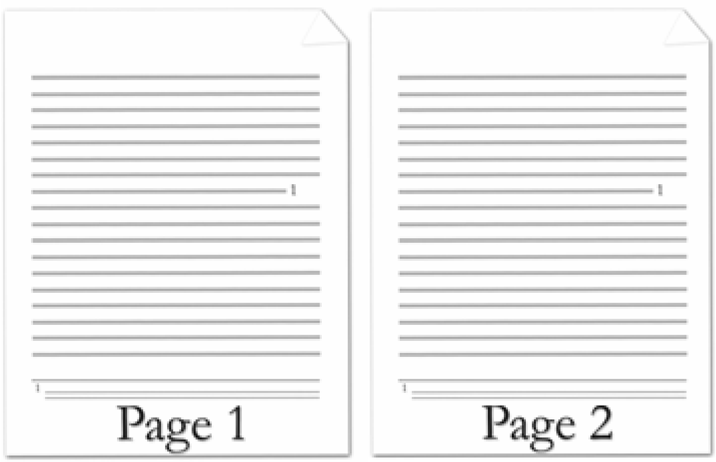
[*Note*: The W3C XML Schema definition of this element's content model (CT_NumFmt) is located in §A.1. *end note*]

17.11.19 numRestart (Footnote and Endnote Numbering Restart Location)

This element specifies when all automatic numbering for the footnote or endnote reference marks shall be restarted. When restarted, the next automatically numbered footnote or endnote in the document (each footnote/endnote type is handled independently) shall restart to the specified numStart value (§17.11.20).

If this element is omitted, then automatic numbering shall not be restarted between each page or section (a vlaue of `continuous`).

[*Example*: Consider the following two page document where the numbering must be reset after each page to its starting value:



The footnote automatic restarting of the numbering is represented by the following WordprocessingML:

```
<w:footnotePr>
  <w:numRestart w:val="eachPage" />
</w:footnotePr>
```

end example]

Attributes	Description
val (Automatic Numbering Restart Value)	Specifies when the automatic numbering shall be restarted for the current set of footnotes or endnotes.

Attributes	Description

	[*Example*: Consider a WordprocessingML document where the numbering for its endnotes must be restarted after each section must be restarted after each page. This setting is represented by the following WordprocessingML:
	`<w:footnotePr>`
	`  <w:numRestart w:val="eachSect" />`
	`</w:footnotePr>`
	The val attribute value of eachSect specifies that numbering shal be restarted after each section. *end example*]
	The possible values for this attribute are defined by the ST_RestartNumber simple type (§17.18.74).

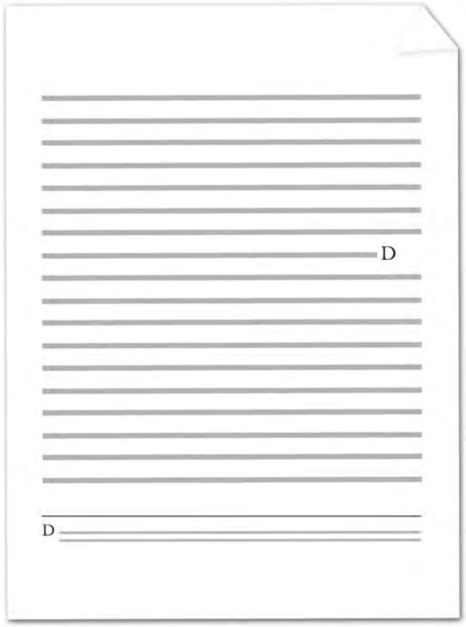
[*Note*: The W3C XML Schema definition of this element's content model (CT_NumRestart) is located in §A.1. *end note*]

17.11.20 numStart (Footnote and Endnote Numbering Starting Value)

This element specifies the starting number or character for the first automatically numbered footnotes or endnote in the document, as well as the first automatically numbered footnotes after each restart point specified by the numRestart element (§17.11.19). This value shall be specified in decimal number units, then translated accordingly to the appropriate numbering format.

If this element is omitted, then the starting value shall be 1.

[*Example*: Consider the following footnote reference with the number format set to upper case letters and starting character set to D:



The number format is specified by the following WordprocessingML:

```
<w:footnotePr>
  <w:numFmt w:val="upperLetter" />
  <w:numStart w:val="4" />
</w:footnotePr>
```

Since D is the fourth letter in the alphabet, the starting character is set to 4. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.11.21 pos (Footnote Placement)

This element specifies where footnotes shall be placed on the page when they are referenced by text in the current document.

If this element is present at the section level, then it shall be ignored.

If this element is omitted at the document level, then footnotes shall be located at the bottom of the current page.

[*Example*: Consider the following one page document, where the footnote appears beneath the text that it is referencing:

Some referenced text.1
1Cool reference

The footnote references the text reading `Some reference text.` and is represented by the following WordprocessingML:

```
<w:p>
  <w:r>
     <w:t>Some referenced text</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:rStyle w:val="FootnoteReference" />
    </w:rPr>
    <w:footnoteReference w:id="2" />
  </w:r>
</w:p>
```

Since the footnote location must be beneath the current text, the section properties must be declared as follows:

```
<w:sectPr>
  …
  <w:footnotePr>
    <w:pos w:val="beneathText" />
  </w:footnotePr>
  …
</w:sectPr>
```

The footnote references the footnote in the footnotes part with an id attribute value equal to 2. Within the section properties of the document, the position of footnotes is specified to be beneath the page's text. *end example*]

Attributes	Description

val (Footnote Position Type)	Specifies the position of footnotes in the document.
	[*Example*: Consider a document in which footnotes must be positioned beneath their text. The footnote properties for this document must be declared as follows:
	`<w:sectPr>`
	`  <w:footnotePr>`
	`    <w:pos w:val="beneathText" />`
	`  </w:footnotePr>`
	`  …`
	`</w:sectPr>`
	The val attribute is `beneathText`, therefore the position of footnotes is specified to be beneath the page's text. *end example*]
	The possible values for this attribute are defined by the ST_FtnPos simple type (§17.18.34).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FtnPos) is located in §A.1. *end note*]

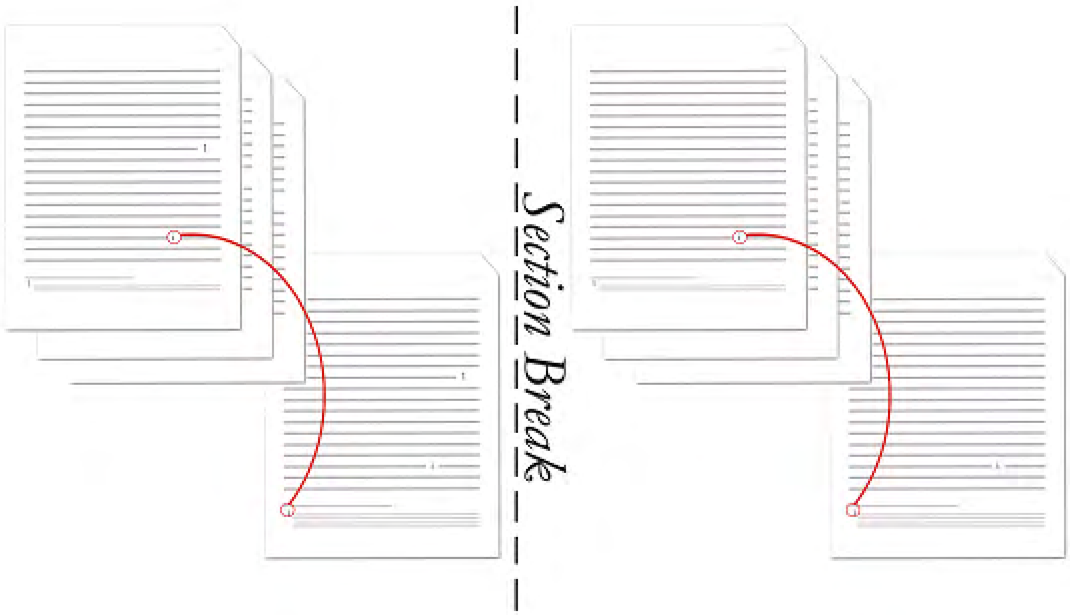
17.11.22 pos (Endnote Placement)

This element specifies where endnotes shall be placed on the page when they are referenced by text in the current document.

If this element is present at the section level, then it shall be ignored.

If this element is omitted at the document level, then endnotes shall be located at the end of the document.

[*Example*: Consider the following multi-page, multi-section document, where the endnote appears at the end of the section in which it is referenced:



The endnote setting is the same for all sections, and is represented by the following WordprocessingML at the document level:

```
<w:settings>
  …
  <w:endnotePr>
    <w:pos w:val="sectEnd" />
    …
  </w:endnotePr>
  …
</w:settings>
```

Within the properties of the document, the position of endnotes is specified to be at the end of each section. *end example*]

Attributes	Description

val (Endnote Position Type)	Specifies the position of endnotes on the parent section or the document.
	[*Example*: Consider a document in which endnotes must be positioned at the end of the section. The section properties for this section must be declared as follows:
	`<w:settings>`
	`  <w:endnotePr>`
	`    <w:pos w:val="sectEnd" />`
	`  </w:endnotePr>`
	`  …`
	`</w:settings>`
	The val attribute is endSect, therefore the position of endnotes is specified to be at the end of the section. *end example*]
	The possible values for this attribute are defined by the ST_EdnPos simple type (§17.18.22).

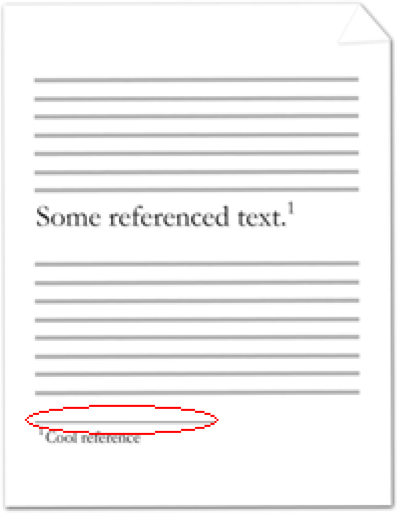
[*Note*: The W3C XML Schema definition of this element's content model (CT_EdnPos) is located in §A.1. *end note*]

17.11.23 separator (Footnote/Endnote Separator Mark)

This element specifies the presence of a separator mark within the current run. A *separator mark* is a horizontal line which spans part of the width text extents.

[*Note*: The separator mark is typically used within the context of separator footnotes or endnotes. These footnote and endnote types define the footnote/endnote used to separate the contents of the main document story from the contents of footnotes or endnotes on that page. *end note*]

[*Example*: Consider the following page in a document, where some text is referenced by a footnote that is located at the bottom of the current page (with the separator circled in red):



The line separating the document text from the normal footnotes is the footnote separator, and is represented by the following WordprocessingML:

```
<w:footnote w:type="separator" w:id="0">
  <w:p>
    <w:r>
      <w:separator />
    </w:r>
  </w:p>
</w:footnote>
```

In this example, the footnote has a content which consists of a single separator, which is displayed as a horizontal line across part of the text extents. *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Empty) is located in §A.1. *end note*]

17.12 Glossary Document

Within a WordprocessingML file, the *glossary document* is a supplemental storage location for additional document content which shall travel with the document, but which shall not be displayed for printed as part of the main document until it is explicitly added to that document by deliberate action.

The glossary document shall also be afforded a separate instance of all of the relationships which are provided on the main document part - this means that the glossary document shall have its own style definitions, numbering definitions, comments, headers, footers, etc. within the WordprocessingML document.

[*Example*: Consider a document which must include ten optional clauses that can be inserted through a user interface. It is clearly not desirable to have these ten clauses appear in the main document story's contents before they are explicitly inserted, therefore each of them can be stored in the glossary document and inserted via the user interface as needed. *end example*]

Within the glossary document, each distinct region of document content is referred to as a *glossary document entry*, and is defined via the docPart element (§17.12.5). These document parts can contain any block-level WordprocessingML element, and can also have a set of classifications and behaviors applied to them via the glossary document entry's properties.

[*Example*: Consider the following definition for the contents of a glossary document part within a WordprocessingML document:

```
<w:glossaryDocument>
  <w:docParts>
    <w:docPart>
      <w:docPartPr>
        …
      </w:docPartPr>
      <w:docPartBody>
        <w:p>
          <w:r>
            <w:t>Sample entry.</w:t>
          </w:r>
        </w:p>
      </w:docPartBody>
    </w:docPart>

    <w:docPart>
      …
    </w:docPart>
  </w:docParts>
</w:glossaryDocument>
```

The glossaryDocument element (§17.12.10) defines the contents of the glossary document part. Within the glossary document, each docPart element contains the definition for one glossary document entry: in this case, there are two entries in the glossary document, the first of which contains a single paragraph with a single run of text. *end example*]

17.12.1 behavior (Entry Insertion Behavior)

This element specifies a single behavior which shall be applied to the contents of the parent glossary document entry (§17.12.5) when it is added to the main document story of a WordprocessingML document. These behaviors shall be used to format the surrounding WordprocessingML around insertion, and do not require the presence of a user interface (i.e. applications without a user interface shall also utilize these settings).

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  <w:docPartPr>
    <w:behaviors>
      <w:behavior w:val="p"/>
    </w:behaviors>
    …
  </w:docPartPr>
  <w:docPartBody>
    <w:p>
      <w:r>
        <w:t>Sample entry.</w:t>
      </w:r>
    </w:p>
  </w:docPartBody>
</w:docPart>
```

The behavior element has a value of p, which specifies that the contents of the parent glossary document entry must be inserted in their own paragraph when they are added to the contents of a document. If the document content to which they are added is defined as follows (and the part is added between the two text runs):

```
<w:body>
  <w:p>
    <w:r>
      <w:t>After this text</w:t>
    </w:r>
    <w:r>
      <w:t>Before this text</w:t>
    </w:r>
  </w:p>
</w:body>
```

This setting specifies that although the part would normally be inserted between the two existing runs in the paragraph, that behavior must ensure that the part is inserted into its own paragraph, resulting in the following WordprocessingML:

```
  <w:body>
    <w:p>
      <w:r>
        <w:t>After this text</w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:r>
        <w:t>Sample entry.</w:t>
      </w:r>
    </w:p>
    <w:p>
      <w:r>
        <w:t>Before this text</w:t>
      </w:r>
    </w:p>
  </w:body>
```

end example]

Attributes	Description

val (Insertion Behavior Value)	Specifies the insertion behavior which shall be associated with the current glossary document entry.
	[*Example*: Consider the WordprocessingML fragment for a glossary document entry's properties, defined as follows:
	`<w:docPartPr>`
	`  <w:behaviors>`
	`    <w:behavior w:val="content"/>`
	`  </w:behaviors>`
	`</w:docPartPr>`
	The val attribute value of `content` specifies that the insertion of this glossary document entry must include only the content (the last paragraph in the part must be merged into the current paragraph in the document). *end example*]
	The possible values for this attribute are defined by the ST_DocPartBehavior simple type (§17.18.15).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocPartBehavior) is located in §A.1. *end note*]

17.12.2 behaviors (Entry Insertion Behaviors)

This element specifies the set of behaviors which shall be applied to the contents of the parent glossary document entry (§17.12.5) when it is added to the main document story of a WordprocessingML document. Since multiple behaviors can be specified for a single part, the sum total of all behaviors shall be used to insert the parent entry into the contents of the WordprocessingML document.

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  <w:docPartPr>
    <w:behaviors>
      <w:behavior w:val="p" />
      <w:behavior w:val="pg" />
    </w:behaviors>
    …
  </w:docPartPr>
  <w:docPartBody>
    <w:p>
      <w:r>
        <w:t>Sample entry.</w:t>
      </w:r>
    </w:p>
  </w:docPartBody>
</w:docPart>
```

The behaviors element contains the set of behaviors which must be applied to this entry when it is inserted into the document, in this case:

- The entry must be inserted into its own paragraph in the document
- The entry must be inserted onto a new page in the document (i.e. it must be preceded by a page break)

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocPartBehaviors) is located in §A.1. *end note*]

17.12.3 category (Entry Categorization)

This element specifies the categorization for the parent glossary document entry. This categorization shall not imply any behaviors around the entry, and is only used to organize the set of glossary document entries within an application or user interface (i.e. to disambiguate between two entries with the same entry name (§17.12.13)).

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:category>
    …
  </w:category>
  …
</w:docPartPr>
```

The category element specifies the categorization applied to the current entry, for the purposes of classification or user interface sorting, for example. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocPartCategory) is located in §A.1. *end note*]

17.12.4 description (Description for Entry)

This element specifies a description for the contents of this glossary document entry. This description can contain any string content, and allows the entry to have additional information contained within the definition for this glossary document entry. [*Note*: This description can be surfaced in a user interface, for example. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  …
  <w:name w:val="Sample Entry" />
  <w:description w:val="This is an example of a glossary document entry for
example purposes." />
  …
</w:docPartPr>
```

The description element specifies that the long description associated with the parent entry must be `This is an example of a glossary document entry for example purposes.` This value can be used as needed by an application, for example, to display in a user interface. *end example*]

Attributes	Description
val (String Value)	Specifies that its contents contain a string. The contents of this string are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment:

Attributes	Description

	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.12.5 docPart (Glossary Document Entry)

This element specifies the details for a single glossary document entry contained in the document. This glossary document entry can consist of one or both of the following:

- The glossary document entry's properties, which define its name, categorization, and behaviors
- The glossary document entry's contents, which consists of one or more block-level elements of WordprocessingML content

Each of these two components is specified by one of the child elements of this element, as seen in the child elements table below.

[*Example*: Consider the following definition for the contents of a Glossary Document part within a WordprocessingML document:

```
<w:glossaryDocument>
  <w:docParts>
    <w:docPart>
      …
    </w:docPart>
    <w:docPart>
      …
    </w:docPart>
  </w:docParts>
</w:glossaryDocument>
```

The docPart element uniquely defines one glossary document entry within the glossary document part, therefore there are two unique entries stored in the current example of a glossary document part. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocPart) is located in §A.1. *end note*]

17.12.6 docPartBody (Contents of Glossary Document Entry)

This element specifies the contents of the parent glossary document entry (§17.12.5). These contents shall consist of one or more block-level elements, analogous to the body element (§17.2.2) of the main document story for the current document.

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  …
  <w:docPartBody>
    <w:p>
      <w:r>
        <w:t>Sample entry.</w:t>
      </w:r>
    </w:p>
  </w:docPartBody>
</w:docPart>
```

The docPartBody element specifies the block-level elements which comprise the contents of the current glossary document entry, in this case, a single paragraph using the p element (§17.3.1.22). *end example*]

When the contents of a glossary document entry are added to a document, the styles, numbering definitions, and all other related parts for this entry shall be taken from the relationships from the Glossary Document part and not from the main document part. These references shall be moved to their main document equivalents when the entry is added to the document.

When the part is inserted, it shall be inserted as though its last paragraph mark does not exist (the content of the final paragraph mark shall be merged with the contents of the paragraph into which this entry is being added).

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  <w:docPartPr>
    <w:behaviors>
      <w:behavior w:val="p"/>
    </w:behaviors>
    …
  </w:docPartPr>
  <w:docPartBody>
    <w:p>
      <w:r>
        <w:t>Sample entry.</w:t>
      </w:r>
    </w:p>
  </w:docPartBody>
</w:docPart>
```

If this entry is inserted into document content to which is defined as follows (and the part is added between the two text runs):

```
<w:body>
  <w:p>
    <w:r>
      <w:t>After this text</w:t>
    </w:r>
    <w:r>
      <w:t>Before this text</w:t>
    </w:r>
  </w:p>
</w:body>
```

This entry has only a single paragraph, which is removed before insertion, and barring any special insertion behaviors (§17.12.2), only the text run is inserted, resulting in the following WordprocessingML:

```
<w:body>
  <w:p>
    <w:r>
      <w:t>After this text</w:t>
    </w:r>
    <w:r>
      <w:t>Sample entry.</w:t>
    </w:r>
    <w:r>
      <w:t>Before this text</w:t>
    </w:r>
  </w:p>
</w:body>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Body) is located in §A.1. *end note*]

17.12.7 docPartPr (Glossary Document Entry Properties)

This element specifies the set of properties which shall be applied to the parent glossary document entry. These properties define its name, categorization, and behaviors.

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  <w:docPartPr>
    <w:name w:val="Sample Entry" />
    …
  </w:docPartPr>
  …
</w:docPart>
```

The docPartPr element specifies the set of properties which have been specified for the parent glossary document entry, the only one visible above being the entry's name of `Sample Entry`. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocPartPr) is located in §A.1. *end note*]

17.12.8 docParts (List of Glossary Document Entries)

This element specifies the collection of glossary document entries which are stored in the current Glossary Document part.

[*Example*: Consider the following definition for the contents of a glossary document part within a WordprocessingML document:

```
<w:glossaryDocument>
  <w:docParts>
    <w:docPart>
      …
    </w:docPart>
    <w:docPart>
      …
    </w:docPart>
  </w:docParts>
</w:glossaryDocument>
```

The docParts element defines the set of entries which are stored in the glossary document part. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocParts) is located in §A.1. *end note*]

17.12.9 gallery (Gallery Associated With Entry)

This element specifies the predefined gallery into which the current glossary document part shall be classified. This classification, although its enumeration values can be interpreted to imply semantics around the contents of the parent glossary document entry, shall only be used to classify and sort this entry (via an application or a user interface).

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:category>
    <w:name w:val="Internal Memo Covers" />
    <w:gallery w:val="coverPg" />
  </w:category>
  …
</w:docPartPr>
```

The gallery element with a value of coverPg specifies that the gallery categorization applied to the current entry, for the purposes of classification or user interface sorting, puts this entry into the Cover Pages classification. *end example*]

Attributes	Description
val (Gallery Value)	Specifies the classification of gallery which shall be associated with the parent glossary document entry. [*Example*: Consider the following WordprocessingML fragment for a single glossary document entry: `<w:gallery w:val="custom1" />`

Attributes	Description
	The val attribute with a value of custom1 specifies that the gallery categorization applied to the current entry is the Custom 1 classification. *end example*] The possible values for this attribute are defined by the ST_DocPartGallery simple type (§17.18.16).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocPartGallery) is located in §A.1. *end note*]

17.12.10 glossaryDocument (Glossary Document Root Element)

This element specifies the root element for a glossary document part within a WordprocessingML document. A glossary document is an supplementary document story in a WordprocessingML that shall be afforded all of the relationships of the Main Document part, such as:

- Style definitions
- Numbering definitions
- Comments
- Headers/footers
- Etc.

The entries stored in this part shall have all of its implicit relationships target these parts, rather than their analogues stored off of the main document part.

[*Example*: Consider the following definition for the contents of a glossary document part within a WordprocessingML document:

```
<w:glossaryDocument>
  <w:docParts>
    <w:docPart>
      …
    </w:docPart>
    <w:docPart>
      …
    </w:docPart>
  </w:docParts>
</w:glossaryDocument>
```

The glossaryDocument element defines the contents of the glossary document part. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_GlossaryDocument) is located in §A.1. *end note*]

17.12.11 guid (Entry ID)

This element specifies a unique identifier (specified using a 128-bit GUID stored on the val attribute) that uniquely identifies this document building block.[*Note*: This unique identifier can be used by an application to uniquely reference a single document building block regardless of different naming, for example when the same part has different names for localization purposes. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  …
  <w:guid w:val="{00000000-5BD2-4BC8-9F70-7020E1357FB2}" />
  …
</w:docPartPr>
```

The guid element specifies that the unique identifier associated with the parent entry must be `{00000000-5BD2-4BC8-9F70-7020E1357FB2}`. This value can be used as needed by an application, for example, to uniquely identify a part regardless of its name. *end example*]

Attributes	Description
val (GUID Value)	Specifies a 128-bit globally unique identifier (GUID) value as defined by the simple type referenced below. The contents of this GUID shall be interpreted based on the context of the parent XML element. If this attribute is omitted, its value shall be assumed to be null (i.e. no GUID shall be associated with the parent XML element). [*Example*: Consider the following WordprocessingML element: `<… w:val="{6A9B8B6F-5BD2-4BC8-9F70-7020E1357FB2}"/>` The val attribute value of `{6A9B8B6F-5BD2-4BC8-9F70-7020E1357FB2}` must be associated with the context of the parent XML element. *end example*] The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Guid) is located in §A.1. *end note*]

17.12.12 name (Category Associated With Entry)

This element specifies the category into which the current glossary document part shall be classified. This classification can consist of any string value as determined by its contents, and shall only be used to classify and sort this entry (via an application or a user interface).

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:category>
    <w:name w:val="Internal Memo Covers" />
    <w:gallery w:val="coverPg" />
  </w:category>
  …
</w:docPartPr>
```

The name element with a value of `Internal Memo Covers` specifies that the category grouping applied to the current entry, for the purposes of classification or user interface sorting, puts this entry into the `Internal Memo Covers` classification. This category can be used as desired. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_String) is located in §A.1. *end note*]

17.12.13 name (Entry Name)

This element specifies a name for the contents of this glossary document entry. This name can contain any string content, and allows the entry to have a friendly identifier contained within the definition for this glossary document entry. [*Note*: This name can be surfaced in a user interface, for example. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  …
  <w:name w:val="Sample Entry" />
  <w:description w:val="This is an example of a glossary document entry for
example purposes." />
  …
</w:docPartPr>
```

The name element specifies that the friendly name associated with the parent entry must be `Sample Entry`. This value can be used as needed by an application, for example, to display in a user interface. *end example*]

Attributes	Description
decorated (Built-In Entry)	Specifies that the name for the current entry is a built-in entry which should not be displayed in the user interface. [*Note*: This information can be used by an application as needed, for example, to disambiguate an entry from one with the same name, ensuring that the built-in entry can be uniquely identified by the application. *end note*] If this attribute is omitted, its value shall be assumed to be `false`. [*Example*: Consider the following WordprocessingML fragment for the name of a single glossary document entry: `<w:name w:decorated="true" w:val=":-)" />` The decorated attribute specifies that the parent entry is a built-in entry, and must be treated as such. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
val (Name Value)	Specifies a string value which contains the name of the current glossary document entry. [*Example*: Consider the following WordprocessingML fragment for the name of a single glossary document entry: `<w:name w:val="Sample Entry" />` The val attribute specifies that the name of the parent entry is `Sample Entry`. *end example*]

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocPartName) is located in §A.1. *end note*]

17.12.14 style (Associated Paragraph Style Name)

This element specifies the style ID for a paragraph style which shall be associated with the current glossary document entry. This paragraph style associated shall not imply anything about the formatting or content of the glossary document entry, and shall only be used to filter and/or sort this entry (via an application or a user interface). [*Note*: One example of the level of classification offered by this element is to only show it as available when the formatting of the paragraph matches the specified style. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:style w:val="Heading1" />
  …
</w:docPartPr>
```

The style element with a val attribute value of `Heading1` specifies that the paragraph style associated with the current glossary document entry must be the style whose style ID is equal to `Heading1`. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`

Attributes	Description

	`…`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.12.15 type (Entry Type)

This element specifies an entry type which shall be applied to the properties of the parent glossary document entry (§17.12.5). Each of these entry types can, based on their values, influence the visibility and behavior of the parent glossary document entry as defined by the associated simple type information.

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:types>
    <w:type w:val="bbPlcHdr" />
  </w:types>
  …
</w:docPartPr>
```

The type element with a value of `bbPlcHdr` specifies that the parent glossary document entry must be treated as if it was the placeholder text for one or more structured document tags in the document. *end example*]

Attributes	Description
val (Type Value)	Specifies the value for the current entry type. [*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry: `<w:type w:val="bbPlcHdr" />` The val attribute value of `bbPlcHdr` specifies that the parent glossary document entry must be treated as if it was the placeholder text for one or more structured document tags in the document. *end example*] The possible values for this attribute are defined by the ST_DocPartType simple type

Attributes	Description
	(§17.18.17).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocPartType) is located in §A.1. *end note*]

17.12.16 types (Entry Types)

This element specifies the set of entry types which shall be applied to the properties of the parent glossary document entry (§17.12.5). Each of these entry types can, based on their values, influence the visibility and behavior of the parent glossary document entry.

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:types>
    …
  </w:types>
  …
</w:docPartPr>
```

The types element specifies the set of entry types which must be associated with the parent glossary document entry. *end example*]

Attributes	Description

all (Entry Is Of All Types)	Specifies that the current glossary document is all entry types. This attribute shall override any information specified as child elements of this element and shall ensure that the current entry is associated with all available entry types.
	If this attribute is omitted, then its default value shall be assumed to be `false`.
	[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:
	`<w:docPartPr>`
	`  <w:types w:all="true">`
	`    <w:type w:val="autoExp" />`
	`  </w:types>`
	`  …`
	`</w:docPartPr>`
	The types element contains a single entry type definition, but because the all attribute is present with a value of `true`, that entry type is augmented to place the parent entry into all possible entry types. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocPartTypes) is located in §A.1. *end note*]

17.13 Annotations

Within a WordprocessingML document, *annotations* refer to various types of supplementary markup which can be stored inside or around a region of text within the document's contents. [*Example*: The types of supplementary information stored within a document can include: comments, revisions, spelling and/or grammatical errors, bookmark information and optional editing permissions. *end example*]

Within a document's contents, annotations are stored in one of three different methods:

- Inline
- "Cross Structure"
- Properties

These three forms are needed in order to maintain compatibility with both the legacy annotations functionality of current word processing applications and the requirements of an XML-based format (i.e. wellformedness of the resulting XML markup). These three forms are referenced within the individual annotation types described in the following sub clauses.

17.13.1 Inline Annotations

Inline annotations describe all annotations which do not require special handling in order to maintain the XML wellformedness requirements of the resulting WordprocessingML output. In these cases, a single XML element shall encapsulate the entire contents of the document content which is being annotated.

[*Example*: Consider the following WordprocessingML markup for a paragraph which reads `The quick brown fox jumps over the jet lagged dog.`, where `jet lagged` replaced the previous text `lazy` when the editing application was tracking revisions:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">The quick brown fox jumps over the </w:t>
  </w:r>
  <w:del … >
    <w:r>
      <w:delText>lazy</w:delText>
    </w:r>
  </w:del>
  <w:ins … >
    <w:r>
      <w:t>jet lagged</w:t>
    </w:r>
  </w:ins>
  <w:r>
    <w:t xml:space="preserve"> dog.</w:t>
  </w:r>
</w:p>
```

The del and ins elements (§17.13.5.14; §17.13.5.18) each fully encapsulate the extent of their respective annotations (a marked deletion and insertion, respectively), as they are inline annotations. *end example*]

17.13.2 "Cross Structure" Annotations

"Cross structure" annotations describe the class of annotations which can span portions of WordprocessingML markup [*Example*: Cross structure annotations can span parts of multiple paragraphs, one half of a custom XML markup element's contents, etc. *end example*]. In these cases, the annotation's region is delimited by two elements: a start element and an end element. These two elements mark the start and end points of the annotated content, but do not contain it. The pairing of the start and end marker are linked via a common value for their id attributes.

[*Example*: Consider the following WordprocessingML markup for two paragraphs, each reading `Example Text`, where a bookmark has been added spanning the second word in paragraph one and the first word in paragraph two:

```
<w:p>
  <w:r>
    <w:t>Example</w:t>
  </w:r>
  <w:bookmarkStart w:id="0" w:name="sampleBookmark" />
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t>Example</w:t>
  </w:r>
  <w:bookmarkEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
```

The bookmarkStart and bookmarkEnd elements (§17.13.6.2; §17.13.6.1) specify the location where the bookmark starts and ends, but cannot contain it because it spans part of two paragraphs. They are part of one group because the id attribute value specifies `0` for both. *end example*]

17.13.3 Property Annotations

Property annotations describe the class of annotations which are stored as a property on an object [*Example*: Property annotations can appear on paragraph properties, run properties, table rows, etc. *end example*] In these cases, the annotation's semantics are defined by the property, as they can affect content and/or formatting.

[*Example*: Consider the following WordprocessingML markup for a paragraph reading `Example Text`, where the first word had the bold property applied when the editing application was tracking revisions:

```
<w:p>
  <w:r>
    <w:rPr>
      <w:b/>
      <w:rPrChange … >
        <w:rPr/>
      </w:rPrChange>
    </w:rPr>
    <w:t>Example</w:t>
  </w:r>
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
```

The rPrChange element (§17.13.5.31; §17.13.5.30) contains the set of previously applied revision properties associated with a particular author at a particular time. It is stored itself as a property on the parent run which was modified. *end example*]

17.13.4 Comments

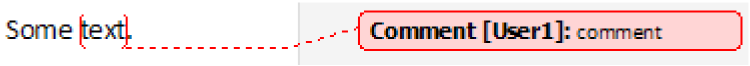
Comments describe annotations which are anchored to a region of document content, but which contain an arbitrary amount of block-level content stored in their own separate document stories. Within a WordprocessingML document, comments are stored in a separate Comments part within the document package.

A comment in a WordprocessingML document is divided into two components:

- The comment anchor (the text on which the comment applies)
- The comment content (the contents of the comment)

The *comment anchor* is the cross structure annotation which defines the region of text on which the comment in anchored. The *comment content* is the content stored in the comments part which contains the actual content of the comment.

[*Example*: Consider a paragraph in a WordprocessingML document whose second word is annotated with a comment:



Some text.

Comment [User1]: comment

The first component to this comment is the document content which defines the extents of the comment and references the specific comment in the comments part:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Some </w:t>
  </w:r>
  <w:commentRangeStart w:id="0" />
  <w:r>
    <w:t>text.</w:t>
  </w:r>
  <w:commentRangeEnd w:id="0" />
  <w:r>
    <w:commentReference w:id="0" />
  </w:r>
</w:p>
```

The commentRangeStart and commentRangeEnd elements (§17.13.4.4; §17.13.4.3) delimit the run content to which the comment with an id of 0 applies (in this case, the single run of text). The following commentReference element (§17.13.4.5) links the preceding run content with a comment in the comments part with an id of 0. Without all three of these elements, the range and comment cannot be linked (although the first two elements are optional, in which case the comment must be anchored at the comment reference mark)

The second component to this comment is the comment content which defines the text in the comment:

```
<w:comment w:id="0" w:author="Joe Smith" w:date="2006-04-06T13:50:00Z"
w:initials="User">
  <w:p>
    <w:pPr>
      <w:pStyle w:val="CommentText" />
    </w:pPr>
    <w:r>
      <w:rPr>
        <w:rStyle w:val="CommentReference" />
      </w:rPr>
      <w:annotationRef />
    </w:r>
    <w:r>
      <w:t>comment</w:t>
    </w:r>
  </w:p>
</w:comment>
```

In this example, the comment specifies that it was inserted by author Joe Smith with the initials User via the author and date attributes. It is linked to the run content via the id attribute, which matches the value of 0 specified using the commentReference element above. The block-level content of the comment specifies that its text is `comment` and the style of the comment content is based off of the character style with the name `CommentReference`. *end example*]

17.13.4.1 annotationRef (Comment Information Block)

This element specifies the presence of an annotation reference mark at the current location in the comment. An *annotation reference mark* is an information block that represents the metadata about the current comment within the document. This annotation reference mark should typically consist of the initials and a unique integer associated with its position in the document, but can be displayed in any desired format.

If this element is omitted from a single comment's contents, then an annotation reference mark can be added at the start of the comment in reading order (right in a right-to-left paragraph or left in a left-to-right paragraph). As well, an annotation reference mark can be relocated as desired within a comment's content.

[*Example*: Consider a document with text with an annotated comment as follows:

Some text[User1].

This comment is represented as the following WordprocessingML fragment:

```
<w:comment … w:initials="User">
  <w:p>
    <w:r>
      <w:annotationRef />
    </w:r>
    …
  </w:p>
</w:comment>
```

The annotationRef element specifies that the comment must start with an annotation reference mark. In this example, this mark is displayed as a combination of the user initial, `User`, and a unique sequential number, `1`. *end example*]

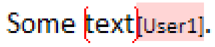
[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.13.4.2 comment (Comment Content)

This element specifies the content of a single comment stored in the Comments part of a WordprocessingML document.

If a comment is not referenced by document content via a matching id attribute on a correct use of the commentReference element (§17.13.4.5), as defined by the WordprocessingML schema, then it can be ignored when loading the document. If more than one comment shares the same value for the id attribute, then only one comment shall be loaded and the others can be ignored.

[*Example*: Consider a document with text with an annotated comment as follows:



Some text[User1].

This comment is represented as the following WordprocessingML fragment:

```
<w:comment w:id="1" w:initials="User" w:author="Dan Battagin">
  …
</w:comment>
```

The comment element specifies the presence of a single comment within the comments part. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type

Attributes	Description
	(§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

initials (Initials of Comment Author)	Specifies the initials of the author of the current comment. [*Note*: This information can be used to format and present the associated comment information block (§17.13.4.1), or in any user interface supported by an application. If there is more than one author with the same initials, it might be more useful to display the author name. *end note*]
	
	If this attribute is omitted, then no author shall be associated with the current comment in the document.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<w:comment w:id="1" w:initials="KB" w:author="Krista Bendig">`
	`  …`
	`</w:comment>`
	
	The initials attribute specifies that the initials of the author of the current comment are KB, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Comment) is located in §A.1. *end note*]

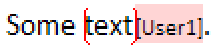
17.13.4.3 commentRangeEnd (Comment Anchor Range End)

This element specifies the end of the range around which a comment is anchored in the content of the WordprocessingML document. The id attribute on this element shall be used to link the corresponding comment anchor range start element and comment reference.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding commentRangeStart element (§17.13.4.4) with a matching id attribute value, then it shall be considered the single anchor point for the associated comment reference.
- If this element appears without a corresponding commentReference element (§17.13.4.5) in the current document story with a matching id attribute value, then the document is non-conformant.
- If this element appears in a comment content story (§17.13.4.2), then the document is non-conformant .

[*Example*: Consider a paragraph in a WordprocessingML document whose second word is annotated with a comment:



Some text[User1].

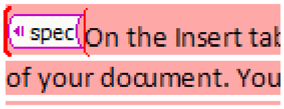
The WordprocessingML fragment for this comment is defined as follows:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Some </w:t>
  </w:r>
  <w:commentRangeStart w:id="0" />
  <w:r>
    <w:t>text.</w:t>
  </w:r>
  <w:commentRangeEnd w:id="0" />
  <w:r>
    <w:commentReference w:id="0" />
  </w:r>
</w:p>
```

The commentRangeEnd element specifies that the end of the comment range for the comment with an id of `0` is after the end of the run containing the word `text`. *end example*]

Attributes	Description
displacedByCustomXml (Annotation Marker Relocated For Custom XML	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and

Attributes	Description

Markup)	custom XML element can differ.
	
	Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example).
	
	If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored.
	
	[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:
	
	
	
	On the Insert tal
	of your document. You
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which must be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`…`
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`  …`
	`</w:p>`
	`…`
	
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and is moved inside the paragraph to be represented physically, the comment with ID 0 must be inside the custom XML, but the comment with ID 1 must be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§17.18.13).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.

Attributes	Description

	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MarkupRange) is located in §A.1. *end note*]

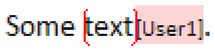
17.13.4.4 commentRangeStart (Comment Anchor Range Start)

This element specifies the start of the range around which a comment is anchored in the content of the WordprocessingML document. The id attribute on this element shall be used to link the corresponding comment anchor range end element and comment reference.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding commentRangeEnd element (§17.13.4.3) with a matching id attribute value, then it shall be considered the single anchor point for the associated comment reference.
- If this element appears without a corresponding commentReference element (§17.13.4.5) in the current document story with a matching id attribute value, then the document is non-conformant .
- If this element appears in a comment content story (§17.13.4.2), then the document is non-conformant .

[*Example*: Consider a paragraph in a WordprocessingML document whose second word is annotated with a comment:

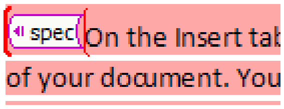


Some text[User1].

The WordprocessingML fragment for this comment is defined as follows:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Some </w:t>
  </w:r>
  <w:commentRangeStart w:id="0" />
  <w:r>
    <w:t>text.</w:t>
  </w:r>
  <w:commentRangeEnd w:id="0" />
  <w:r>
    <w:commentReference w:id="0" />
  </w:r>
</w:p>
```

The commentRangeStart element specifies that the start of the comment range for the comment with an id of 0 is after the end of the run containing the word Some. *end example*]

Attributes	Description
displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element can differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example). If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored. [*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:  spec On the Insert tal of your document. You Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which must be displaced by the physical custom XML element

Attributes	Description

	specifies this information, resulting in the following WordprocessingML:
	`…`
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`  …`
	`</w:p>`
	`…`
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and is moved inside the paragraph to be represented physically, the comment with ID 0 must be inside the custom XML, but the comment with ID 1 must be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§17.18.13).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MarkupRange) is located in §A.1. *end note*]

17.13.4.5 commentReference (Comment Content Reference Mark)

This element specifies the presence of a comment content reference mark, which links the comment content (§17.13.4.2) with the contents of a document story. This link is established by matching the comment whose id attribute matches the id attribute on this element. The resulting comment is anchored to the range with comment range elements with the same id attribute values (if present) as follows:

- If both of the commentRangeStart and commentRangeEnd elements (§17.13.4.4; §17.13.4.3) are present, then the comment reference shall anchor the comment to the resulting range.
- If only one of the commentRangeStart and commentRangeEnd elements (§17.13.4.4; §17.13.4.3) is present, then the document is non-conformant.
- If neither element is present, then the comment reference shall anchor the comment to its current location.

If this element appears in a comment content story (§17.13.4.2), then it can be ignored. If no comment exists with an id attribute which matches the id attribute on this element, then this document is non-conformant.

[*Example*: Consider a paragraph in a WordprocessingML document whose second word is annotated with a comment:

Some text[User1].

The WordprocessingML fragment for this comment is defined as follows:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Some </w:t>
  </w:r>
  <w:commentRangeStart w:id="0" />
  <w:r>
    <w:t>text.</w:t>
  </w:r>
  <w:commentRangeEnd w:id="0" />
  <w:r>
    <w:commentReference w:id="0" />
  </w:r>
</w:p>
```

The commentReference element specifies that the associated comment in the comments part must be the comment whose id attribute value is 0. As well, since a start and end marker exist with a matching ID, this comment is anchored to that region of the document. *end example*]

Attributes	Description
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant.

Attributes	Description

	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Markup) is located in §A.1. *end note*]

17.13.4.6 comments (Comments Collection)

This element specifies all of the comments defined in the current document. It is the root element of the Comments part of a WordprocessingML document.

[*Example*: Consider the following WordprocessingML fragment for the content of a comments part in a WordprocessingML document:

```
<w:comments>
  <w:comment … >
    …
  </w:comment>
</w:comments>
```

The comments element contains the single comment specified by this document in this example. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Comments) is located in §A.1. *end note*]

17.13.5 Revisions

Revisions in WordprocessingML provide a mechanism for storing information about the evolution of the document (i.e. the set of modifications made to a document by one of more authors). When an application adds revisions to the content of a WordprocessingML document, they are specifying this by storing either (depending on the revision type):

- The current state of the document (a deletion stores the current state of the text as deleted, and implies that its original state was the content used to exist)

- The initial state of the document (a run's initial properties are explicitly stored in a previous run properties block, as the current run properties are always those that are the child of the rPr element (§17.7.9.1))

A revision consists of two required pieces of information:

- The revision type (specified via the name of the revision element)
- A unique revision identifier (used to uniquely identify revisions)

As well as optional information:

- The author of the revision
- The date and time of the revision

[*Example*: Consider a paragraph of text in a WordprocessingML document in which one word has been inserted, as follows:

Some text

This paragraph has the word text marked inserted as a revision, and is represented as the following WordprocessingML:

```
<w:p>
  <w:r>
    <w:t>Some</w:t>
  </w:r>
  <w:ins w:id="0" w:author="Joe Smith" w:date="2006-03-31T12:50:00Z">
    <w:r>
      <w:t>text</w:t>
    </w:r>
  </w:ins>
</w:p>
```

The ins element contains both the required information: all of the content which must be treated as revision marked as inserted (the word text); a unique revision identifier of 0.

The element also stores the optional information about the revision: the word text was inserted by Joe Smith on March 31, 2006 at 12:50pm. *end example*]

Within a WordprocessingML document, the following types of revisions can be used to track the changes to a document:

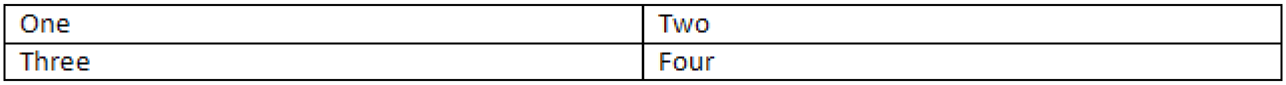
- Insertions
- Deletions
- Moves

- Changes to run/paragraph/table/numbering/section properties
- Changes to custom XML markup

17.13.5.1 cellDel (Table Cell Deletion)

This element specifies that the parent table cell shall be treated as though it was deleted from the document while revisions were being recorded. This means that although the table cell element exists in the structure of the table, the table cell technically no longer exists in the document.

[*Example*: Consider a document with a two row by two columns table as follows:



One	Two
Three	Four

If this table has each cell in its final column deleted and this is tracked as a revision, the resulting WordprocessingML would show each of these cells as deleted as follows:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>One</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:cellDel w:id="0" … />
      </w:tcPr>
      <w:r>
        <w:t>Two</w:t>
      </w:r>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>Three</w:t>
      </w:r>
    </w:tc>
```

```
      <w:tc>
        <w:tcPr>
          <w:cellDel w:id="1" … />
        </w:tcPr>
        <w:r>
          <w:t>Four</w:t>
        </w:r>
      </w:tc>
    </w:tr>
  </w:tbl>
```

The cellDel elements in the table cell properties of the cells with text Two and Four specify that each of those cells have been deleted from the document. Their attributes (omitted) can optionally provide information about the time at which this deletion took place. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`

Attributes	Description
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*] The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

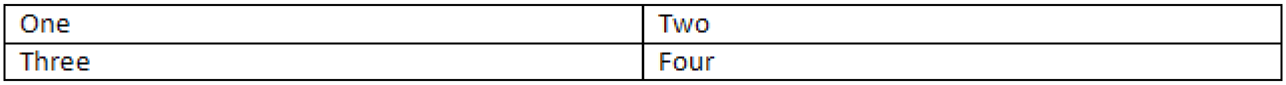
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.2 cellIns (Table Cell Insertion)

This element specifies that the parent table cell shall be treated as though it was inserted into the document while revisions were being recorded.

[*Example*: Consider a document with a two row by two columns table as follows:



One	Two
Three	Four

If this table has two cells added by appending an additional column, and this is tracked as a revision, the resulting WordprocessingML would show each of these cells as inserted as follows:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>One</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:r>
        <w:t>Two</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:cellIns w:id="0" … />
      </w:tcPr>
      <w:r>
        <w:t>New</w:t>
      </w:r>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>Three</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:r>
        <w:t>Four</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:cellIns w:id="1" … />
      </w:tcPr>
      <w:r>
        <w:t>New</w:t>
      </w:r>
    </w:tc>
  </w:tr>
</w:tbl>
```

The cellIns elements in the table cell properties of the cells with text New specify that each of those cells have been inserted into the document. Their attributes (omitted) can optionally provide information about the insertion of these cells (author, date, etc.). *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant.

Attributes	Description

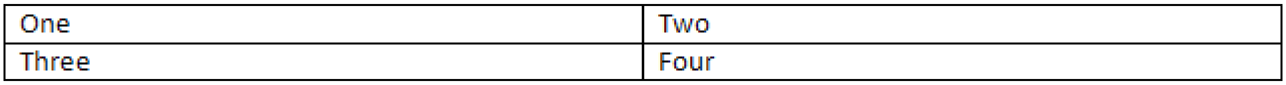
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`…`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.3 cellMerge (Vertically Merged/Split Table Cells)

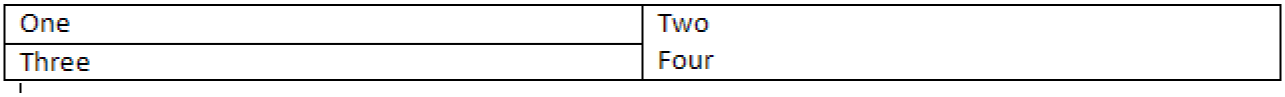
This element specifies that the vertical merge state of the parent table cell has been modified while revisions were being tracked for the document. The vmerge and vmergeOrig attributes on this element specify the original and revised vertical merge states of the table cell.

[*Example*: Consider a document with a two row by two columns table as follows:



One	Two
Three	Four

If this table has the two cells in the second column merged into one and this modification is tracked as a revision, as follows:



One	Two Four
Three	

The resulting WordprocessingML for the revision would appear as follows:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>One</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:r>
        <w:t>Two</w:t>
      </w:r>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:r>
        <w:t>Three</w:t>
      </w:r>
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:cellMerge w:id="0" w:vMerge="cont"/>
      </w:tcPr>
      <w:r>
        <w:t>Four</w:t>
      </w:r>
    </w:tc>
  </w:tr>
</w:tbl>
```

The cellMerge element specifies that changes were made to the vertical merge settings of the last cell in the table, specifically; the cell was vertically merged with the cell above it (gaining a revised vmerge attribute value of cont). *end example*]

Attributes	Description
author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document. If this attribute is omitted, then no author shall be associated with the parent annotation type. [*Example*: Consider a comment represented using the following WordprocessingML fragment:

Attributes	Description

	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is `1`. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

vMerge (Revised	Specifies the vertical merge setting which was applied to the parent table cell by this

Attributes	Description

Vertical Merge Setting)	revision.
	
	If this attribute is omitted, then no revised vertical merge setting is supplied for this revision (if neither this nor the vmergeOrig attribute is specified, the revision can be ignored).
	
	[*Example*: Consider a two row by two column table in which the cells in the second column are merged, and this change is tracked as a revision. The annotation on the last cell in the table would appear as follows:
	
	`<w:tc>`
	`  <w:tcPr>`
	`    <w:cellMerge … w:vMerge="cont" />`
	`  </w:tcPr>`
	`  …`
	`</w:tc>`
	
	The vmerge attribute value of `cont` specifies that the revision on the table cell resulted in it being merged with the previous set of vertically merged cells above it (whether that was one cell or many). *end example*]
	
	The possible values for this attribute are defined by the ST_AnnotationVMerge simple type (§17.18.1).

vMergeOrig (Vertical Merge Setting Removed by Revision)	Specifies the vertical merge setting which was removed from the parent table cell by this revision.
	
	If this attribute is omitted, then the original vertical merge setting shall be assumed to be `rest` (not merged).
	
	[*Example*: Consider a two row by two column table in which the merged cells in the second column are split, and this change is tracked as a revision. The annotation on the last cell in the table would appear as follows:
	
	`<w:tc>`
	`  <w:tcPr>`
	`    <w:cellMerge … w:vMergeOrig="cont" />`
	`  </w:tcPr>`
	`  …`
	`</w:tc>`
	
	The vmergeOrig attribute value of `cont` specifies that the revision on the table cell resulted in it having its vertical merge property removed. *end example*]
	
	The possible values for this attribute are defined by the ST_AnnotationVMerge simple type (§17.18.1).

©ISO/IEC 2012 – All rights reserved

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellMergeTrackChange) is located in §A.1. *end note*]

17.13.5.4 customXmlDelRangeEnd (Custom XML Markup Deletion End)

This element specifies the end of a region in which custom XML markup has been deleted and tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML markup deletion start marker in the document.

Providing a physical representation of custom XML markup results in regions which can be inserted and deleted, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this deletion applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlDelRangeStart element (§17.13.5.5) with a matching id attribute value, then the document is non-conformant .
- If this element and its paired start encapsulate a range with no custom XML markup, then the document is non-conformant .
- If multiple end elements exist with the same id attribute value, then the document is non-conformant .

[*Example*: Consider a document with two inline custom XML markup elements, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

Now, if each custom XML markup element's start and end tag have a physical representation, imagine that the region from the start of the paragraph until the point between the two end points is deleted with revisions enabled. This revision cannot be encapsulated by one del element, since it starts outside of the first custom XML markup element and ends just inside of it, so it must be done using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:p>
```

```
    <w:customXmlDelRangeStart w:id="0" w:author="Devan" />
    <w:customXml … >
      <w:customXml … >
        <w:del … >
          <w:r>
            <w:delText>Text.</w:delText>
          </w:r>
        </w:del>
      </w:customXml>
      <w:customXmlDelRangeEnd w:id="0" />
    </w:customXml>
    <w:r>
      <w:t>More text.</w:t>
    </w:r>
  </w:p>
```

The customXmlDelRangeEnd element delimits the end of the region in which all custom XML elements have been deleted with revisions enabled, and the del element (§17.13.5.14) handles the deletion of the text performed by this revision. Since the end of the outer customXml element was not in the deleted range, it is not revision marked deleted, but the corresponding physical character for the start element is. *end example*]

Attributes	Description

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Markup) is located in §A.1. *end note*]

17.13.5.5 customXmlDelRangeStart (Custom XML Markup Deletion Start)

This element specifies the beginning of a region in which all custom XML markup has been deleted and tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML markup deletion end marker in the document.

Providing a physical representation of custom XML markup results in regions which can be inserted and deleted, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this deletion applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlDelRangeEnd element (§17.13.5.4) with a matching id attribute value, then the document is non-conformant .
- If this element and its paired end encapsulate a range with no custom XML markup, then the document is non-conformant .
- If multiple start elements exist with the same id attribute value, then the document is non-conformant .

[*Example*: Consider a document with two inline custom XML markup elements, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

Now, if each custom XML markup element's start and end tag have a physical representation, imagine that the region from the start of the paragraph until the point between the two end points is deleted with revisions enabled. This revision cannot be encapsulated by one del element, since it starts outside of the first custom XML markup element and ends just inside of it, so it must be done using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:p>
```

```
    <w:customXmlDelRangeStart w:id="0" w:author="Jamie" />
    <w:customXml … >
      <w:customXml … >
        <w:del … >
          <w:r>
            <w:delText>Text.</w:delText>
          </w:r>
        </w:del>
      </w:customXml>
      <w:customXmlDelRangeEnd w:id="0" />
    </w:customXml>
    <w:r>
      <w:t>More text.</w:t>
    </w:r>
  </w:p>
```

The customXmlDelRangeStart element delimits the start of the region in which all custom XML elements have been deleted with revisions enabled, and the del element (§17.13.5.14) handles the deletion of the text performed by this revision. Since the end of the outer customXml element was not in the deleted range, it is not revision marked deleted, but the corresponding physical character for the start element is. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500. If this attribute is omitted, then no date information shall be associated with the parent annotation type.

Attributes	Description

	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.6 customXmlInsRangeEnd (Custom XML Markup Insertion End)

This element specifies the end of a region within which all custom XML markup has been inserted and tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML markup insertion start marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this insertion applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlInsRangeStart element (§17.13.5.7) with a matching id attribute value, then the document is non-conformant .
- If this element and its paired start encapsulate a range with no custom XML markup, then the document is non-conformant .
- If multiple end elements exist with the same id attribute value, then the document is non-conformant .

[*Example*: Consider a document with two inline custom XML markup elements, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

If each custom XML markup element's start and end tag have a physical representation, consider that the inner XML element (but not its content) is inserted with revisions enabled. This revision cannot be encapsulated by one ins element, since the text in the element is not an insertion, so it must be done using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXmlInsRangeStart w:id="0" />
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
    <w:customXmlInsRangeEnd w:id="0" />
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

The customXmlInsRangeEnd element delimits the end of the region in which all custom XML elements have been inserted with revisions enabled. Since this element only affects custom XML, the text is not revision marked inserted, but the corresponding physical characters for the custom XML element are. *end example*]

Attributes	Description

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Markup) is located in §A.1. *end note*]

17.13.5.7 customXmlInsRangeStart (Custom XML Markup Insertion Start)

This element specifies the beginning of a region in which all custom XML markup has been inserted and tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML markup insertion end marker in the document.

Providing a physical representation of custom XML markup start and end tags results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this deletion applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlInsRangeEnd element (§17.13.5.6) with a matching id attribute value, then the document is non-conformant .
- If this element and its paired end encapsulate a range with no custom XML markup, then the document is non-conformant .
- If multiple start elements exist with the same id attribute value, then the document is non-conformant .

This element specifies the end of a region within which all custom XML markup has been inserted and tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML markup insertion start marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this insertion applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlInsRangeStart element (§17.13.5.7) with a matching id attribute value, then it shall be ignored and no insertions shall be present in the document.
- If this element and its paired start encapsulate a range with no custom XML markup, then they shall be ignored and can be omitted when the document is subsequently saved.
- If multiple end elements exist with the same id attribute value, then the first instance in the document shall be used and subsequent elements should be treated as unmatched (no corresponding start).

[*Example*: Consider a document with two inline custom XML markup elements, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

If each custom XML markup element's start and end tag have a physical representation, consider that the inner XML element (but not its content) is inserted with revisions enabled. This revision cannot be encapsulated by one ins element, since the text in the element is not an insertion, so it must be done using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:p>
  <w:customXml … >
    <w:customXmlInsRangeStart w:id="0" />
    <w:customXml … >
      <w:r>
        <w:t>Text.</w:t>
      </w:r>
    </w:customXml>
    <w:customXmlInsRangeEnd w:id="0" />
  </w:customXml>
  <w:r>
    <w:t>More text.</w:t>
  </w:r>
</w:p>
```

The customXmlInsRangeStart element delimits the start of the region in which all custom XML elements have been inserted with revisions enabled. Since this element only affects custom XML, the text is not revision marked inserted, but the corresponding physical characters for the custom XML element are. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500. If this attribute is omitted, then no date information shall be associated with the parent annotation type. [*Example*: Consider a comment represented using the following WordprocessingML fragment:

Attributes	Description

	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.8 customXmlMoveFromRangeEnd (Custom XML Markup Move Source End)

This element specifies the end of a region within which all custom XML markup was moved to another location in the document and this move was tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML move source start marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this move source applies.

The following restrictions shall be applied to this element:

©ISO/IEC 2012 – All rights reserved

- If this element occurs without a corresponding customXmlMoveFromRangeStart element (§17.13.5.9) with a matching id attribute value, then the document is non-conformant .
- If this element and its paired start encapsulate a range with no custom XML markup, then the document is non-conformant .
- If this element and its paired start occur outside of a well-formed move source container (§17.13.5.24; §17.13.5.23) with a matching move destination container (§17.13.5.28; §17.13.5.27), then the document is non-conformant.
- If multiple end elements exist with the same id attribute value, then the document is non-conformant .

[*Example*: Consider a three-paragraph document with a single block-level custom XML markup element, as follows:

```
<w:body>
  <w:p/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:p/>
</w:body>
```

If the second paragraph is moved to the end of the document with revisions enabled. This revision must therefore be stored using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:body>
  <w:p/>
  <w:moveFromRangeStart w:id="0" w:name="move1" w:displacedByCustomXml="next"
w:author="Luna"/>
  <w:customXmlMoveFromRangeStart w:id="1"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveFromRangeEnd w:id="1"/>
  <w:moveFromRangeEnd w:id="0" w:displacedByCustomXml="prev"/>
  <w:p/>
  <w:moveToRangeStart w:id="2" w:name="move1" w:displacedByCustomXml="next"/>
  <w:customXmlMoveToRangeStart w:id="3"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveToRangeEnd w:id="3"/>
  <w:moveFromRangeEnd w:id="2" w:displacedByCustomXml="prev"/>
</w:body>
```

The customXmlMoveFromRangeEnd element delimits the end of the region in which all custom XML elements have been moved from this location with revisions enabled. Since this element only affects custom XML, any text in the region is not revision marked moved by this element when present, but the corresponding physical characters for the custom XML element are. *end example*]

Attributes	Description

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Markup) is located in §A.1. *end note*]

17.13.5.9 customXmlMoveFromRangeStart (Custom XML Markup Move Source Start)

This element specifies the start of a region within which all custom XML markup was moved to another location in the document and this move was tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML move source end marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this move source applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlMoveFromRangeStart element (§17.13.5.9) with a matching id attribute value, then the document is non-conformant .
- If this element and its paired start encapsulate a range with no custom XML markup, then the document is non-conformant .

- If this element and its paired end occur outside of a well-formed move source container (§17.13.5.24; §17.13.5.23) with a matching move destination container (§17.13.5.28; §17.13.5.27), then the document is non-conformant.
- If multiple start elements exist with the same id attribute value, then the document is non-conformant .

[*Example*: Consider a three-paragraph document with a single block-level custom XML markup element, as follows:

```
<w:body>
  <w:p/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:p/>
</w:body>
```

If the second paragraph is moved to the end of the document with revisions enabled. This revision must therefore be stored using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:body>
  <w:p/>
  <w:moveFromRangeStart w:id="0" w:name="move1" w:displacedByCustomXml="next"
w:author="Luke"/>
  <w:customXmlMoveFromRangeStart w:id="1"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveFromRangeEnd w:id="1"/>
  <w:moveFromRangeEnd w:id="0" w:displacedByCustomXml="prev"/>
  <w:p/>
  <w:moveToRangeStart w:id="2" w:name="move1" w:displacedByCustomXml="next"/>
  <w:customXmlMoveToRangeStart w:id="3"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveToRangeEnd w:id="3"/>
  <w:moveFromRangeEnd w:id="2" w:displacedByCustomXml="prev"/>
</w:body>
```

The customXmlMoveFromRangeStart element delimits the start of the region in which all custom XML elements have been moved from this location with revisions enabled. Since this element only affects custom XML, any text in the region is not revision marked moved by this element when present, but the corresponding physical characters for the custom XML element are. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`

Attributes	Description
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.10 customXmlMoveToRangeEnd (Custom XML Markup Move Destination Location End)

This element specifies the end of a region within which all custom XML markup was moved to this location in the document and this move was tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML move destination start marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this move destination applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlMoveToRangeStart element (§17.13.5.11) with a matching id attribute value, then the document is non-conformant.
- If this element and its paired start encapsulate a range with no custom XML markup, then the document is non-conformant.
- If this element and its paired start occur outside of a well-formed move source container (§17.13.5.24; §17.13.5.23) with a matching move destination container (§17.13.5.28; §17.13.5.27), then the document is non-conformant.
- If multiple end elements exist with the same id attribute value, then the document is non-conformant .

[*Example*: Consider a three-paragraph document with a single block-level custom XML markup element, as follows:

```
<w:body>
  <w:p/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:p/>
</w:body>
```

If the second paragraph is moved to the end of the document with revisions enabled. This revision must therefore be stored using the custom XML markup revision "cross structure" syntax, as follows:

```
<w:body>
  <w:p/>
  <w:moveFromRangeStart w:id="0" w:name="move1" w:displacedByCustomXml="next"
w:author="Example Author"/>
  <w:customXmlMoveFromRangeStart w:id="1"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveFromRangeEnd w:id="1"/>
  <w:moveFromRangeEnd w:id="0" w:displacedByCustomXml="prev"/>
  <w:p/>
  <w:moveToRangeStart w:id="2" w:name="move1" w:displacedByCustomXml="next"/>
  <w:customXmlMoveToRangeStart w:id="3"/>
  <w:customXml … >
    <w:p/>
  </w:customXml>
  <w:customXmlMoveToRangeEnd w:id="3"/>
  <w:moveFromRangeEnd w:id="2" w:displacedByCustomXml="prev"/>
</w:body>
```

The customXmlMoveToRangeEnd element delimits the end of the region in which all custom XML elements have been moved to this location with revisions enabled. Since this element only affects custom XML, any text in the region is not revision marked moved by this element when present, but the corresponding physical characters for the custom XML element are. *end example*]

Attributes	Description
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element. If this attribute is omitted, then the document is non-conformant. [*Example*: Consider an annotation represented using the following WordprocessingML fragment:

Attributes	Description

	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Markup) is located in §A.1. *end note*]

17.13.5.11 customXmlMoveToRangeStart (Custom XML Markup Move Destination Location Start)

This element specifies the start of a region within which all custom XML markup was moved to this location in the document and this move was tracked as a revision. The id attribute on this element shall be used to link this element with the corresponding custom XML move destination end marker in the document.

Providing a physical representation of the start and end tags of custom XML markup results in regions which can be inserted and deleted independently, but cannot be encapsulated by a single revision element, since their representation in WordprocessingML is the start or end XML tag for the custom XML markup which it represents. Therefore, the start/end "cross structure" annotation format surrounds the WordprocessingML region to which this move destination applies.

The following restrictions shall be applied to this element:

- If this element occurs without a corresponding customXmlMoveFromRangeEnd element (§17.13.5.8) with a matching id attribute value, then the document is non-conformant.
- If this element and its paired start encapsulate a range with no custom XML markup, then the document is non-conformant.
- If this element and its paired end occur outside of a well-formed move source container (§17.13.5.24; §17.13.5.23) with a matching move destination container (§17.13.5.28; §17.13.5.27), then the document is non-conformant.
- If multiple start elements exist with the same id attribute value, then the document is non-conformant.

[*Example*: Consider a three-paragraph document with a single block-level custom XML markup element, as follows:

```
<w:body>
  <w:p/>
```

```
    <w:customXml … >
      <w:p/>
    </w:customXml>
    <w:p/>
  </w:body>
```

If the second paragraph is moved to the end of the document with revisions enabled. This revision must therefore be stored using the custom XML markup revision "cross structure" syntax, as follows:

```
  <w:body>
    <w:p/>
    <w:moveFromRangeStart w:id="0" w:name="move1" w:displacedByCustomXml="next"
  w:author="Example Author"/>
    <w:customXmlMoveFromRangeStart w:id="1"/>
    <w:customXml … >
      <w:p/>
    </w:customXml>
    <w:customXmlMoveFromRangeEnd w:id="1"/>
    <w:moveFromRangeEnd w:id="0" w:displacedByCustomXml="prev"/>
    <w:p/>
    <w:moveToRangeStart w:id="2" w:name="move1" w:displacedByCustomXml="next"/>
    <w:customXmlMoveToRangeStart w:id="3"/>
    <w:customXml … >
      <w:p/>
    </w:customXml>
    <w:customXmlMoveToRangeEnd w:id="3"/>
    <w:moveFromRangeEnd w:id="2" w:displacedByCustomXml="prev"/>
  </w:body>
```

The customXmlMoveFromRangeStart element delimits the start of the region in which all custom XML elements have been moved from this location with revisions enabled. Since this element only affects custom XML, any text in the region is not revision marked moved by this element when present, but the corresponding physical characters for the custom XML element are. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	`</…>` The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is `1`. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.12 del (Deleted Table Row)

This element specifies that the parent table row shall be treated as a deleted row whose deletion has been tracked as a revision. This setting shall not imply any revision state about the table cells in this row or their contents (which shall be revision marked independently), and shall only affect the table row itself.

[*Example*: Consider a two row by two column table in which the second row has been marked as deleted using a revision. This requirement would be specified using the following WordprocessingML:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:trPr>
      <w:del w:id="0" … />
    </w:trPr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
</w:tbl>
```

The del element on the table row properties for the second table row specifies that this row was deleted, and this deletion was tracked as a revision. *end example*]

Attributes	Description
author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document. If this attribute is omitted, then no author shall be associated with the parent annotation type. [*Example*: Consider a comment represented using the following WordprocessingML fragment: `<… w:id="1" w:author="Example Author">`

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	`…`
	`</…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`…`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is `1`. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.13 del (Deleted Math Control Character)

This element specifies that the Office Open XML Math control character which contains this element was deleted and tracked as a revision. [*Example*: The deletion of a fraction bar. *end example*]

[*Example*: Consider a region of Office Open XML Math in a WordprocessingML document in which the control character for the fraction bar has been deleted, as follows:

$$a+\frac{b}{c}$$

This deletion is represented as the following WordprocessingML:

```
<m:f>
  <m:fPr>
    <m:ctrlPr>
      <w:del w:id="0" w:author="Joe Smith" w:date="2006-03-31T12:50:00Z">
        …
      </w:del>
    </m:ctrlPr>
  </m:fPr>
  …
</m:f>
```

The del element contains all of the content which must be treated as revision marked as deleted; in this case, the fraction bar was deleted by Joe Smith on March 31, 2006 at 12:50pm. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is Example Author, which can be used as desired. *end example*]

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MathCtrlDel) is located in §A.1. *end note*]

17.13.5.14 del (Deleted Run Content)

This element specifies that the inline-level content contained within it shall be treated as deleted content which has been tracked as a revision.

[*Example*: Consider a paragraph of text in a WordprocessingML document in which one word has been deleted, as follows:

Some ~~text~~

This paragraph has the word `text` marked deleted as a revision, and is represented as the following WordprocessingML:

```
<w:p>
  <w:r>
    <w:t>Some</w:t>
  </w:r>
  <w:del w:id="0" w:author="Joe Smith" w:date="2006-03-31T12:50:00Z">
    <w:r>
      <w:delText>text</w:delText>
    </w:r>
  </w:del>
</w:p>
```

The del element contains all of the content which must be treated as revision marked as deleted; in this case, the word `text` was deleted by `Joe Smith` on March 31, 2006 at 12:50pm. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500. If this attribute is omitted, then no date information shall be associated with the parent annotation type.

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

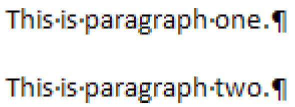
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_RunTrackChange) is located in §A.1. *end note*]

17.13.5.15 del (Deleted Paragraph)

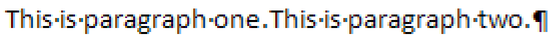
This element specifies that the paragraph mark delimiting the end of a paragraph within a WordprocessingML document shall be treated as deleted (i.e. the contents of this paragraph are no longer delimited by this paragraph mark, and are combined with the following paragraph - but those contents shall not automatically be marked as deleted) as part of a tracked revision.

[*Example*: Consider a document consisting of two paragraphs (with each paragraph delimited by a pilcrow ¶), as follows:


This·is·paragraph·one.¶

This·is·paragraph·two.¶

If the physical character delimiting the end of the first paragraph is deleted and this change is tracked as a revision, resulting in the following:


This·is·paragraph·one.This·is·paragraph·two.¶

This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:rPr>
      <w:del w:id="0" … />
    </w:rPr>
  </w:pPr>
  <w:r>
    <w:t>This is paragraph one.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t>This is paragraph two.</w:t>
  </w:r>
</w:p>
```

The del element on the run properties for the first paragraph mark specifies that this paragraph mark was deleted, and this deletion was tracked as a revision. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`

Attributes	Description
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is `1`. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.16 ins (Inserted Math Control Character)

This element specifies that the Office Open XML Math control character which contains this element was inserted and tracked as a revision. [*Example*: The insertion of a fraction bar. *end example*]

[*Example*: Consider a region of Office Open XML Math in a WordprocessingML document in which the control character for the fraction bar has been inserted, as follows:

$$a+\frac{b}{c}$$

This insertion is represented as the following WordprocessingML:

```
<m:f>
  <m:fPr>
    <m:ctrlPr>
      <w:ins w:id="0" w:author="Joe Smith" w:date="2006-03-31T12:50:00Z">
        …
      </w:ins>
    </m:ctrlPr>
  </m:fPr>
  …
</m:f>
```

The ins element contains all of the content which must be treated as revision marked as inserted; in this case, the fraction bar was inserted by `Joe Smith` on March 31, 2006 at 12:50pm. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation	Specifies the date information for an annotation within a WordprocessingML document.

Attributes	Description

Date)	The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MathCtrlIns) is located in §A.1. *end note*]

17.13.5.17 ins (Inserted Table Row)

This element specifies that the parent table row shall be treated as an inserted row whose insertion has been tracked as a revision. This setting shall not imply any revision state about the table cells in this row or their contents (which shall be revision marked independently), and shall only affect the table row itself.

[*Example*: Consider a two row by two column table in which the second row has been marked as inserted using a revision. This requirement would be specified using the following WordprocessingML:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:trPr>
      <w:ins w:id="0" … />
    </w:trPr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
</w:tbl>
```

The ins element on the table row properties for the second table row specifies that this row was inserted, and this insertion was tracked as a revision. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

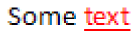
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.18 ins (Inserted Run Content)

This element specifies that the inline-level content contained within it shall be treated as inserted content which has been tracked as a revision.

[*Example*: Consider a paragraph of text in a WordprocessingML document in which one word has been inserted, as follows:



Some text

This paragraph has the word `text` marked inserted as a revision, and is represented as the following WordprocessingML:

```
<w:p>
  <w:r>
    <w:t>Some</w:t>
  </w:r>
  <w:ins w:id="0" w:author="Joe Smith" w:date="2006-03-31T12:50:00Z">
    <w:r>
      <w:t>text</w:t>
    </w:r>
  </w:ins>
</w:p>
```

The ins element contains all of the content which must be treated as revision marked as inserted; in this case, the word `text` was inserted by `Joe Smith` on March 31, 2006 at 12:50pm. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500. If this attribute is omitted, then no date information shall be associated with the parent annotation type.

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

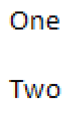
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_RunTrackChange) is located in §A.1. *end note*]

17.13.5.19 ins (Inserted Numbering Properties)

This element specifies that the numbering information defined by the parent element shall be treated as numbering information which was recorded as an insertion using revisions.

[*Example*: Consider two paragraphs in a WordprocessingML document, with the words one and two respectively, as follows:



One

Two

If numbering is then applied to these two paragraphs, and this numbering is tracked as a revision, this revision is represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="1" />
      <w:ins w:id="0" w:author="Joe Smith" w:date="2005-01-01T10:00:00Z" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>one</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:numPr>
      <w:ilvl w:val="0" />
      <w:numId w:val="1" />
      <w:ins w:id="0" w:author="Joe Smith" w:date="2005-01-01T10:00:00Z" />
    </w:numPr>
  </w:pPr>
  <w:r>
    <w:t>two</w:t>
  </w:r>
</w:p>
```

The ins element as a child of the numbering properties specifies that the paragraphs in this document have been given numbering properties by `Joe Smith` and that this change was marked as a revision. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example`

Attributes	Description
	`Author`, which can be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

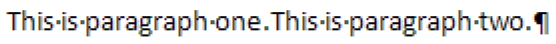
id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.20 ins (Inserted Paragraph)

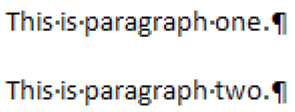
This element specifies that the paragraph mark delimiting the end of a paragraph within a WordprocessingML document shall be treated as deleted (i.e. the contents of this paragraph are no longer delimited by this paragraph mark, and are combined with the following paragraph) as part of a tracked revision.

[*Example*: Consider a document consisting of a single paragraph, as follows:



This·is·paragraph·one.This·is·paragraph·two.¶

If the first sentence is moved into its own new paragraph, and this change is tracked as a revision, resulting in the following:



This·is·paragraph·one.¶

This·is·paragraph·two.¶

This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:pPr>
    <w:rPr>
      <w:ins w:id="0" … />
    </w:rPr>
  </w:pPr>
  <w:r>
    <w:t>This is paragraph one.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t>This is paragraph two.</w:t>
  </w:r>
</w:p>
```

The ins element on the run properties for the first paragraph mark specifies that this paragraph mark was inserted, and this insertion was tracked as a revision. *end example*]

Attributes	Description
author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document. If this attribute is omitted, then no author shall be associated with the parent annotation type.

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.21 moveFrom (Move Source Paragraph)

This element specifies that the parent paragraph has been moved away from this location and tracked as a revision. This does not imply anything about the revision state of the contents of the paragraph, and applies only to the existence of the paragraph as its own unique paragraph.

The following restrictions shall be applied to this content:

- If this element occurs outside of a move source container (§17.13.5.24; §17.13.5.23) for which a matching move destination container (§17.13.5.28; §17.13.5.27) exists in the document, then the document is non-conformant.

[*Example*: Consider a WordprocessingML document in which a paragraph of text is moved down in the document. This moved paragraph would be represented using the following WordprocessingML markup:

```
<w:moveFromRangeStart w:id="0" w:name="aMove"/>
<w:p>
  <w:pPr>
    <w:rPr>
      <w:moveFrom w:id="1" … />
    </w:rPr>
  </w:pPr>
  …
</w:p>
</w:moveFromRangeEnd w:id="0"/>
```

The moveFrom element as a child of the run properties of the paragraph mark specify that this paragraph mark was part of the content which was moved in the document. This implies nothing about the contents, since they can have been added later and not tracked as a revision (they must be marked as a move using the moveFrom element (§17.13.5.22) around the run content). *end example*]

ttributes	Description
author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document. If this attribute is omitted, then no author shall be associated with the parent annotation type.

ttributes	Description

	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is `1`. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type

ttributes	Description
	(§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrackChange) is located in §A.1. *end note*]

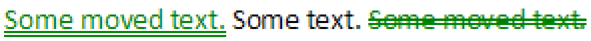
17.13.5.22 moveFrom (Move Source Run Content)

This element specifies that the inline-level content contained within it shall be treated as content which has been moved away from this location and tracked as a revision.

The following restrictions shall be applied to this content:

- If this element occurs outside of a move source container (§17.13.5.24; §17.13.5.23) for which a matching move destination container (§17.13.5.28; §17.13.5.27) exists in the document, then the document is non-conformant.

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this image, green underline indicates the move destination and the green strikethrough indicates the move source location):



Some moved text. Some text. ~~Some moved text.~~

This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

The moveFrom element specifies that all of the inline-level content contained within must be revision marked as content which was moved from its current location in the document. Because this moved content is contained within a complete move source container (moveFromRangeStart and moveFromRangeEnd) with a corresponding move destination, this content is tracked as a move. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.

Attributes	Description

	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_RunTrackChange) is located in §A.1. *end note*]

17.13.5.23 moveFromRangeEnd (Move Source Location Container - End)

This element specifies the end of a region whose move source contents are part of a single named move. When a move source is stored as a revision in a WordprocessingML document, two pieces of information shall be stored about that move source:

- A set of pieces of content which were moved - both inline-level content (§17.13.5.22) and paragraphs (§17.13.5.21)

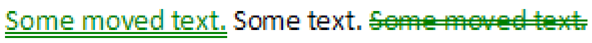
- A move source container (or "bookmark") which specifies that all content within it which marked as a move source is part of a single named move. The name attribute on the move container links a group of move source content with the corresponding group of move destination content.

This element defines the end of the latter piece of the move revision data - the container. The id attribute on this element shall be used to link this element with the corresponding start of a move source container in the document.

The following restrictions are applied to the use of this element:

- If this element occurs without a corresponding moveFromRangeStart element (§17.13.5.24) with a matching id attribute value, then the document is non-conformant.
- If this element and its paired end occur without a matching move destination container (§17.13.5.28; §17.13.5.27), then the document is non-conformant.
- If multiple move source containers surround the same text, the document is non-conformant.

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this image, green underline indicates the move destination and the green strikethrough indicates the move source location):



Some moved text. Some text. ~~Some moved text.~~

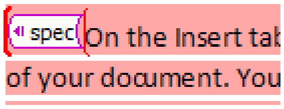
This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

The moveFromRangeEnd element specifies the end of the move source container within which all moved content is part of the move named `move1`. *end example*]

Attributes	Description
displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element can differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example). If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored. [*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:

Attributes	Description

	
	spec On the Insert tab
	of your document. You
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which must be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`…`
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`  …`
	`</w:p>`
	`…`
	
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and is moved inside the paragraph to be represented physically, the comment with ID 0 must be inside the custom XML, but the comment with ID 1 must be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§17.18.13).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MarkupRange) is located in §A.1. *end note*]

17.13.5.24 moveFromRangeStart (Move Source Location Container - Start)

This element specifies the start of a region whose move source contents are part of a single named move. When a move source is stored as a revision in a WordprocessingML document, two pieces of information shall be stored about that move source:

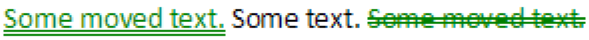
- A set of pieces of content which were moved - both inline-level content (§17.13.5.22) and paragraphs (§17.13.5.21)
- A move source container (or "bookmark") which specifies that all content within it which marked as a move source is part of a single named move. The name attribute on the move container links a group of move source content with the corresponding group of move destination content.

This element defines the start of the latter piece of the move revision data - the container. The id attribute on this element shall be used to link this element with the corresponding end of a move source container in the document.

The following restrictions are applied to the use of this element

- If this element occurs without a corresponding moveFromRangeEnd element (§17.13.5.23) with a matching id attribute value, then the document is non-conformant.
- If this element and its paired end occur without a matching move destination container (§17.13.5.28; §17.13.5.27), then the document is non-conformant.
- If multiple start elements exist with the same id attribute value, then the document is non-conformant.
- If multiple move source containers surround the same text, the document is non-conformant .

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this image, green underline indicates the move destination and the green strikethrough indicates the move source location):



Some moved text. Some text. ~~Some moved text.~~

This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

The moveFromRangeStart element specifies the start of the move source container within which all moved content is part of the move named `move1`. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

colFirst (First Table Column Covered By Bookmark)	Specifies the zero-based index of the first column in this row which shall be part of this bookmark. When a bookmark is contained within a table, it is possible for that bookmark to only

Attributes	Description
	cover cells within a certain column and row range within that table, by specifying:

- The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row.
- The first column included in the bookmark for each of the specified row(s) via this attribute.
- The last column included in the bookmark for each of the specified row(s) via the colLast attribute.
- The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row.

If this attribute appears, then the colLast attribute shall also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value exceeds the value of colLast or the number of columns in the table, then both values should be ignored.

[*Example*: Consider a three row by three column table where a table bookmark must be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):

■	■	□
■	■	□
□	□	□

This bookmark would be specified using the following WordprocessingML for the table's conents:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0"
w:name="table"/>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
```

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	`    <w:tc>`
	`      <w:p/>`
	`    </w:tc>`
	`    <w:tc>`
	`      <w:p/>`
	`    </w:tc>`
	`    <w:tc>`
	`      <w:p/>`
	`    </w:tc>`
	`    <w:bookmarkEnd w:id="0" />`
	`  </w:tr>`
	`  <w:tr>`
	`    <w:tc>`
	`      <w:p/>`
	`    </w:tc>`
	`    <w:tc>`
	`      <w:p/>`
	`    </w:tc>`
	`    <w:tc>`
	`      <w:p/>`
	`    </w:tc>`
	`  </w:tr>`
	`</w:tbl>`
	
	The colFirst attribute specifies that all columns starting with the first column must be included in the table bookmark. This applies starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

colLast (Last Table Column Covered By Bookmark)	Specifies the zero-based index of the last column in this row which shall be part of this bookmark. When a bookmark is contained within a table, it is possible for that bookmark to only cover cells within a certain column and row range within that table, by specifying: • The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row. • The first column included in the bookmark for each of the specified row(s) via the colFirst attribute. • The last column included in the bookmark for each of the specified row(s) via this attribute. • The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row.

Attributes	Description
	If this attribute appears, then the colFirst attribute shall also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value does not equal or exceed the value of colFirst or the number of columns in the table, then both values should be ignored. [*Example*: Consider a three row by three column table where a table bookmark shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):

■	■	
■	■	

This bookmark would be specified using the following WordprocessingML for the table's conents:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0"
w:name="table"/>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:bookmarkEnd w:id="0" />
  </w:tr>
  <w:tr>
    <w:tc>
```

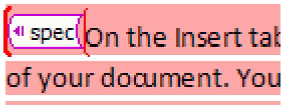
Attributes	Description

	`      <w:p/>`
	`    </w:tc>`
	`    <w:tc>`
	`      <w:p/>`
	`    </w:tc>`
	`    <w:tc>`
	`      <w:p/>`
	`    </w:tc>`
	`  </w:tr>`
	`</w:tbl>`
	
	The colLast attribute specifies that the last column that shall be included in the table bookmark is the second column. This applies starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element can differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example).

Attributes	Description

	If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored.
	[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:
	
	spec On the Insert tab
	of your document. You
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which must be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	`…`
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`  …`
	`</w:p>`
	`…`
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and is moved inside the paragraph to be represented physically, the comment with ID 0 must be inside the custom XML, but the comment with ID 1 must be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§17.18.13).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`

Attributes	Description
	`</…>` The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

name (Bookmark Name)	Specifies the bookmark name.
	
	If multiple bookmarks in a document share the same name, then the first bookmark (defined by the location of the bookmarkStart element in document order) shall be maintained, and all subsequent bookmarks should be ignored.
	
	[*Example*: Consider the following XML for a bookmark around a single word:
	
	`<w:p>`
	`  <w:bookmarkStart w:id="0" w:name="place" />`
	`  <w:r>`
	`    <w:t>Seattle</w:t>`
	`  </w:r>`
	`  <w:bookmarkEnd w:id="0" />`
	`</w:p>`
	
	The name attribute specifies that the name for this bookmark is `place`. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MoveBookmark) is located in §A.1. *end note*]

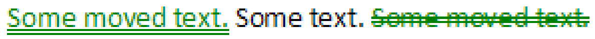
17.13.5.25 moveTo (Move Destination Run Content)

This element specifies that the inline-level content contained within it shall be treated as content which has been moved to this location and tracked as a revision.

The following restrictions shall be applied to this content:

- If this element occurs outside of a move destination container (§17.13.5.28; §17.13.5.27) for which a matching move source container (§17.13.5.24; §17.13.5.23) exists in the document, then the document is non-conformant.

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this image, green underline indicates the move destination and the green strikethrough indicates the move source location):



Some moved text. Some text. ~~Some moved text.~~

This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

The moveTo element specifies that all of the inline-level content contained within must be revision marked as content which was moved to its current location in the document. Because this moved content is contained within a complete move destination container (moveToRangeStart and moveToRangeEnd) with a corresponding move source, this content is tracked as a move. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`

Attributes	Description
	`</…>` The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is `1`. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_RunTrackChange) is located in §A.1. *end note*]

17.13.5.26 moveTo (Move Destination Paragraph)

This element specifies that the parent paragraph has been moved to this location and tracked as a revision. This does not imply anything about the revision state of the contents of the paragraph, and applies only to the existence of the paragraph as its own unique paragraph.

The following restrictions shall be applied to this content:

- If this element occurs outside of a move destination container (§17.13.5.28; §17.13.5.27) for which a matching move source container (§17.13.5.24; §17.13.5.23) exists in the document, then the document is non-conformant.

[*Example*: Consider a WordprocessingML document in which a paragraph of text is moved down in the document. This moved paragraph would be represented using the following WordprocessingML markup:

```
<w:moveToRangeStart w:id="0" w:name="aMove"/>
<w:p>
  <w:pPr>
    <w:rPr>
      <w:moveTo w:id="1" … />
    </w:rPr>
  </w:pPr>
  …
</w:p>
</w:moveToRangeEnd w:id="0"/>
```

The moveTo element as a child of the run properties of the paragraph mark specify that this paragraph mark was part of the content which was moved in the document. This implies nothing about the contents, since they might have been added later and not tracked as a revision (they must be marked as a move using the moveTo element (§17.13.5.25) around the run content). *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_TrackChange) is located in §A.1. *end note*]

17.13.5.27 moveToRangeEnd (Move Destination Location Container - End)

This element specifies the end of a region whose move destination contents are part of a single named move. When a move source is stored as a revision in a WordprocessingML document, two pieces of information shall be stored about that move destination:

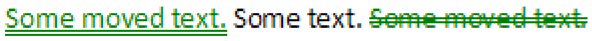
- A set of pieces of content which were moved - both inline-level content (§17.13.5.25) and paragraphs (§17.13.5.26)
- A move destination container (or "bookmark") which specifies that all content within it which marked as a move destination is part of a single named move. The name attribute on the move container links a group of move destination content with the corresponding group of move source content.

This element defines the end of the latter piece of the move revision data - the container. The id attribute on this element shall be used to link this element with the corresponding start of a move destination container in the document.

The following restrictions are applied to the use of this element:

- If this element occurs without a corresponding moveToRangeStart element (§17.13.5.28) with a matching id attribute value, then the document is non-conformant.
- If this element and its paired end occur without a matching move source container (§17.13.5.24; §17.13.5.23), then the document is non-conformant.
- If multiple move destination containers surround the same text, the document is non-conformant.

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this image, green underline indicates the move destination and the green strikethrough indicates the move source location):



Some moved text. Some text. ~~Some moved text.~~

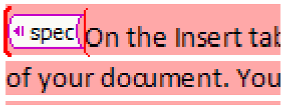
This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

The moveToRangeEnd element specifies the end of the move destination container within which all moved content is part of the move named `move1`. *end example*]

Attributes	Description
displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element can differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example). If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored. [*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:

Attributes	Description

	
	spec On the Insert tab
	of your document. You
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which must be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`…`
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`  …`
	`</w:p>`
	`…`
	
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and is moved inside the paragraph to be represented physically, the comment with ID 0 must be inside the custom XML, but the comment with ID 1 must be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§17.18.13).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

©ISO/IEC 2012 – All rights reserved

[*Note*: The W3C XML Schema definition of this element's content model (CT_MarkupRange) is located in §A.1. *end note*]

17.13.5.28 moveToRangeStart (Move Destination Location Container - Start)

This element specifies the start of the region whose move destination contents are part of a single named move. When a move destination is stored as a revision in a WordprocessingML document, two pieces of information shall be stored about that move destination:

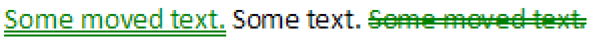
- A set of pieces of content which were moved - both inline-level content (§17.13.5.25) and paragraphs (§17.13.5.26)
- A move destination container (or "bookmark") which specifies that all content within it which marked as a move destination is part of a single named move. The name attribute on the move container links a group of move destination content with the corresponding group of move source content.

This element defines the start of the latter piece of the move revision data - the container. The id attribute on this element shall be used to link this element with the corresponding end of a move destination container in the document.

The following restrictions are applied to the use of this element

- If this element occurs without a corresponding moveToRangeEnd element (§17.13.5.27) with a matching id attribute value, then the document is non-conformant.
- If this element and its paired end occur without a matching move source container (§17.13.5.24; §17.13.5.23), then the document is non-conformant.
- If multiple start elements exist with the same id attribute value, then the document is non-conformant.
- If multiple move destination containers surround the same text, the document is non-conformant.

[*Example*: Consider a WordprocessingML document in which the first paragraph contains two sentences, and the first sentence is moved before the second sentence, and this move is tracked as a revision, as follows (in this image, green underline indicates the move destination and the green strikethrough indicates the move source location):



Some moved text. Some text. ~~Some moved text.~~

This document has the sentence `Some moved text.` moved to the first sentence in the document. This revision is represented using the following WordprocessingML:

```
<w:p>
  <w:moveToRangeStart w:id="0" … w:name="move1" />
  <w:moveTo w:id="1" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveTo>
  <w:moveToRangeEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve">Some text.</w:t>
  </w:r>
  <w:moveFromRangeStart w:id="2" … w:name="move1" />
  <w:moveFrom w:id="3" … >
    <w:r>
      <w:t>Some moved text.</w:t>
    </w:r>
  </w:moveFrom>
  <w:moveFromRangeEnd w:id="2" />
</w:p>
```

The moveToRangeStart element specifies the start of the move destination container within which all moved content is part of the move named `move1`. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

colFirst (First Table Column Covered By Bookmark)	Specifies the zero-based index of the first column in this row which shall be part of this bookmark. When a bookmark is contained within a table, it is possible for that bookmark to only

©ISO/IEC 2012 – All rights reserved

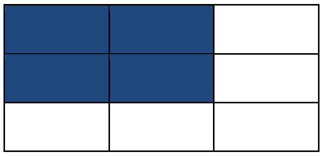
<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

cover cells within a certain column and row range within that table, by specifying:

- The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row.
- The first column included in the bookmark for each of the specified row(s) via this attribute.
- The last column included in the bookmark for each of the specified row(s) via the colLast attribute.
- The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row.

If this attribute appears, then the colLast attribute shall also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value exceeds the value of colLast or the number of columns in the table, then both values should be ignored.

[*Example*: Consider a three row by three column table where a table bookmark must be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):



This bookmark would be specified using the following WordprocessingML for the table's conents:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0"
w:name="table"/>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
```

</td></tr>
</table>

Attributes	Description

	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:bookmarkEnd w:id="0" />`
	`    </w:tr>`
	`    <w:tr>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`    </w:tr>`
	`  </w:tbl>`
	
	The colFirst attribute specifies that all columns starting with the first column must be included in the table bookmark. This applies starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

colLast (Last Table Column Covered By Bookmark)	Specifies the zero-based index of the last column in this row which shall be part of this bookmark. When a bookmark is contained within a table, it is possible for that bookmark to only cover cells within a certain column and row range within that table, by specifying: • The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row. • The first column included in the bookmark for each of the specified row(s) via the colFirst attribute. • The last column included in the bookmark for each of the specified row(s) via this attribute. • The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row.

Attributes	Description
	If this attribute appears, then the colFirst attribute shall also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value does not equal or exceed the value of colFirst or the number of columns in the table, then both values should be ignored. [*Example*: Consider a three row by three column table where a table bookmark shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):

■	■	
■	■	

This bookmark would be specified using the following WordprocessingML for the table's conents:

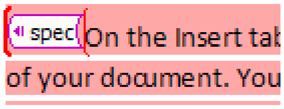
```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0"
w:name="table"/>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:bookmarkEnd w:id="0" />
  </w:tr>
  <w:tr>
    <w:tc>
```

Attributes	Description

	`<w:p/>`
	`</w:tc>`
	`<w:tc>`
	`<w:p/>`
	`</w:tc>`
	`<w:tc>`
	`<w:p/>`
	`</w:tc>`
	`</w:tr>`
	`</w:tbl>`
	
	The colLast attribute specifies that the last column that shall be included in the table bookmark is the second column. This applies starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element can differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example).

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<p>If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored.</p>
<p>[<i>Example</i>: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:</p>

<p>spec On the Insert tab
of your document. You</p>
<p>Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which must be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:</p>
<pre>
…
<w:commentRangeStart w:id="0" />
<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />
<w:customXml w:element="spec" … />
<w:p>
  …
</w:p>
…
</pre>
<p>The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and is moved inside the paragraph to be represented physically, the comment with ID 0 must be inside the custom XML, but the comment with ID 1 must be displaced to stay outside of the relative location of the next custom XML element (the spec element). <i>end example</i>]</p>
<p>The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§17.18.13).</p>
</td></tr>
<tr><td>id (Annotation Identifier)</td><td>
<p>Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.</p>
<p>If this attribute is omitted, then the document is non-conformant.</p>
<p>[<i>Example</i>: Consider an annotation represented using the following WordprocessingML fragment:</p>
<pre>
<… w:id="1" … >
  …
</pre>
</td></tr>
</table>

Attributes	Description
	`</…>` The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

name (Bookmark Name)	Specifies the bookmark name.
	If multiple bookmarks in a document share the same name, then the first bookmark (defined by the location of the bookmarkStart element in document order) shall be maintained, and all subsequent bookmarks should be ignored.
	[*Example*: Consider the following XML for a bookmark around a single word:
	`<w:p>`
	`  <w:bookmarkStart w:id="0" w:name="place" />`
	`  <w:r>`
	`    <w:t>Seattle</w:t>`
	`  </w:r>`
	`  <w:bookmarkEnd w:id="0" />`
	`</w:p>`
	The name attribute specifies that the name for this bookmark is `place`. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MoveBookmark) is located in §A.1. *end note*]

17.13.5.29 pPrChange (Revision Information for Paragraph Properties)

This element specifies the details about a single revision to a set of paragraph properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of paragraph properties which were applied to this paragraph before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of paragraph properties).

[*Example*: Consider a paragraph in a WordprocessingML document which is centered, and this change in the paragraph properties is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:pPr>
  <w:jc w:val="center"/>
  <w:pPrChange w:id="0" w:date="01-01-2006T12:00:00" w:author="John Doe">
    <w:pPr/>
  </w:pPrChange>
</w:pPr>
```

The pPrChange element specifies that there was a revision to the paragraph properties at `01-01-2006` by `John Doe`, and the previous set of paragraph properties on the paragraph were the null set (i.e. no paragraph properties explicitly present under the pPr element). *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`

Attributes	Description
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*] The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_PPrChange) is located in §A.1. *end note*]

17.13.5.30 rPrChange (Revision Information for Run Properties on the Paragraph Mark)

This element specifies the details about a single revision to a set of run properties applied to a paragraph mark within a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of run properties which were applied to this paragraph mark before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of run properties).

[*Example*: Consider an italicized paragraph mark in a WordprocessingML document which is also made bold and the latter change in the run properties is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:rPr>
  <w:b/>
  <w:i/>
  <w:rPrChange w:id="0" w:date="01-01-2006T12:00:00" w:author="John Doe">
    <w:rPr>
      <w:i/>
    </w:rPr>
  </w:rPrChange>
</w:rPr>
```

The rPrChange element specifies that there was a revision to the paragraph mark's run properties at 01-01-2006 by John Doe, and the previous set of run properties was simply the italicization using the i element (§17.3.2.16). *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is Example Author, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at

Attributes	Description
	10:00 AM, which can be used as desired. *end example*] The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ParaRPrChange) is located in §A.1. *end note*]

17.13.5.31 rPrChange (Revision Information for Run Properties)

This element specifies the details about a single revision to a set of run properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of run properties which were applied to this run before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of run properties).

[*Example*: Consider an italicized run in a WordprocessingML document which is also made bold and the latter change in the run properties is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
  <w:rPr>
    <w:b/>
    <w:i/>
    <w:rPrChange w:id="0" w:date="01-01-2006T12:00:00" w:author="John Doe">
      <w:rPr>
        <w:i/>
      </w:rPr>
    </w:rPrChange>
  </w:rPr>
```

The rPrChange element specifies that there was a revision to the run properties at `01-01-2006` by `John Doe`, and the previous set of run properties was simply the italicization using the i element (§17.3.2.16). *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_RPrChange) is located in §A.1. *end note*]

17.13.5.32 sectPrChange (Revision Information for Section Properties)

This element specifies the details about a single revision to a set of section properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of section properties which were applied to the parent section before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of section properties).

[*Example*: Consider a section in a WordprocessingML document which is set to be divided into three columns, and this change in the section properties is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:sectPr>
  <w:cols w:num="3" … >
    …
  </w:cols>
  <w:sectPrChange w:id="0" w:date="01-01-2006T12:00:00" w:author="John Doe">
    <w:sectPr/>
  </w:sectPrChange>
</w:sectPr>
```

The sectPrChange element specifies that there was a revision to the section properties at `01-01-2006` by `John Doe`, and the previous set of properties on the section were the null set (i.e. no section properties explicitly present under the sectPr element). *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SectPrChange) is located in §A.1. *end note*]

17.13.5.33 tblGridChange (Revision Information for Table Grid Column Definitions)

This element specifies the details about a single revision to a table's grid column definitions within a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the definition of the table grid which was applied to the parent table before this revision

[*Example*: Consider a two column table in a WordprocessingML document which has the width of its first column significantly reduced, and this change in the table grid is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:tblGrid>
  <w:gridCol w:w="1548" />
  <w:gridCol w:w="8028" />
  <w:tblGridChange w:id="1">
    <w:tblGrid>
      <w:gridCol w:w="4788" />
      <w:gridCol w:w="4788" />
    </w:tblGrid>
  </w:tblGridChange>
</w:tblGrid>
```

The tblGridChange element specifies that there was a revision to the table grid, and the previous table grid had both columns with a width of 4788 twentieths of a point, vs. their current widths of 1548 and 8028 twentieths of a point respectively. *end example*]

Attributes	Description

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblGridChange) is located in §A.1. *end note*]

17.13.5.34 tblPrChange (Revision Information for Table Properties)

This element specifies the details about a single revision to a set of table properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of table properties which were applied to the parent table before this revision

- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of table properties).

[*Example*: Consider a table in a WordprocessingML document which has the associated table style changed from `LightList` to `LightShading`, and this change in the table properties is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:tblPr>
  <w:tblStyle w:val="LightShading"/>
  <w:tblW w:w="0" w:type="auto"/>
  <w:tblLook w:firstRow="true" w:firstColumn="true"
      w:noVBand="true" />
  <w:tblPrChange w:id="0" w:author="Tristan Davis" w:date="2006-06-
01T13:39:00Z">
    <w:tblPr>
      <w:tblStyle w:val="LightList"/>
      <w:tblW w:w="0" w:type="auto"/>
      <w:tblLook w:firstRow="true" w:firstColumn="true"
      w:noVBand="true" />
    </w:tblPr>
  </w:tblPrChange>
</w:tblPr>
```

The tblPrChange element specifies that there was a revision to the table properties at `2006-06-01` by `Tristan Davis`, and the previous set of properties on the table was the set specifies in the child tblPr element (including the table style of `LightList`). *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

Attributes	Description

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblPrChange) is located in §A.1. *end note*]

17.13.5.35 tblPrExChange (Revision Information for Table-Level Property Exceptions)

This element specifies the details about a single revision to a set of table-level property exceptions in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of table-level property exceptions which were applied to the parent table row before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of table-level exception properties).

[*Example*: Consider a set of table rows which are part of a table in a WordprocessingML document, have table-level property exceptions, and this change in the table-level properties to a fixed table width of ten inches is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:tblPrEx>
  <w:tblW w:w="14400" w:type="dxa"/>
  <w:tblPrExChange w:id="0" w:author="Tristan Davis" w:date="2006-06-
01T13:39:00Z">
    <w:tblPrEx>
      <w:tblW w:w="0" w:type="auto"/>
    </w:tblPrEx>
  </w:tblPrExChange>
</w:tblPrEx>
```

The tblPrExChange element specifies that there was a revision to the table-level property exceptions at `2006-06-01` by `Tristan Davis`, and the previous set of table-level property exceptions set specifies in the child tblPrEx element. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500. If this attribute is omitted, then no date information shall be associated with the parent

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TblPrExChange) is located in §A.1. *end note*]

17.13.5.36 tcPrChange (Revision Information for Table Cell Properties)

This element specifies the details about a single revision to a set of table cell properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of table cell properties which were applied to the parent table before this revision

- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of table cell properties).

[*Example*: Consider a table cell in a WordprocessingML document which has a change in the table cell properties that is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:tcPr>
  <w:cnfStyle w:firstColumn="true" />
  <w:tcW w:w="3192" w:type="dxa"/>
  <w:tcPrChange w:id="8" w:author="Tristan Davis" w:date="2006-06-01T13:39:00Z">
    <w:tcPr>
      <w:tcW w:w="3192" w:type="dxa"/>
    </w:tcPr>
  </w:tcPrChange>
</w:tcPr>
```

The tcPrChange element specifies that there was a revision to the table cell properties at `2006-06-01` by `Tristan Davis`, and the previous set of properties on the table cell was the set specifies in the child tcPr element. *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`  …`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500. If this attribute is omitted, then no date information shall be associated with the parent annotation type. [*Example*: Consider a comment represented using the following WordprocessingML fragment:

Attributes	Description

	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`  …`
	`</…>`
	
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*]
	
	The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	
	If this attribute is omitted, then the document is non-conformant.
	
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	
	`<… w:id="1" … >`
	`  …`
	`</…>`
	
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_TcPrChange) is located in §A.1. *end note*]

17.13.5.37 trPrChange (Revision Information for Table Row Properties)

This element specifies the details about a single revision to a set of table row properties in a WordprocessingML document.

This element stores this revision as follows:

- The child element of this element contains the complete set of table row properties which were applied to the parent table row before this revision
- The attributes of this element contain information about when this revision took place (i.e. when these properties became a 'former' set of table row properties).

[*Example*: Consider a table cell in a WordprocessingML document which has a change in the table row properties that is tracked as a revision. This revision would be specified using the following WordprocessingML markup:

```
<w:trPr>
  <w:cantSplit/>
  <w:trPrChange w:id="8" w:author="Tristan Davis" w:date="2006-06-01T13:39:00Z">
    <w:trPr/>
  </w:trPrChange>
</w:trPr>
```

The trPrChange element specifies that there was a revision to the table row properties at `2006-06-01` by `Tristan Davis`, and the previous set of properties on the table row was the set specified in the child trPr element (in this case, the null set). *end example*]

Attributes	Description

author (Annotation Author)	Specifies the author for an annotation within a WordprocessingML document.
	If this attribute is omitted, then no author shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:author="Example Author">`
	`…`
	`</…>`
	The author attribute specifies that the author of the current annotation is `Example Author`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

date (Annotation Date)	Specifies the date information for an annotation within a WordprocessingML document. The use of this information is outside of the scope of ISO/IEC 29500.
	If this attribute is omitted, then no date information shall be associated with the parent annotation type.
	[*Example*: Consider a comment represented using the following WordprocessingML fragment:
	`<… w:id="1" w:date="2006-01-01T10:00:00">`
	`…`
	`</…>`

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	The date attribute specifies that the date of the current annotation is January 1st 2006 at 10:00 AM, which can be used as desired. *end example*] The possible values for this attribute are defined by the ST_DateTime simple type (§17.18.9).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TrPrChange) is located in §A.1. *end note*]

17.13.6 Bookmarks

Within a WordprocessingML document, *bookmarks* refer to arbitrary regions of content which are bounded and have a unique name associated with them.

Because bookmarks are a legacy word processing function which predates the concepts of XML and well-formedness, they can start and end at any location within a document's contents and therefore shall use the "cross-structure" annotation format described in §17.13.2.

[*Example*: Consider the following WordprocessingML markup for two paragraphs, each reading `Example Text`, where a bookmark has been added spanning the second word in paragraph one and the first word in paragraph two:

```
<w:p>
  <w:r>
    <w:t>Example</w:t>
  </w:r>
  <w:bookmarkStart w:id="0" w:name="sampleBookmark" />
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t>Example</w:t>
  </w:r>
  <w:bookmarkEnd w:id="0" />
  <w:r>
    <w:t xml:space="preserve"> text.</w:t>
  </w:r>
</w:p>
```

The bookmarkStart and bookmarkEnd elements (§17.13.6.2; §17.13.6.1) specify the location where the bookmark starts and ends, but cannot contain it using a single tag because it spans part of two paragraphs. However, the two tags are part of one group because the id attribute value specifies `0` for both. *end example*]

17.13.6.1 bookmarkEnd (Bookmark End)

This element specifies the end of a bookmark within a WordprocessingML document. This end marker is matched with the appropriately paired start marker by matching the value of the id attribute from the associated bookmarkStart element.

If no bookmarkStart element exists prior to this element in document order with a matching id attribute value, then the document is non-conformant.

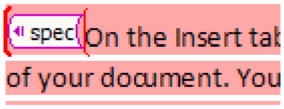
[*Example*: Consider a document with a bookmark which spans half of paragraph one, and part of paragraph two. The following WordprocessingML illustrates an example of content which fufills this constraint:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is sentence one.</w:t>
  </w:r>
  <w:bookmarkStart w:id="0" w:name="testing123"/>
  <w:r>
    <w:t>This is sentence two.</w:t>
  </w:r>
</w:p>
```

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This </w:t>
  </w:r>
  <w:bookmarkEnd w:id="0"/>
  <w:r>
    <w:t>is sentence three.</w:t>
  </w:r>
</w:p>
```

The bookmarkEnd element specifies the end of the region for the bookmark whose bookmarkStart element has an id attribute value of `0`. In this case, this refers to the `testing123` bookmark. *end example*]

Attributes	Description

displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element can differ.
	
	Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example).
	
	If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored.
	
	[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:
	
	
	spec On the Insert tal
	of your document. You
	
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which must be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	
	`…`
	`<w:commentRangeStart w:id="0" />`

Attributes	Description

	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`  …`
	`</w:p>`
	`…`
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and is moved inside the paragraph to be represented physically, the comment with ID 0 must be inside the custom XML, but the comment with ID 1 must be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§17.18.13).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MarkupRange) is located in §A.1. *end note*]

17.13.6.2 bookmarkStart (Bookmark Start)

This element specifies the start of a bookmark within a WordprocessingML document. This start marker is matched with the appropriately paired end marker by matching the value of the id attribute from the associated bookmarkEnd element.

If no bookmarkEnd element exists subsequent to this element in document order with a matching id attribute value, then the document is non-conformant.

If a bookmark begins and ends within a single table, it is possible for that bookmark to cover discontiguous parts of that table which are logically related (e.g. a single column in a table). This kind of placement for a bookmark is accomplished (and described in detail) on the colFirst and colLast attributes on this element.

[*Example*: Consider a document with a bookmark which spans half of paragraph one, and part of paragraph two. The following WordprocessingML illustrates an example of content which fufills this constraint:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is sentence one.</w:t>
  </w:r>
  <w:bookmarkStart w:id="0" w:name="testing123"/>
  <w:r>
    <w:t>This is sentence two.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t xml:space="preserve">This </w:t>
  </w:r>
  <w:bookmarkEnd w:id="0"/>
  <w:r>
    <w:t>is sentence three.</w:t>
  </w:r>
</w:p>
```

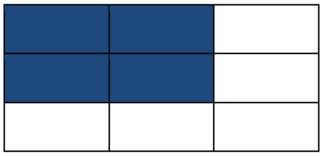
The bookmarkStart element specifies the start of the region for the `testing123` bookmark. This element is then linked to the bookmarkEnd element which also has an id attribute value of `0`. *end example*]

Attributes	Description
colFirst (First Table Column Covered By Bookmark)	Specifies the zero-based index of the first column in this row which shall be part of this bookmark. When a bookmark is contained within a table, it is possible for that bookmark to only cover cells within a certain column and row range within that table, by specifying: • The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row. • The first column included in the bookmark for each of the specified row(s) via this attribute. • The last column included in the bookmark for each of the specified row(s) via the colLast attribute. • The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

If this attribute appears, then the colLast attribute shall also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value exceeds the value of colLast or the number of columns in the table, then both values should be ignored.

[*Example*: Consider a three row by three column table where a table bookmark must be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):



This bookmark would be specified using the following WordprocessingML for the table's conents:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0"
w:name="table"/>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:bookmarkEnd w:id="0" />
  </w:tr>
  <w:tr>
```

</td></tr>
</table>

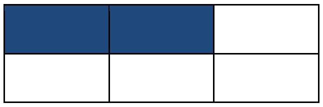
©ISO/IEC 2012 – All rights reserved

Attributes	Description

	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`    </w:tr>`
	`  </w:tbl>`
	
	The colFirst attribute specifies that all columns starting with the first column must be included in the table bookmark. This applies starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

colLast (Last Table Column Covered By Bookmark)	Specifies the zero-based index of the last column in this row which shall be part of this bookmark. When a bookmark is contained within a table, it is possible for that bookmark to only cover cells within a certain column and row range within that table, by specifying: • The first row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkStart element in the first table cell in that row. • The first column included in the bookmark for each of the specified row(s) via the colFirst attribute. • The last column included in the bookmark for each of the specified row(s) via this attribute. • The last row for which the specified columns are part of the table bookmark. This is accomplished by placing the bookmarkEnd element at the end of that table row. If this attribute appears, then the colFirst attribute shall also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value does not equal or exceed the value of colFirst or the number of columns in the table, then both values should be ignored. [*Example*: Consider a three row by three column table where a table bookmark shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below): ▨ ▨ □

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>



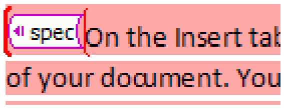
This bookmark would be specified using the following WordprocessingML for the table's conents:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:bookMarkStart w:colFirst="0" w:colLast="1" w:id="0"
w:name="table"/>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:bookmarkEnd w:id="0" />
  </w:tr>
  <w:tr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
</w:tbl>
```

The colLast attribute specifies that the last column that shall be included in the table

</td></tr>
</table>

Attributes	Description
	bookmark is the second column. This applies starting with the first row and ending with the second row (the two rows within the bookmark's start and end). *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

displacedByCustomXml (Annotation Marker Relocated For Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element can differ.
	Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example).
	If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored.
	[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:
	
	spec On the Insert tab
	of your document. You
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which must be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	`…`
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`  …`
	`</w:p>`
	`…`
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and is moved inside the paragraph to be represented

Attributes	Description
	physically, the comment with ID 0 must be inside the custom XML, but the comment with ID 1 must be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*] The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§17.18.13).

id (Annotation Identifier)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

name (Bookmark Name)	Specifies the bookmark name.
	If multiple bookmarks in a document share the same name, then the first bookmark (defined by the location of the bookmarkStart element in document order) shall be maintained, and all subsequent bookmarks should be ignored.
	[*Example*: Consider the following XML for a bookmark around a single word:
	`<w:p>`
	`  <w:bookmarkStart w:id="0" w:name="place" />`
	`  <w:r>`
	`    <w:t>Seattle</w:t>`
	`  </w:r>`
	`  <w:bookmarkEnd w:id="0" />`
	`</w:p>`
	The name attribute specifies that the name for this bookmark is place. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Bookmark) is located in §A.1. *end note*]

17.13.7 Range Permissions

Range permissions in a WordprocessingML document refer to a special kind of bookmark used to control which subset(s) of users can edit a particular region of a document. Range permissions specify the user or set of users which are allowed to edit all content between them whenever the document protection specified by the documentProtection element (§17.15.1.29) is enabled and set to `readOnly` or `comments`.

Like bookmarks, range permissions are a legacy word processing function which predates the concepts of XML and well-formedness, so they can start and end at any location within a document's contents and therefore shall use the "cross-structure" annotation format described in §17.13.2.

[*Example*: Consider the following WordprocessingML markup for a single paragraph, where a range permission has been added spanning the words `range permission`:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is a </w:t>
  </w:r>
  <w:permStart w:id="0" w:edGrp="everyone"/>
  <w:r>
    <w:t>range permission</w:t>
  </w:r>
  <w:permEnd w:id="0"/>
  <w:r>
    <w:t>.</w:t>
  </w:r>
</w:p>
```

The permStart and permEnd elements (§17.13.7.1; §17.13.7.2) specify the location where the range permission starts and ends. The two tags are part of one group because the id attribute value specifies `0` for both.

If document protection was enabled, then no content in this document must be editable except for this range permission, which is editable by all users that open the document (specified using an editor group of everyone). *end example*]

17.13.7.1 permEnd (Range Permission End)

This element specifies the end of a single range permission within a WordprocessingML document. This end marker is matched with the appropriately paired start marker by matching the value of the id attribute from the associated permStart element.

If no permStart element exists prior to this element in document order with a matching id attribute value, then the document is non-conformant.

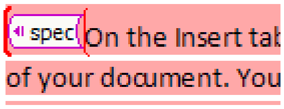
[*Example*: Consider a document with a range permission which spans half of paragraph one, and part of paragraph two. The following WordprocessingML illustrates an example of content which fufills this constraint:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is sentence one.</w:t>
  </w:r>
  <w:permStart w:id="0" w:edGrp="everyone"/>
  <w:r>
    <w:t>This is sentence two.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t xml:space="preserve">This </w:t>
  </w:r>
  <w:permEnd w:id="0"/>
  <w:r>
    <w:t>is sentence three.</w:t>
  </w:r>
</w:p>
```

The permEnd element specifies the end of the region for the range permission whose permStart element has an id attribute value of `0`. *end example*]

Attributes	Description
displacedByCustomXml (Annotation Displaced By Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element can differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example). If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored. [*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical

Attributes	Description

	representation), as follows:
	
	spec On the Insert ta
	of your document. You
	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which must be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	`…`
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`  …`
	`</w:p>`
	`…`
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and is moved inside the paragraph to be represented physically, the comment with ID 0 must be inside the custom XML, but the comment with ID 1 must be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§17.18.13).

id (Annotation ID)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Perm) is located in §A.1. *end note*]

17.13.7.2 permStart (Range Permission Start)

This element specifies the start of a range permission within a WordprocessingML document. This start marker is matched with the appropriately paired end marker by matching the value of the id attribute from the associated permEnd element.

If no permEnd element exists subsequent to this element in document order with a matching id attribute value, then the document is non-conformant.

If a range permission begins and ends within a single table, it is possible for that permission to cover discontiguous parts of that table which are logically related (e.g. a single column in a table). This type of placement for a range permission is accomplished (and described in detail) on the colFirst and colLast attributes on this element.

[*Example*: Consider a document with a range permission which spans half of paragraph one, and part of paragraph two. The following WordprocessingML illustrates an example of content which fufills this constraint:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">This is sentence one.</w:t>
  </w:r>
  <w:permStart w:id="0" w:edGrp="everyone"/>
  <w:r>
    <w:t>This is sentence two.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:r>
    <w:t xml:space="preserve">This </w:t>
  </w:r>
  <w:permEnd w:id="0"/>
  <w:r>
    <w:t>is sentence three.</w:t>
  </w:r>
</w:p>
```

The permStart element specifies the start of the region for the range permission. This element is then linked to the permEnd element which also has an id attribute value of `0`. *end example*]

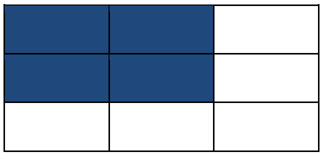
Attributes	Description
colFirst (First Table Column Covered By Range Permission)	Specifies the zero-based index of the first column in this row which shall be part of this range permission.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
When a range permission is contained within a table, it is possible for that range permission to only cover cells within a certain column and row range within that table, by specifying:

- The first row for which the specified columns are part of the table range permission. This is accomplished by placing the permStart element in the first table cell in that row.
- The first column included in the range permission for each of the specified row(s) via this attribute.
- The last column included in the range permission for each of the specified row(s) via the colLast attribute.
- The last row for which the specified columns are part of the table range permission. This is accomplished by placing the permEnd element at the end of that table row.

If this attribute appears, then the colLast attribute shall also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a range permission which is not contained in a table, then their values should be ignored. If this value exceeds the value of colLast or the number of columns in the table, then both values should be ignored.

[*Example*: Consider a three row by three column table where a table range permission shall be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):



This bookmark would be specified using the following WordprocessingML for the table's conents:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:permStart w:colFirst="0" w:colLast="1" w:id="0"
w:edGrp="everyone"/>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
```
</td></tr>
</table>

Attributes	Description

	`    </w:tr>`
	`    <w:tr>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:permEnd w:id="0" />`
	`    </w:tr>`
	`    <w:tr>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`      <w:tc>`
	`        <w:p/>`
	`      </w:tc>`
	`    </w:tr>`
	`  </w:tbl>`
	
	The colFirst attribute specifies that all columns starting with the first column must be included in the table range permission. This applies starting with the first row and ending with the second row (the two rows within the range permission's start and end). *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

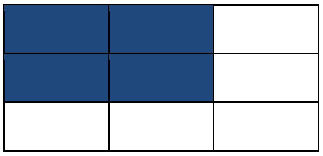
colLast (Last Table Column Covered By Range Permission)	Specifies the zero-based index of the last column in this row which shall be part of this range permission. When a range permission is contained within a table, it is possible for that range permission to only cover cells within a certain column and row range within that table, by specifying: • The first row for which the specified columns are part of the table range permission. This is accomplished by placing the permStart element in the first table cell in that row. • The first column included in the range permission for each of the specified row(s) via the colFirst attribute. • The last column included in the range permission for each of the specified row(s) via this attribute.

Attributes	Description

- The last row for which the specified columns are part of the table range permission. This is accomplished by placing the permEnd element at the end of that table row.

If this attribute appears, then the colFirst attribute shall also appear (regardless of where this bookmark is located) or the document shall be considered non-conformant. If this attibute and its pair occur on a bookmark which is not contained in a table, then their values should be ignored. If this value does not equal or exceed the value of colFirst or the number of columns in the table, then both values should be ignored.

[*Example*: Consider a three row by three column table where a table range permission must be applied to the contents of the first two cells in the first two rows in the table (the cells shaded below):

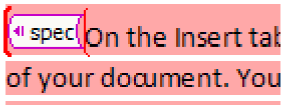


This bookmark would be specified using the following WordprocessingML for the table's conents:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      <w:permStart w:colFirst="0" w:colLast="1" w:id="0"
w:edGrp="everyone"/>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
  </w:tr>
  <w:tr>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
    <w:tc>
      <w:p/>
    </w:tc>
```

Attributes	Description

	`<w:permEnd w:id="0" />`
	`</w:tr>`
	`<w:tr>`
	`<w:tc>`
	`<w:p/>`
	`</w:tc>`
	`<w:tc>`
	`<w:p/>`
	`</w:tc>`
	`<w:tc>`
	`<w:p/>`
	`</w:tc>`
	`</w:tr>`
	`</w:tbl>`
	
	The colLast attribute specifies that the last column that must be included in the table range permission is the second column. This applies starting with the first row and ending with the second row (the two rows within the range permission's start and end). *end example*]
	
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

displacedByCustomXml (Annotation Displaced By Custom XML Markup)	Specifies that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document. This element only has an effect when the custom XML element is block-level (i.e. surrounds an entire paragraph), as in this scenario the logical and physical placement of the annotation and custom XML element can differ. Specifically, in this case, the custom XML is presented *around* the block-level object it encloses (the paragraph, table, table row, or table cell), but is physically represented within that same object (i.e. within the paragraph, table, table row or table cell). This requirement stems from the fact that there is no location for the location of the annotation within the document at its logical location (around a table, for example). If this element is omitted, then the annotation shall be anchored inside of all block-level custom XML elements in the paragraph. If this element is present, but no block-level custom XML tag is located at the position it specifies (before or after), then it shall be ignored. [*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:  spec On the Insert tab of your document. You

Attributes	Description

	Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which must be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:
	`…`
	`<w:commentRangeStart w:id="0" />`
	`<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />`
	`<w:customXml w:element="spec" … />`
	`<w:p>`
	`  …`
	`</w:p>`
	`…`
	The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and is moved inside the paragraph to be represented physically, the comment with ID 0 must be inside the custom XML, but the comment with ID 1 must be displaced to stay outside of the relative location of the next custom XML element (the `spec` element). *end example*]
	The possible values for this attribute are defined by the ST_DisplacedByCustomXml simple type (§17.18.13).

ed (Single User For Range Permission)	Specifies a single user for which this range permission shall be enabled (i.e. a user which shall be able to edit this range when document protection is enabled).
	This editor can be stored in one of the following forms:
	• `DOMAIN\username` - for users whose access shall be authenticated using the current user's domain credentials
	• `user@domain.com` - for users whose access shall be authenticated using the user's e-mail address as credentials
	• `user` - for users whose access shall be authenticated using the current user's machine credentials
	[*Example*: Consider a range permission defined as follows:
	`<w:permStart w:id="0" w:ed="example@example.com" … />`
	`…`
	`<w:permEnd w:id="0" />`
	The ed attribute value of `example@example.com` specifies that only user(s) who can authenticate with an application as associated with that e-mail address must be allowed to edit the contents between the start and end markers when document protection is being enforced. *end example*]
	The possible values for this attribute are defined by the ST_String simple type

Attributes	Description
	(§22.9.2.13).

edGrp (Editor Group For Range Permission)	Specifies an alias (or editing group) which shall be used to determine if the current user shall be allowed to edit this range of the document. This mechanism simply provides a set of predefined editing groups which can be associated with user accounts by applications in any desired manner.
	[*Example*: Consider a range permission defined as follows:
	`<w:permStart w:id="0" w:edGrp="editors" … />`
	`…`
	`<w:permEnd w:id="0" />`
	The edGrp attribute value of `editors` specifies that only user(s) who the current application associates with the editors group shall be allowed to edit the contents between the start and end markers when document protection is being enforced. *end example*]
	The possible values for this attribute are defined by the ST_EdGrp simple type (§17.18.21).

id (Annotation ID)	Specifies a unique identifier for an annotation within a WordprocessingML document. The restrictions on the id attribute, if any, are defined by the parent XML element.
	If this attribute is omitted, then the document is non-conformant.
	[*Example*: Consider an annotation represented using the following WordprocessingML fragment:
	`<… w:id="1" … >`
	`  …`
	`</…>`
	The id attribute specifies that the ID of the current annotation is 1. This value is used to uniquely identify this annotation within the document content. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PermStart) is located in §A.1. *end note*]

17.13.8 Spelling and Grammar

The remaining kind of annotation stored in a WordprocessingML document, *spelling and grammar errors* are annotations used to specify the locations of existing spelling and grammatical errors within the contents of a document.

[*Rationale*: When a WordprocessingML document is saved, applications can choose to save currently flagged spelling and grammar errors, for two reasons:

- In order to increase the performance subsequent loads of the document (as those load operations can rely on the persisted proofing state of the document)
- In order to store words which must not be marked as proofing errors regardless of how they would normally be flagged by the proofing tools engine (i.e. to store spelling and grammar exceptions).

end rationale]

[*Example*: Consider the following paragraph consisting of two misspelled words, where the second word has been explicitly flagged as not being a spelling error. This paragraph would consist of the following WordprocessingML markup:

```
<w:p>
  <w:proofErr w:type="spellStart"/>
  <w:r>
    <w:t>erqwt</w:t>
  </w:r>
  <w:proofErr w:type="spellEnd"/>
  <w:r>
    <w:t xml:space="preserve"> werewr</w:t>
  </w:r>
</w:p>
```

The proofErr elements, with a val attribute value of `spellStart` and `spellEnd` respectively, delimit the start and end the content in this paragraph which is stored as a spelling error. Since the second word is not included in that range, it is not stored as a spelling error. *end example*]

17.13.8.1 proofErr (Proofing Error Anchor)

This element specifies the presence of a start or end anchor for a single proofing error within a WordprocessingML document.

When proofing errors are stored in a document, their semantics shall be interpreted as follows:

- Each proofing error with a type attribute value of `spellStart` shall be linked with the next error with a type attribute of `spellEnd`. If one does not exist, then this error should be ignored.
- Each proofing error with a type attribute value of `spellEnd` which was not preceded by an error with a type attribute value of `spellStart` (that was not previously matched to an end) should be ignored.
- Each proofing error with a type attribute value of `gramStart` shall be linked with the next error with a type attribute of `gramEnd`. If one does not exist, then this error should be ignored.
- Each proofing error with a type attribute value of `gramEnd` which was not preceded by an error with a type attribute value of `gramStart` (that was not previously matched to an end) should be ignored.

[*Example*: Consider the following sentence with a grammatical error in its subject/verb agreement. If an application recognized this error and wished to persist it to the document, this paragraph would consist of the following WordprocessingML markup:

```
<w:p>
  <w:proofErr w:type="gramStart"/>
  <w:r>
    <w:t>This are</w:t>
  </w:r>
  <w:proofErr w:type="gramEnd"/>
  <w:r>
    <w:t xml:space="preserve"> an error.</w:t>
  </w:r>
</w:p>
```

The proofErr elements, with a val attribute value of gramStart and gramEnd respectively, delimit the start and end the content in this paragraph which is stored as a grammatical error. *end example*]

Attributes	Description

type (Proofing Error Anchor Type)	Specifies the type of proofing error anchor at this location in the document. This proofing error type implies the necessary semantics for this element as defined by the parent element.
	[*Example*: Consider the following sentence with a proofing error, consisting of the following WordprocessingML markup:
	`<w:r>`
	`  <w:t>are</w:t>`
	`</w:r>`
	`<w:proofErr w:val="gramEnd"/>`
	`<w:r>`
	`  <w:t xml:space="preserve"> an error.</w:t>`
	`</w:r>`
	The val attribute value of gramEnd specifies that the proofing error is the location of the end of content which is stored as a grammatical error. *end example*]
	The possible values for this attribute are defined by the ST_ProofErr simple type (§17.18.70).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_ProofErr) is located in §A.1. *end note*]

17.14 Mail Merge

Mail merge refers to an operation by which WordprocessingML documents can work in conjunction with data from an external data source, importing this data into a document according to a set of codes contained in WordprocessingML known as fields.

A WordprocessingML document that contains the mailMerge element (§17.14.20) and is therefore connected to an external data source, is known as a *source document*. In addition to being connected to an external data source and containing fields, a source document can contain any regular WordprocessingML constructs such as:

- Text runs
- Paragraphs
- Images
- Tables
- Numbering
- Etc.

There are two key parts of the mail merge data stored in a WordprocessingML document

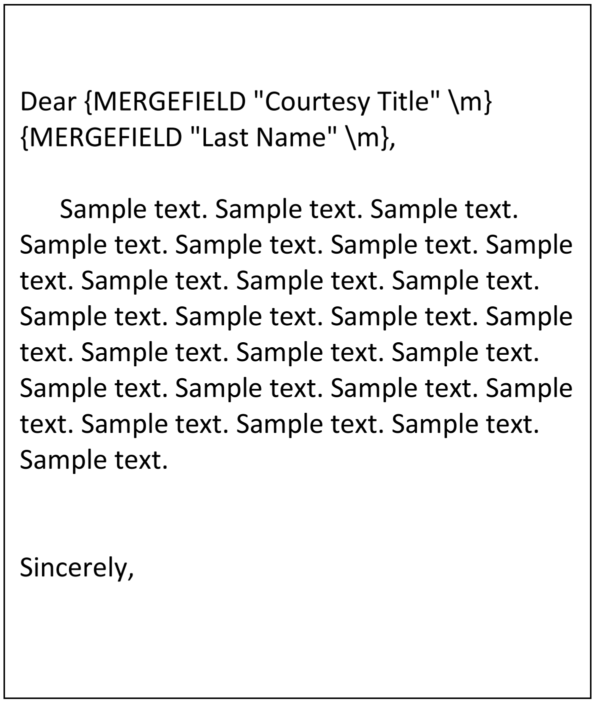
9. Information connecting a document to an external data source
10. Information populating fields within that document with external data.

Once the fields in a merged document have been populated with external data, mail merge has been completed and the resulting files are known as *mail merged documents* or simply *merged documents*.

The mail merge settings for a WordprocessingML document are stored in two locations:

- The standard mail merge settings are stored as the child elements of the mailMerge element (§17.14.20)
- A set of additional mail merge settings stored in the odso element (§17.14.25), and collectively referred to as the Office Data Source Object settings. The *Office Data Source Object* is an extension to the standard settings stored with a mail merge which performs two functions: First, it provides additional information about the mail merge data source, specifically: information about how to map the columns in the data source to MERGEFIELD fields and information about records which shall be included and excluded when creating merged documents. Second, it provides an alternate set of connection information which should be used when the dataType element (§17.14.10) specifies a value of native. This alternate connection string provides additional connection information for applications which choose to support the ODSO connection string syntax.

[*Example*: Consider a WordprocessingML document containing static WordprocessingML constructs such as text runs and paragraphs in addition to two WordprocessingML MERGEFIELD fields (§17.16.5.35) calling for Courtesy Title and Last Name data. The field codes for each field are displayed, delimited by {} characters:



Dear {MERGEFIELD "Courtesy Title" \m} {MERGEFIELD "Last Name" \m},

Sample text. Sample text.

Sincerely,

If the following WordprocessingML was added to this document, this document would become a *source document* rather than just a standard WordprocessingML document, as the mailMerge (§17.14.20) element specifies the elements and attributes necessary to enabled the document to connect to an external spreadsheet data source.

```
<w:mailMerge>
  …
  <w:dataType w:val="spreadsheet" />
  <w:query w:val="SELECT * FROM `Sheet1$`" />
  <w:dataSource r:id="rId1" />
  …
</w:mailMerge>
```

Here, the dataType (§17.14.10) and dataSource (§17.14.9) elements specify that the given document must be connected to the external data source target by the relationship whose relationship value is `rId1` as specified in the dataSource element (§17.14.9). While connected to the external data source, the *source document* together with the hosting application and/or data source access application extracts data from the external data source to perform the merge as specified by the connectString (§17.14.8) and query (§17.14.26) elements. *end example*]

17.14.1 active (Record Is Included in Mail Merge)

This element specifies whether a specific record from the specified external data source shall be imported into a merged WordprocessingML document when the mail merge defined for a source document is performed. If this element's val attribute is false, then the record specified by the parent element shall not used to create a merged document.

If this element is omitted for a given record, the data record associated with it shall be imported into a merged WordprocessingML document when the mail merge is performed.

[*Example*: Consider the following fragment from a source WordprocessingML document that is connected to an external data source containing two records, one of which is not imported into a merged WordprocessingML document when the conforming hosting application performs the data import.

```
<w:recipients>
  <w:recipientData>
    <w:active w:val="false" />
    …
    <w:uniqueTag>1126664175</w:uniqueTag>
  </w:recipientData>
  <w:recipientData>
    …
    <w:uniqueTag>1530576378</w:uniqueTag>
  </w:recipientData>
</w:recipients>
```

In this XML fragment, the external data record who is identified by the uniqueTag element (§17.14.35) with a val attribute equal to 1126664175 is not imported into a merged document as the active element associated with it has a val attribute equal to false. Conversely, the external data record associated with the uniqueTag element with a val attribute equal to 1530576378 is imported into a merged document, as its active element has been omitted (implying the default value of true). *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.14.2 activeRecord (Record Currently Displayed In Merged Document)

This element specifies that the hosting application shall display the given record from the specified external data source in place of the MERGEFIELD fields (§17.16.5.35) its data is mapped to via the fieldMapData element (§17.14.15) in a merged document. When this element is present, the val attribute shall specify the one-based index of the record from that data source which shall be used to populate this document.

If the activeRecord element is omitted with the viewMergedData element's val attribute equal to true, the hosting application shall behave as if the activeRecord element's val attribute was equal to 1. If the viewMergedData element (§17.14.36) is omitted or present with a val attribute equal to 0 or false, then this element shall be ignored. If the activeRecord record is given a val attribute that is less than one or greater than the number of records in the specified external data source, the hosting application shall treat this val attribute as if it were equal to 1.

[*Example*: Consider a merged WordprocessingML document containing two WordprocessingML fields calling for `Courtesy Title` and `Last Name` data and a sample text paragraph. Also, note that the external data source this merged document is connected to contains two records, both containing name and address information, with the first record pertaining to `Mr. Doe`, and the second pertaining to `Ms. Smith`.

This table below illustrates the necessary WordprocessingML to display applicable data from the specified external data source within the merged document where fields have been inserted:

`<w:viewMergedData val="0"/>`	`<w:viewMergedData val="1" />`
	`<w:activeRecord w:val="1" />`
		`<w:viewMergedData val="1" />`
		`<w:activeRecord w:val="2" />`

Dear {MERGEFIELD "Courtesy Title" \m} {MERGEFIELD "Last Name" \m}, Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,	Dear Mr. Doe: Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,	Dear Ms. Smith: Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,

end example]

[*Example*: Consider the following WordprocessingML from a merged WordprocessingML document:

```
<w:viewMergedData />
<w:activeRecord w:val="2" />
```

The activeRecord element is present and has a val attribute equal to 2, therefore this WordprocessingML specifies that a conforming hosting application must display data from the second record of the specified external data source in place of fields its data is mapped to within the merged document. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />`

Attributes	Description
	The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.14.3 addressFieldName (Column Containing E-mail Address)

This element specifies the column within a given external data source that contains e-mail addresses. This element is specified independently of the field mappings specified for a given merged document via the fieldMapData element (§17.14.15).

If this element is omitted, or no column exists in the data source with this column name, then the source document specifies that no e-mail address data shall be associated with this mail merge.

[*Note*: This element is generally used to allow the e-mailing of merged documents resulting from populating the fields within a merged document with external data.

This element is independent of the field mapping specified for a given merged document via the fieldMapData element (§17.14.15). This separation enables applications to email the documents resulting from the population of WordprocessingML fields with external data regardless of the presence or absence of a field mapped to external data specifying email addresses. *end note*]

[*Example*: Consider a merged WordprocessingML document that is connected to an external data source containing a column of data tilted `Alternate Email Addresses`. The following WordprocessingML would be included in the source and merged documents to specify which column in the external data source contains email addresses.

```
<w:addressFieldName w:val="Alternate Email Address" />
```

The addressFieldName element specifies that the `Alternate Email Address` column contains e-mail addresses for each record. *end example*]

Attributes	Description
val (String Value)	Specifies that its contents contain a string. The contents of this string are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment:

Attributes	Description

	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_String) is located in §A.1. *end note*]

17.14.4 checkErrors (Mail Merge Error Reporting Setting)

This element specifies the type of error reporting which shall be conducted by an application when performing a mail merge against the specified source data.

The type of error reporting implied by this element shall be defined as follows:

- Simulate the population of fields with mapped external data and report errors in a new document if the val attribute is equal to 1.
- While populating fields with mapped external data, pausing to report each error as it occurs if the val attribute is equal to 2.
- Populate fields with mapped external data and report errors in a new document if the val attribute is equal to 3.
- Application-defined behaviors can be used if the val attribute is equal to any other value.

If this element is omitted, or its value is set to a value outside of those specified below that is not understood by the hosting application, then its value shall be assumed to be 2.

[*Example*: Consider a mail merge whose WordprocessingML definition includes the following:

```
<w:checkErrors w:val="3" />
```

The presence of a checkErrors element with a val attribute of 3 indicates that the hosting application must conduct the type of error reporting specified above, performing the mail merge, populating fields with mapped external data and reporting errors in a new document . *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.14.5 colDelim (Column Delimiter for Data Source)

This element specifies the character which shall be interpreted as the column delimiter used to separate columns within external data sources. The character representing the specific delimiter used for the external data source referenced by a source or merged WordprocessingML document is specified via a decimal number representing the decimal number for the Unicode character representation within this element's val attribute.

If this element is omitted, then no column delimiter shall be specified for the data source in this mail merge.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:colDelim w:val="44" />
```

Here, the colDelim element's val attribute specifies that the given external data source is using the comma character (,) to delimit column data, as 44 is the decimal value for the Unicode character representation of a comma. *end Example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent

Attributes	Description
	XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.14.6 column (Index of Column Being Mapped)

This element specifies the zero-based index of the column within a given external data source which shall be mapped to the local name of a specific `MERGEFIELD` field (§17.16.5.35) specified by the parent field mapping data. The val attribute specifies this index value, which is used to look up the appropriate column in the data source.

If this element is omitted, or its value exceeds the number of columns in the associated data source, then the index of the referenced column shall be assumed to be `0`.

[*Example*: Consider a source document that is connected to an external data source with three columns. Within this external data source, these are three columns are ordered and titled as follows: `first`, `middle`, and `last`, respectively. The following WordprocessingML specifies that when this document was connected to the data source, these columns were ordered in this manner:

```
<w:fieldMapData>
  …
  <w:name w:val="first" />
  <w:column w:val="0" />
</w:fieldMapData>
<w:fieldMapData>
  …
  <w:name w:val="middle" />
  <w:column w:val="1" />
</w:fieldMapData>
<w:fieldMapData>
  …
  <w:name w:val="last" />
  <w:column w:val="2" />
</w:fieldMapData>
```

The WordprocessingML above demonstrates that the column titled `first` must be associated with the first column in the external database by specifying a column element with its val attribute equal to `0.` In addition, the column titled `middle` must be associated with the second column in the external database by specifying a column element with its val attribute equal to `1.` Finally, the column titled `last` must be associated with the third column in the external database by specifying a column element with its val attribute equal to 2. *end example*]

It is important to realize that the name element's values are a cache of the last time the document was connected to the database, and the indices specified must be used to connect the field mappings with the columns in the data source. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.14.7 column (Index of Column Containing Unique Values for Record)

This element specifies the column within the specified external data source that contains unique data for the current record within that data source. This element shall be used in conjunction with the uniqueTag element (§17.14.35) to maintain a relationship between a specific record within an external data source and a given source or merged document. The val attribute on this element shall be interpreted as a zero-based index into the columns specified by the data source, specifying the resulting column as the column in which the uniqueTag element shall be looked up.

If this element specifies a column number which exceeds the number of columns in the specified external data source, then its value shall be ignored.

[*Note*: This information is necessary as part of a mail merge as records can be added or deleted from external data sources, and a means must be provided to maintain record-specific inclusion or exclusion data using the active element (§17.14.1) and the affected external data record when the WordprocessingML document is reconnected to the external data source irrespective of the ordering of the records within the external data source. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the information about a single record in a source document for a mail merge:

```
<w:recipientData>
  <w:active w:val="0" />
  <w:column w:val="12" />
  <w:uniqueTag>258865469</w:uniqueTag>
</w:recipientData>
```

The external data record associated with this information is specified via the column in the external data source corresponding to the column element with a val attribute equal to 12, which contains a row whose value in this column has a value corresponding to the uniqueTag element (§17.14.35) with a val attribute equal to 258865469. This record is not imported into the merged WordprocessingML document as the active (§17.14.1) element associated with the given external data record has a val attribute equal to 0.

In other words, when the specified external data source is connected to, the record within the thirteenth column of the external data source that has the contents 258865469, and not populate mapped fields in a merged document with data from that record. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element.

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	[*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.14.8 connectString (Data Source Connection String)

This element specifies the connection string used to reconnect to an external data source. The string within this element's val attribute shall contain the connection string that the hosting application shall pass to a external data source access application to enable the WordprocessingML document to be reconnected to the specified external data source.

[*Note*: This string is generally comprised of a series of name/value pairs, delimited by semicolons, determined by the data source access application and the external data source that is accessed. *end note*]

If this string is omitted, then no legacy connection string shall be associated with this mail merge.

This connection string should be ignored under the following conditions:

- The udl element (§17.14.34) is present within the mail merge data
- The dataType element (§17.14.10) is set to `native`
- The current application is able to use the information contained in the odso element (§17.14.25) to access the data source

[*Guidance*: In this case, using the connection string in the udl element provides an equal or greater amount of connection information for the mail merge data source for clients which support it. *end guidance*]

[*Example*: Consider a merged WordprocessingML document that has been connected to an external data source for the purposes of a mail merge. The following WordprocessingML fragment represents the legacy connection string used to connect to the external data source when the merged WordprocessingML document is reopened:

```
<w:connectString w:val="Provider=Example;Password=Test;User ID=readonly;…" />
```

The connectString element specifies that the string `Provider=Example;Password=Test;User ID=readonly;…` must be used to enable the given WordprocessingML document to be reconnected to the specified external data source. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.14.9 dataSource (Data Source File Path)

This element specifies the relationship whose target is the location of the external data source to be connected to a given WordprocessingML document to perform the mail merge (for a source document) or to find the associated field data (for a merged document).

If this element is omitted, then no file location is specified for the data source for the current mail merge. If no relationship exists with the given relationship ID, or this relationship is not of type http://purl.oclc.org/ooxml/officeDocument/relationships/mailMergeSource then this document shall be considered non-conformant.

The data source location can also be ignored under the following conditions:

- The src element (§17.14.30) is present within the mail merge data
- The dataType element (§17.14.10) is set to `native`
- The current application is able to use the information contained in the odso element (§17.14.25) to access the data source

[*Guidance*: In this case, using the data source file path in the src element provides an equal or greater amount of information for the mail merge data source for clients which can consume it. *end guidance*]

[*Example*: Consider a WordprocessingML source document containing the following mail merge data:

```
<w:mailMerge>
  …
  <w:dataSource r:id="rId1" />
  …
</w:mailMerge>
```

This mail merge's dataSource element specifies via its r:id attribute value of `rId1` that the external data source to be connected to the given WordprocessingML document is the data source targeted by the relationship whose Id attribute is equal to `rId1`. If we examine the corresponding relationship part item for the setting part, as follows:

```
<Relationships>
  <Relationship Id="rId1"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/mailMergeSource"
Target="file:///c:/example_file.mdb" TargetMode="External" />
</Relationships>
```

Since the relationship whose Id attribute value is `rId1` specifies the source file path for the data source, that data source effectively specifies a file path of `c:\example_file.mdb`. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element

Attributes	Description
	- http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rel) is located in §A.1. *end note*]

17.14.10 dataType (Data Source Type)

This element specifies the type of external data source to be connected to via the Dynamic Data Exchange (DDE) system (such as a spreadsheet or database), or the alternative method of data access if the Dynamic Data Exchange system is not used. This setting is purely a suggestion of the data source access mechanism which shall be used, and can be ignored in favor of an alternative mechanism if one is present.

[*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document:

```
<w:dataType w:val="odbc" />
```

The dataType element's val attribute is equal to `odbc`, specifying that the given merged WordprocessingML document has been connected to an external data source via the Open Database Connectivity interface. *end example*]

Attributes	Description
val (Value)	Specifies the exact type of external data source to which a given merged WordprocessingML document is to be connected. [*Example*: Consider the following WordprocessingML fragment for a mail merge document. `<w:dataType w:val="database" />` The val attribute is equal to `database`, specifying that the given WordprocessingML document has been connected to a database via the Dynamic Data Exchange (DDE) system. *end example*] The possible values for this attribute are defined by the ST_MailMergeDataType simple type (§17.18.52).

©ISO/IEC 2012 – All rights reserved

[*Note*: The W3C XML Schema definition of this element's content model (CT_MailMergeDataType) is located in §A.1. *end note*]

17.14.11 destination (Merged Document Destination)

This element specifies what the result which shall be generated when a mail merge is carried out on a given WordprocessingML source document. In other words, this element is used to specify what is to be done with the merged documents that result from populating the fields within a given merged WordprocessingML document with data from the specified external data source.

If this element is omitted, then the default destination of merged documents shall be assumed to be of type `newDocument`.

[*Note*: The aspects of the mail merge outside of connecting to an external data source and populating the fields within a given merged document with external data from the specified external data source are not specified by ISO/IEC 29500.

For example, if a given merged WordprocessingML document contains a destination element with its val attribute equal to `email`, the hosting application can surface a user interface specific to creating emails with the data resulting from populating fields within a given merged WordprocessingML document with external data from the specified external data source. WordprocessingML only provides a flag (via the destination element) to tell the hosting application to surface this user interface. *end note*]

[*Example*: Consider a WordprocessingML source document containing the following WordprocessingML:

```
<w:mailMerge>
  <w:destination w:val="newDocument" />
  …
</w:mailMerge>
```

The destination element's val attribute is set to `newDocument`, specifying that when the mail merge is carried out, the source document must be used to generate a specified number of new documents, which can be handled as appropriate. *end example*]

Attributes	Description
val (Mail Merge Merged Document Type)	Specifies the type of merged documents which shall be the result of carrying out a mail merge on a given source WordprocessingML document. [*Example*: Consider the WordprocessingML mail merge data specified as follows: `<w:destination w:val="email" />` This specifies that a given merged WordprocessingML document is used by the hosting application to generate e-mails containing the static contents of the merged document as

Attributes	Description
	well as external data populated into mapped fields. *end example*] The possible values for this attribute are defined by the ST_MailMergeDest simple type (§17.18.53).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MailMergeDest) is located in §A.1. *end note*]

17.14.12 doNotSuppressBlankLines (Remove Blank Lines from Merged Documents)

This element specifies how an application performing the mail merge shall handle blank lines in the merged documents resulting from the mail merge. Typically, when a mail merge is performed, any blank lines which result from lines whose sole contents are merge fields with no content are removed from the merged document in order to prevent extraneous blank lines from appearing in the merged documents. When this element is present, the merged documents which are generated from the mail merge shall not have any blank lines removed before they are sent to their destination format.

If this element is omitted, the merged documents generated from this mail merge shall have all blank lines suppressed if they consist of only merge fields with values consisting of empty strings.

[*Example*: Consider a WordprocessingML document containing a single WordprocessingML field calling for `Test` data as seen in the first column of the table below. If the current record in the mail merge data source contains an empty string for the `Test` column, the resulting merged document would be displayed as follows depending on the setting for the doNotSuppressBlankLines element:

Source Document	`<w:doNotSuppressBlankLines val="true" />`	`<w:doNotSuppressBlankLines val="false" />`
One Two {MERGEFIELD "Test" } Three	One Two Three	One Two Three

With this element set to a value of `true`, the blank lines in the resulting document must not be suppressed when the resulting merged documents are created. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.14.13 dynamicAddress (Use Country-Based Address Field Ordering)

This element specifies that the contents of the `AddressBlock MERGEFIELD` field shall be dynamically ordered based on the country associated with the current record or if the country-invariant version of the address field shall be used in its place. [*Rationale*: When a source document is combined with the contents of a data source in order to produce multiple merged documents, it is often necessary to use an address form specific to the destination country for each particular record in the data source, rather than one static address form for all records. *end rationale*] If this element is set to `true`, then the mail merge shall use an address form suited to the country associated with the current record in the external data source.

If this element is omitted, then the form of the address shall be dynamically determined based on the country specified in the current record.

[*Example*: Consider a merged WordprocessingML document that is specified to *not* dynamically create the address field order based on the country associated with the current record. This requirement might be specified using the following WordprocessingML:

```
<w:fieldMapData>
  …
  <w:dynamicAddress w:val="off" />
</w:fieldMapData>
```

The dynamicAddress element is set to a value of `off`, specifying that the dynamic address format must not be used when performing a mail merge with the specified data source. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.14.14 fHdr (First Row of Data Source Contains Column Names)

This element specifies that a hosting application shall treat the first row of data in the specified external data source as a header row containing the names of each column in the data source, rather than data to populate mapped fields in a merged document. When present, this information shall not change the indices specified in the recipientData elements (§17.14.27), but shall indicate that the first row is not part of the mail merge when it is performed.

If this element is omitted, then the first row of the data source shall not be considered a header row when a mail merge is performed.

[*Example*: Consider a WordprocessingML source document that has been connected to an external data source whose first row of data is not data the hosting application is to populate mapped fields with, but rather contains column names for each column in the data source. This setting on the data source is specified using the following fragment of WordprocessingML:

```
<w:fHdr w:val="on" />
```

The fHdr element specifies that the data source's first row is a header row, rather than regular data to be used in the mail merge. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.14.15 fieldMapData (External Data Source to Merge Field Mapping)

This element specifies how a column specified in the external data source that has been connected to a WordprocessingML document shall be mapped to the pre-defined MERGEFIELD fields (§17.16.5.35) within the given merged document's contents. Each instance of a fieldMapData element contains the information needed to map one column in the external data source to a single type of pre-defined MERGEFIELD field for the purposes of the mail merge in the current document.

[*Example*: Consider a single merged document. The WordprocessingML below demonstrates the mapping of the Country column from the external data source to the predefined WordprocessingML Country or Region merge field when the merged document is populated with external data as part of a mail merge:

```
<w:odso>
  …
  <w:fieldMapData>
    <w:type w:val="dbColumn" />
    <w:name w:val="Country" />
    <w:mappedName w:val="Country or Region" />
    <w:column w:val="9" />
    …
  </w:fieldMapData>
</w:odso>
```

The fieldMapData element specifies the mapping between the external data source and a single merge field as follows: the child elements specify that the tenth column in the data source, last titled Country in the specified external data source when the connection was last made is to be mapped to the predefined WordprocessingML merge field calling for Country or Region data.

With the fieldMapData element configured as such, an application can be used in conjunction with this WordprocessingML document to populate the document with data mapped from the specified external data source to fields within the merged document. *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_OdsoFieldMapData) is located in §A.1. *end note*]

17.14.16 headerSource (Header Definition File Path)

This element specifies the location of a file that contains the column header information which shall be used when connecting to an external data source that does not have column header data specified. Specifically, this element specifies a file that corresponds with the aforementioned external data source. [*Note*: Column headers are needed to enable a hosting application to associate external data source's columns to fields via the fieldMapData element (§17.14.15).

If this element is omitted, then the column header definition data is not specified in an external file and shall be retrieved from the primary data source associated with the mail merge.

[*Example*: Consider a WordprocessingML merged document containing the following WordprocessingML:

```
<w:settings>
  …
  <w:headerSource r:id="rId2" />
  …
</w:settings>
```

This mail merge's headerSource element specifies via its r:id attribute value of `rId2` that the external data source to be used for the column header information for the given WordprocessingML document is the data source targeted by the relationship whose Id attribute is equal to `rId2`. If we examine the corresponding relationship part item for the setting part, as follows:

```
<Relationships>
  <Relationship Id="rId2"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/mailMergeSource"
Target="file:///c:/headerData.txt" TargetMode="External" />
</Relationships>
```

Since the relationship whose Id attribute value is `rId2` specifies the column header data file path for the data source effectively specifies a file path of `c:\headerData.txt`. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rel) is located in §A.1. *end note*]

17.14.17 lid (Merge Field Name Language ID)

This element specifies the language ID for the language which was used to generate the merge field name which was associated with a given column in the data source, as specified by the fieldMapData element (§17.14.15). This element specifies that when this field mapping is processed by an application, it shall interpret the merge field name as the name for the specified language.

If this element is omitted, then the mapped field names specified in the current document can be interpreted using any method desired by the consuming application (i.e. no language data is included with the field mapping information).

[*Example*: Consider the following WordprocessingML fragment for a field mapping for a document to be merged to an external data source. If the merge field name stored in the file corresponds with the U.S. English version of the merge field names, that information would be stored as follows:

```
<w:fieldMapData>
  <w:name w:val="Title" />
  <w:mappedName w:val="Courtesy Title" />
  <w:column w:val="3" />
  <w:lid w:val="en-US" />
  …
</w:fieldMapData>
```

The lid element specifies that the mapping of the data contained in the fourth column of external data source named Title to the WordprocessingML of 'Courtesy Title', must be associated with the U.S. English language as specified by the val attribute equal to en-US. *end example*]

Attributes	Description
val (Language Code)	Specifies an identifier for a specific language. This code is interpreted in the context of the parent XML element. [*Example*: Consider an object which must specify the English(Canada) language. That object would use an identifier of en-CA to specify this language. *end example*] The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Lang) is located in §A.1. *end note*]

17.14.18 linkToQuery (Query Contains Link to External Query File)

This element specifies that the current WordprocessingML document's query string, stored in the query element (§17.14.26) and used to specify the data to be imported from the external data source, actually contains a reference to an external query file which contains the actual query data to be used against the specified external data source for the mail merge. This query shall mimic a SQL query and be of the following form: `SELECT * FROM <query file path>`.

If this element is omitted, then the query specified for the data source attached to the current document shall be assumed to not be a query containing a link to an external file.

[*Example*: Consider a mail merge source document that uses the linkToQuery element to specify that the query used is stored in the specified external data source as follows:

```
<w:mailMerge>
  …
  <w:linkToQuery />
  <w:query w:val="SELECT * FROM C:\queryExample.txt" />
  …
</w:mailMerge>
```

The linkToQuery element specifies that the query string stored in the query element (§17.14.26) is actually just a reference to an externally stored query file, in this case, an external query file stored at `c:\queryExample.txt`. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.14.19 mailAsAttachment (Merged Document To E-Mail Attachment)

This element specifies that, after importing external data into fields to generate a series of destination WordprocessingML documents as e-mails, the resulting documents should be emailed as an attachment rather than the body of the actual e-mail.

If this element is omitted, then its value shall be assumed to be `false` (i.e. the destination source is not an e-mail attachment). If the destination element (§17.14.11) specifies that the merged document destination is not `email`, then this element shall be ignored.

[*Example*: Consider a merged WordprocessingML document that has been connected to an external data source containing three records and that contains the following WordprocessingML in its mail merge properties as follows:

```
<w:mailMerge>
  …
  <w:destination w:val="email" />
  <w:mailAsAttachment />
  …
</w:mailMerge>
```

After the external data has been imported into the merged document's respective merge fields, three emails is generated (as specified by the destination element (§17.14.11) with a val attribute of `email`, each an attachment consisting of one of the three documents resulting from the mail merge result (rather than just including the merged document as the body of the email). *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.14.20 mailMerge (Mail Merge Settings)

This element specifies all of the mail merge information for a document that has been connected to an external data source as part of a mail merge operation.

The document which contains this mail merge data can be of one of two types:

- A *source document*, the document which contains all of the information for the mail merge, and is used in conjunction with an application to connect to an external data source and create one document for each record in that data source.
- A *merged document*, a document which contains all of the information for the mail merge as well as a reference to a single specific record which shall be used to populate the values of all of the merge fields in that document.

The information in this element shall contain all data needed to connect to a data source and populate any merge fields in the document with data from that data source.

[*Example*: Consider the following WordprocessingML fragment for a document which is part of a mail merge:

```
<w:mailMerge>
  …
  <w:dataType w:val="spreadsheet" />
  <w:query w:val="SELECT * FROM `Sheet1$`" />
  <w:dataSource r:id="rId1" />
  …
</w:mailMerge>
```

Here, the dataType (§17.14.10) and dataSource (§17.14.9) elements specify that the given document must be connected to the external data source referenced by the relationship whose id value is equal to `rId1`. While connected to the external data source, the document together with a hosting application can extract data from the external data source as specified by the query (§17.14.26) element. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_MailMerge) is located in §A.1. *end note*]

17.14.21 mailSubject (Merged E-mail or Fax Subject Line)

This element specifies the text which shall appear in the subject line of the e-mails or faxes that result after the actions of a mail merge have imported external data into fields within a merged WordprocessingML document whose destination, as specified in the destination element (§17.14.21), is `email` or `fax`.

If this element is omitted, then no subject line text shall be associated with each merged document produced via a mail merge using the specified mail merge data. If the destination element (§17.14.11) specifies that the merged document destination is not `email` or `fax`, this element shall be ignored.

[*Example*: Consider a merged WordprocessingML document containing fields and the following WordprocessingML as part of its mail merge data:

```
<w:mailMerge>
  …
  <w:destination w:val="email" />
  <w:mailSubject w:val="Example Subject Line" />
  …
</w:mailMerge>
```

The mailSubject element specifies that after the specified external data has been imported into the specified fields in the merged document, each record merged must result in a single e-mail message, each with their subject line reading `Example Subject Line`. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`

Attributes	Description
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.14.22 mainDocumentType (Source Document Type)

This element specifies the document type of a given WordprocessingML source document.

If this element is omitted, then its value shall be assumed to be `formLetters`.

[*Note*: This element is generally used in conjunction with the behavior of an application to customize aspects of the mail merge user interface and experience independent of the WordprocessingML file format. For example, if a given WordprocessingML merged document contains a mainDocumentType element with its val attribute equal to `envelopes`, the hosting application can surface a piece of user interface specific to creating envelopes when the given document is opened.

In addition, what a hosting application does with the documents that result from importing external data into specified fields can be determined based on the mainDocumentType element, but other than this, is independent of a given merged document's WordprocessingML. For example, if a given merged WordprocessingML document contains a mainDocumentType element with its val attribute equal to `email`, the hosting application can call a email service after importing external data into specified fields, in order to generate emails containing the resulting documents.

WordprocessingML simply provides the mainDocumentType that can serve as a trigger for an application to surface user interface specific to a document type of mail merge. *end note*]

[*Example*: Consider the WordprocessingML below:

```
<w:mailMerge>
  <w:mainDocumentType w:val="formLetters" />
  …
</w:mailMerge>
```

In this example, the source document is of the `formLetters` type, as specified by the mainDocumentType element's val attribute being equal to `formLetters`. *end example*]

Attributes	Description
val (Mail Merge	Specifies the type of source document which is specified by the given WordprocessingML

Attributes	Description
Source Document Type)	document. [*Example*: Consider the WordprocessingML below: `<w:mainDocumentType w:val="formLetters" />` This WordprocessingML specifies that a given source document is a `formLetters` document. This setting implies nothing about the file, but can be interpreted by an application as desired. *end example*] The possible values for this attribute are defined by the ST_MailMergeDocType simple type (§17.18.54).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MailMergeDocType) is located in §A.1. *end note*]

17.14.23 mappedName (Predefined Merge Field Name)

This element specifies the predefined WordprocessingML `MERGEFIELD` field name which shall be mapped to the column number specified by the column element (§17.14.6) within this field mapping. [*Guidance*: This element allows the current column from the specified data source to be mapped to a predefined field name, allowing applications to have one standard set of field names to use regardless of the data source column names, for example, to create the address formats to place into an `ADDRESSBLOCK` field. *end guidance*]

If this element is omitted, then the current data source column mapping shall not have a predefined merge field name mapped to its contents, and shall only be referenced via the data source column name specified by the name element (§17.14.24) when referenced by one or more `MERGEFIELD` fields. If the application does not have a predefined merge field whose name matches the name specified using the val attribute, then this element can be ignored.

[*Example*: Consider the following WordprocessingML fragment, representing two columns from an external data source which have been mapped to the built-in fields `First Name` and `Last Name`, respectively:

```
<w:fieldMapData>
  <w:name w:val="Column Name A" />
  <w:mappedName w:val="First Name" />
  <w:column w:val="0" />
  …
</w:fieldMapData>
<w:fieldMapData>
  <w:name w:val="Column Name B" />
  <w:mappedName w:val="Last Name" />
  <w:column w:val="1" />
```

```
    …
  </w:fieldMapData>
```

The first and second columns, specified by the column element values of `0` and `1` respectively, specify that the predefined WordprocessingML field names `First Name` and `Last Name` are mapped to the columns of the external data source, and the data source names for those columns are `Column Name A` and `Column Name B`, respectively.

Therefore, if `MERGEFIELD` fields calling for `First Name` and `Last Name` are inserted in a WordprocessingML document connected to the external data source with the field mappings specified above, when the mail merge takes place, the data from the first and second column populates the fields calling for `First Name` and `Last Name` data within the merged WordprocessingML document. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_String) is located in §A.1. *end note*]

17.14.24 name (Data Source Name for Column)

This element specifies the column name within a given external data source for the column whose index is specified via the column element (§17.14.6). This data source name provides a column name which shall be used to map a specific MERGEFIELD field in the document, as specified by the parent field mapping data. The val attribute specifies the name of this column in the data source when the connection is initially established, which is then used permanently to link columns in the database to MERGEFIELD fields in the document.

If this element is omitted, no data source name is provided for the current column.

[*Example*: Consider a source document that is connected to an external data source with three columns. Within this external data source, these are three columns are ordered and titled as follows: first, middle, and last, respectively. The following WordprocessingML specifies that when this document was connected to the data source, these columns were ordered in this manner:

```
<w:fieldMapData>
  …
  <w:name w:val="first" />
  <w:column w:val="0" />
</w:fieldMapData>
<w:fieldMapData>
  …
  <w:name w:val="middle" />
  <w:column w:val="1" />
</w:fieldMapData>
<w:fieldMapData>
  …
  <w:name w:val="last" />
  <w:column w:val="2" />
</w:fieldMapData>
```

The WordprocessingML above demonstrates that the column name first must be associated with the first column in the external database by specifying a column element with its val attribute equal to 0. In addition, the column name middle must be associated with the second column in the external database by specifying a column element with its val attribute equal to 1. Finally, the column name last must be associated with the third column in the external database by specifying a column element with its val attribute equal to 2. *end example*]

Attributes	Description
val (String Value)	Specifies that its contents contain a string. The contents of this string are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment:

Attributes	Description

	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.14.25 odso (Office Data Source Object Settings)

This element specifies a group of additional settings for the mail merge information which comprise an extension to the standard settings stored with a mail merge which performs two functions:

- First, it provides additional information about the mail merge data source, specifically: information about how to map the columns in the data source to `MERGEFIELD` fields and information about records which shall be included and excluded when creating merged documents, and column delimiters used in text data sources. This information can be used regardless of the value of the dataType element (§17.14.10) when it is present.
- Second, it provides an alternate set of connection information which should be used when the dataType element (§17.14.10) specifies a value of `native`. This alternate connection string provides additional connection information for applications which choose to support the ODSO connection string syntax. If the dataType element (§17.14.10) specifies that the data type of the current mail merge is not `native`, then the second group of settings specified within this element shall be ignored in favor of their non-ODSO equivalents.

[*Example*: Consider the WordprocessingML for a source document whose mail merge includes mail merge information including ODSO settings as follows:

```
<w:odso>
  <w:udl w:val="…" />
  <w:table w:val="Sheet1$" />
  <w:src r:id="rId1" />
  <w:colDelim w:val="9" />
  <w:fHdr w:val="1" />
  <w:fieldMapData>
    <w:type w:val="dbColumn" />
    <w:name w:val="Title" />
    <w:mappedName w:val="Courtesy Title" />
    <w:lid w:val="en-US" />
  </w:fieldMapData>
  <w:recipientData r:id="rId2" />
  …
</w:odso>
```

The odso element and its child elements provide all of the information specified above that is needed to carry out a mail merge with the current document. This includes alternate connection information using the udl, table, and src elements, and additional mail merge information in the other child elements. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Odso) is located in §A.1. *end note*]

17.14.26 query (Query For Data Source Records To Merge)

This element contains the Structured Query Language string (as defined by the normative reference in§3) that shall be run against the specified external data source to return the set of records from the external data which shall be imported into merged WordprocessingML documents when the mail merge operation is performed.

If this element is omitted, then no query shall be associated with the current data source.

[*Example*: Consider a WordprocessingML document that has been connected to an external database. In addition, consider that the data specifies that the table within the database titled `Documentation` shall be the specific table whose data is imported. This shall be specified in WordprocessingML as follows:

```
<w:query w:val="SELECT * FROM Documentation" />
```

The query element specifies the syntax for the data source query via its val attribute. *end example*]

Attributes	Description
val (String Value)	Specifies that its contents contain a string. The contents of this string are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment:

Attributes	Description

	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	
	However, consider the following fragment:
	
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.14.27 recipientData (Data About Single Data Source Record)

This element specifies information about a single record within an external data source. If a record shall be merged into a merged document, then no information is needed about that record within this part. However, if a given record shall not be merged into a merged document, then the value of the unique key for that record shall be stored within the uniqueTag element as a child of this element (along with the active element) to indicate this exclusion.

[*Note*: This mapping is necessary in place of simply using the element order to correspond to the record indices in the external data source, as records can be added or deleted from external data sources, and a means must be provided to maintain WordprocessingML external record specific data like that specified in the active element (§17.14.1) and the corresponding external data record when the WordprocessingML document is reconnected to the external data source; irrespective of the ordering of the records within the external data source. In other words, this element, and its child elements enable merged WordprocessingML documents to maintain the relationship between the records within an external data and record specific WordprocessingML parameters. *end note*]

[*Example*: Consider a merged WordprocessingML document that:

1. Has been connected to a specified external data source containing three records; and
2. Has been configured by the hosting application to not populate a merged document with the record pertaining to `John Smith` in the external data source.

Consider also that the first time the given WordprocessingML document was connected to the external data source, John Smith's record was in the second record in the data source.

When this merged document is connected to the external data source the recipientData element can be used to store the number and value of the column containing the unique key for each data record within the external data source including John Smith's. This setting is represented using the following WordprocessingML to use the hash codes within the recipientData element to uniquely identify the three records within the external data source.

```
<w:recipientData>
  <w:column w:val="1" />
  <w:uniqueTag>1408613399</w:uniqueTag>
</w:recipientData>
<w:recipientData>
  <w:active w:val="0" />
  <w:column w:val="1" />
  <w:uniqueTag>870254691</w:uniqueTag>
</w:recipientData>
<w:recipientData>
  <w:column w:val="1" />
  <w:uniqueTag>1107777181</w:uniqueTag>
</w:recipientData>
```

Here, the first, second (John Smith record), and third records within the specified data source whose unique key values are 1408613399, 870254691, and 1107777181 have been associated with with recipient data via the active element to specify that the record associated with the given record (John Smith's record) must not be used to populate a merged WordprocessingML document.

With these association in place, if a fourth record is added to the given external data source above John Smith's record, when the given merged WordprocessingML document is reconnected to the external data source, the hosting application still knows that John Smith's record must not be used to populate a merged WordprocessingML document as it is associated via its unique key value and is not dependent on the given record's ordinal position within the external data source. *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_RecipientData) is located in §A.1. *end note*]

17.14.28 recipientData (Reference to Inclusion/Exclusion Data for Data Source)

This element shall specify a reference to the part which contains data about whether the set of records in the associated data source have been explicitly included or excluded from the specified mail merge. Only those records which must not be used to generate merged WordprocessingML documents shall be stored within the referenced part, as all records shall be merged by default as part of the mail merge operation. [*Guidance*: Applications can choose to store only those records which are excluded for efficiency, or a list of all records in order to determine which set of records were added/removed between mail merge operations. *end guidance*]

[*Rationale*: When defining a mail merge, it is possible that a user wishes to connect to a specified data source, but specify only a subset of the records returned by the query specified by the query element (§17.14.26) which must be merged as part of the mail merge operation. This element allows applications to utilize a separate part to store this information, either the shared part defined by ISO/IEC 29500, or an application-defined part as needed. *end rationale*]

If the relationship type of the relationship specified by this element is not http://purl.oclc.org/ooxml/officeDocument/relationships/mailMergeRecipientData, is not present, or does not have a TargetMode attribute value of `Internal`, then the document shall be considered non-conformant. If an application cannot process external content of the content type specified by the targeted part, then it can be ignored.

ISO/IEC 29500 defines one shared mechanism for storing this data: using the Mail Merge Recipient Data part. This mechanism shall be used if the associated data source has a column which can be used as the unique key. However, when using data sources which do not have a unique key, applications can store their own part (of an application-defined content type) using this relationship.

[*Example*: Consider a WordprocessingML document which is a mail merge source document, containing inclusion/exclusion data for the data source. The document settings part would contain the mail merge data:

```
<w:settings>
  …
  <w:mailMerge>
     …
     <w:odso>
       …
       <w:recipientData r:id="recipient1" />
     </w:odso>
  </w:mailMerge>
</w:settings>
```

The recipientData element specifies that the external content targeted by the relationship with an ID of `recipient1` contains the recipient inclusion/exclusion data for the mail merge operation. Examining the contents of the corresponding relationship part item, we can see the targets for that relationship:

```
<Relationships … >
  …
  <Relationship Id="recipient1" TargetMode="Internal"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/mailMergeRecipientData,"
Target="recipientData.xml" />
  …
</Relationships>
```

The corresponding relationship part item shows that the file containing this data is located next to the main document and is named `recipientData.xml`. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rel) is located in §A.1. *end note*]

17.14.29 recipients (Inclusion/Exclusion Data for Data Source)

This element specifies all of the inclusion/exclusion data for the contents of the specified mail merge data source. It is the root element for the Mail Merge Recipient Data part.

[*Example*: Consider a document which is the source document for a mail merge operation. If two records of the three specified by the data source were excluded from the mail merge, the resulting recipient data part would appear as follows:

```
<w:recipients>
  <w:recipientData>
    <w:active w:val="false"/>
    …
  </w:recipientData>
```

```
    <w:recipientData>
      <w:active w:val="false"/>
      …
    </w:recipientData>
    <w:recipientData>
      …
    </w:recipientData>
  </w:recipients>
```

The recipients element contains all of the recipient inclusion/exclusion data for this mail merge document. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Recipients) is located in §A.1. *end note*]

17.14.30 src (ODSO Data Source File Path)

This element specifies the relationship whose target is the location of the external data source to be connected to a given WordprocessingML document to perform the mail merge (for a source document) or to find the associated field data (for a merged document) when the merge type, specified using the dataType element (§17.14.10), is set to native.

If this element is omitted, then no file location is specified for the data source for the current mail merge. If no relationship exists with the given relationship ID, or this relationship is not of the Mail Merge Data Source relationship type, then this document shall be considered non-conformant.

The data source location is only used under the following conditions:

- The dataType element (§17.14.10) is set to native
- The current application is able to use the information contained in the odso element (§17.14.25) to access the data source

[*Guidance*: In this case, using the data source file path in the src element provides an equal or greater amount of information for the mail merge data source for clients which can consume it. *end guidance*]

[*Example*: Consider a WordprocessingML source document containing the following mail merge data:

```
  <w:odso>
    …
    <w:src r:id="rId1" />
    …
  </w:odso>
```

This mail merge's src element specifies via its r:id attribute value of rId1 that the external data source to be connected to the given WordprocessingML document is the data source targeted by the relationship whose Id attribute is equal to rId1. If we examine the corresponding relationship part item for the setting part, as follows:

```
<Relationships>
  <Relationship Id="rId1"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/mailMergeSource"
Target="file:///c:/example_file.mdb" TargetMode="External" />
</Relationships>
```

Since the relationship whose Id attribute value is rId1 specifies the source file path for the data source, that data source effectively specifies a file path of c:\example_file.mdb. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID rId1 contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rel) is located in §A.1. *end note*]

17.14.31 table (Data Source Table Name)

This element specifies the particular set of data that a source or merged WordprocessingML document shall be connected to within an external data source containing multiple data sets. In other words, when connecting to a

WordprocessingML document to an external data source that can have more than one repository of data within it, such as a database with multiple tables or a spreadsheet with multiple worksheets, this element is used to distinguish the specific table or spreadsheet from which data is imported from within the external data source.

[*Example*: Consider a WordprocessingML document that has been connected to database containing two tables named `Table One` and `Table Two`, respectively. To specify that the mail merge must import data from `Table One` into the WordprocessingML document, this requirement would be specified using the following WordprocessingML:

```
<w:odso>
  …
  <w:table w:val="Table One" />
  …
</w:odso>
```

The table element with a value of `Table One` specifies that the external data must be retrieved from this table in the data source. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.14.32 type (ODSO Data Source Type)

This element specifies the type of external data source to be connected to via as part of the ODSO connection information for this mail merge. This setting is purely a suggestion of the data source type which is being used for this mail merge, and can be ignored in favor of an alternative mechanism if one is present.

[*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document:

```
<w:type w:val="database" />
```

The type element's val attribute is equal to `database`, specifying that the given merged WordprocessingML document has been connected to an external data source via the ODSO settings, and that the resulting data source was a database. *end example*]

Attributes	Description
val (Data Source Type Value)	Specifies the type of an external data source used for a mail merge operation. [*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document: `<w:type w:val="text" />` The val attribute is equal to `text`, specifying that the given merged WordprocessingML document has been connected to an external data source via the ODSO settings, and that the resulting data source was a text file. *end example*] The possible values for this attribute are defined by the ST_MailMergeSourceType simple type (§17.18.56).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MailMergeSourceType) is located in §A.1. *end note*]

17.14.33 type (Merge Field Mapping)

This element specifies if a given mail merge field has been mapped to a column in the given external data source or not.

If this element is omitted, then the field mapping shall be considered to be of type `null` (i.e. not mapped).

[*Example*: Consider the WordprocessingML for a single field mapping within a mail merge source document:

```
<w:odso>
  …
  <w:fieldMapData>
    <w:type w:val="dbColumn" />
    <w:name w:val="Country" />
    <w:mappedName w:val="Country or Region" />
    <w:column w:val="9" />
    …
  </w:fieldMapData>
</w:odso>
```

In this example, the `country` column within the given external data source must be mapped to the mail merge field `Country or Region`, as specified by the type element's val attribute being equal to `dbColumn`. *end example*]

Attributes	Description
val (Merge Field Mapping Type)	Specifies if the given mail merge field has been mapped to a column in the given external data source (i.e. if the merge field mapping is active or not). [*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document: `<w:type w:val="null" />` In this example, the given mail merge field must not be mapped to a column in the given external data source, as specified by the type element's val attribute being equal to `null`. *end example*] The possible values for this attribute are defined by the ST_MailMergeOdsoFMDFieldType simple type (§17.18.55).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_MailMergeOdsoFMDFieldType) is located in §A.1. *end note*]

17.14.34 udl (UDL Connection String)

This element specifies the Universal Data Link (UDL) connection string used to reconnect to an external data source. The string within this element's val attribute shall contain the connection string that the hosting application shall pass to a external data source access application to enable the WordprocessingML document to be reconnected to the specified external data source.

If this string is omitted, then no UDL connection string shall be associated with the ODSO data for this mail merge.

This connection string is only used under the following conditions:

- The dataType element (§17.14.10) is set to native
- The current application is able to use the information contained in the odso element (§17.14.25) to access the data source

[*Guidance*: In this case, using the connection string in the udl element provides an equal or greater amount of information for the mail merge data source for clients which can consume it. *end guidance*]

[*Example*: Consider a merged WordprocessingML document that has been connected to an external data source for the purposes of a mail merge. The following WordprocessingML fragment represents the legacy connection string used to connect to the external data source when the merged WordprocessingML document is reopened:

```
<w:udl w:val="Provider=Example;Password=Test;User ID=readonly;…" />
```

The udl element specifies that the string `Provider=Example;Password=Test;User ID=readonly;…` must be used to enable the given WordprocessingML document to be reconnected to the specified external data source. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_String) is located in §A.1. *end note*]

17.14.35 uniqueTag (Unique Value for Record)

This element specifies the contents of a given record within the specified external data source, in the column containing unique data for every record within the external data source. given external data source. This element is used in conjunction with the column (§17.14.6) element to maintain a relationship between the records within an external data source and a given merged WordprocessingML document.

The contents of this attribute shall be the base64-encoded value of the unique tag value as specified by the data source.

[*Note*: This information is necessary as part of a mail merge as records can be added or deleted from external data sources, and a means must be provided to maintain record-specific inclusion or exclusion data using the active element (§17.14.1) and the affected external data record when the WordprocessingML document is reconnected to the external data source irrespective of the ordering of the records within the external data source. *end note*]

[*Example*: Consider the following WordprocessingML fragment for the information about a single record in a source document for a mail merge:

```
<w:recipientData>
  <w:active w:val="0" />
  <w:column w:val="12" />
  <w:uniqueTag>258865469</w:uniqueTag>
</w:recipientData>
```

The external data record associated with this information is specified via the column in the external data source corresponding to the column element with a val attribute equal to 12, which contains a row whose value in this column has a value corresponding to the uniqueTag element with a val attribute equal to 258865469. This record is not imported into the merged WordprocessingML document as the active (§17.14.1) element associated with the given external data record has a val attribute equal to 0.

In other words, when the specified external data source is connected to, the record within the thirteenth column of the external data source that has the contents 258865469, and not populate mapped fields in a merged document with data from that record. *end example*]

The possible values for this element are defined by the W3C XML Schema CT_Base64Binarydatatype.

17.14.36 viewMergedData (View Merged Data Within Document)

This element specifies that a specific merged document shall display the data from the specified external data source where merge fields have been inserted. The activeRecord element (§17.14.2) is used to specify which record within the external data source is to have its applicable data displayed where applicable within the WordprocessingML merged document.

If the activeRecord element is not present in the WordprocessingML for the document with the viewMergedData's val attribute equal to true, the hosting application can behave as if the acitveRecord element's val attribute was equal to 1. This element is ignored if the viewMergedData (§17.14.36) element is not present or present with a val attribute equal to `Off`, `0`, or `false`.

[*Example*: Consider a merged WordprocessingML document containing two WordprocessingML fields calling for `Courtesy Title` and `Last Name` data and a sample text paragraph. Also, note that the external data source this merged document is connected to contains two records, both containing name and address information, with the first record pertaining to `Mr.` Doe, and the second pertaining to `Ms. Smith`.

This table below illustrates the necessary WordprocessingML to display applicable data from the specified external data source within the merged document where fields have been inserted:

`<w:viewMergedData val="0"/>`	`<w:viewMergedData val="1" />`
	`<w:activeRecord w:val="1" />`
		`<w:viewMergedData val="0" />`
		`<w:activeRecord w:val="1" />`

Dear {MERGEFIELD "Courtesy Title" \m} {MERGEFIELD "Last Name" \m}, Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,	Dear Mr. Doe: Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,	Dear {MERGEFIELD "Courtesy Title" \m} {MERGEFIELD "Last Name" \m}, Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sample letter text. Sincerely,

The viewMergedData element specifies that the specified record in the external data source must be displayed in place of merge fields in the current document. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15 Settings

Within a WordprocessingML document, *settings* specify stored preferences which shall be used when processing the contents of the document. These settings are typically divided into three categories:

- *Document Settings* - settings which influence the appearance and behavior of the current document, as well as store document-level state.
- *Compatibility Settings* - settings which tell applications to perform behaviors which are designed to maintain visual output of previous word processing applications. These settings are for backward compatibility and are all ignorable.
- *Web Settings* - settings which affect how this document shall be handled when it is saved as HTML. These settings exist primarily for backward compatibility reasons and are all ignorable.

The first two groups are stored in the Document Settings part, and the last group is stored in the Web Settings part.

17.15.1 Document Settings

The first group of settings stored in WordprocessingML is document settings. These settings specify all document-level properties which affect the handling of the current document.

[*Example*: Consider the following WordprocessingML fragment for the document settings in a WordprocessingML document:

```
<w:settings>
  <w:defaultTabStop w:val="720" />
  <w:characterSpacingControl w:val="doNotCompress" />
</w:settings>
```

The settings element contains all of the document settings for this document. In this case, the two settings applied are automatic tab stop increments of 0.5" using the defaultTabStop element (§17.15.1.25), and no character level whitespace compression using the characterSpacingControl element (§17.15.1.18). *end example*]

17.15.1.1 activeWritingStyle (Grammar Checking Settings)

This element specifies information about the parameters of the grammar checking which was performed on the contents of the current WordprocessingML document. [*Note*: This information can be used as desired by applications; for example, to determine if the current grammar checking state, specified by the proofState element (§17.15.1.65) is sufficient. *end note*]

[*Example*: Consider the following WordprocessingML fragment from the document settings:

```
<w:activeWritingStyle w:lang="en-CA" w:vendorID="64" w:dllVersion="131078"
w:nlCheck="1" w:checkStyle="0" w:appName="testApp" />
```

The activeWritingStyle element's lang attribute specifies that the English (Canada) language setting for grammatical and stylistic checks must be applied; the vendorID attribute specifies information about the vendor associated with the DLL used to perform the grammatical and stylistic checks; the dllVersion attribute specifies the version of this DLL; the nlCheck attribute specifies if natural language checks were performed or not; the checkStyle attribute specifies that the hosting application should allow its grammar engine to check both the grammar and style of the given WordprocessingML document, if that functionality is available; and the appName attribute indicates that an application called `testApp` specified the grammar checking rules of the given WordprocessingML. *end example*]

Attributes	Description
appName (Application Name)	Specifies the name of the application which specified the grammatical settings contained on the attributes for this element.

Attributes	Description
	If an application reads these settings and does not understand the value of this attribute, then its settings can be ignored and the application's default settings used instead. [*Example*: Consider the WordprocessingML below: `<w:activeWritingStyle … w:appName="testApp"/>` The appName attribute has a value of `testApp`, specifying that the application called `testApp` specified the grammar checking rules of the given WordprocessingML document. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
checkStyle (Check Stylistic Rules With Grammar)	Specifies if the grammar content checking performed on this document included stylistic rules for the document content. If specified, applications which support this functionality shall check stylistic rules as well as grammatical ones when checking the grammatical content of this document. [*Example*: Consider the WordprocessingML below: `<w:activeWritingStyle … w:checkStyle="false"/>` The checkStyle attribute has a value of `false`, specifying that hosting applications must only check grammatical rules of the given WordprocessingML document. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
dllVersion (Grammatical Check Engine Version)	Specifies the version of the engine that was used to check the grammatical content of the WordprocessingML document. [*Example*: Consider the following WordprocessingML fragment: `<w:activeWritingStyle … w:dllVersion="131078" />` The dllVersion attribute specifies that the writing style DLL version used to check the writing style of is the writing style DLL version associated with the string `131078`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
lang (Writing Style Language)	Specifies the language of the engine used to perform the grammatical content checking. [*Example*: Consider the following WordprocessingML fragment: `<w:activeWritingStyle w:lang="en-CA" …/>`

Attributes	Description
	The lang attribute has a value of en-CA, therefore the grammatical check language is specified as English (en) and Canada (CA), resulting in use of the English (Canada) grammar checker. *end example*] The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).
nlCheck (Natural Language Grammar Check)	Specifies whether the engine that was used to check the grammatical content of the WordprocessingML document performed natural language-based analysis. [*Example*: Consider the WordprocessingML below: `<w:activeWritingStyle … w:nlCheck="1" />` The nlCheck attribute has a value of 1, specifying that the writing style DLL supported natural language analysis . *end example*]. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
vendorID (Grammatical Engine ID)	Specifies a value indicating a unique ID for the writing style engine that was used to check the grammatical content of the WordprocessingML document. [*Example*: Consider the WordprocessingML below: `<w:activeWritingStyle … w:vendorID="64"/>` The vendorId attribute has a value of 64, specifying that the grammatical checker used is identified by the string 64.*end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_WritingStyle) is located in §A.1. *end note*]

17.15.1.2 alignBordersAndEdges (Align Paragraph and Table Borders with Page Border)

This element specifies that paragraph borders specified using the pBdr element (§17.3.1.24) and table borders using the tblBorders element (§17.4.39) shall be adjusted to align with extents of the page border defined using the pgBorders element (§17.6.10) if the spacing between these borders is less than or equal to 10.5 points (one character width) or less from the page border. The presence of this setting shall ensure there are no gaps of one character width or less between adjoining page and paragraph/table borders, as borders which are perfectly aligning shall not be displayed in favor of the intervening page border.

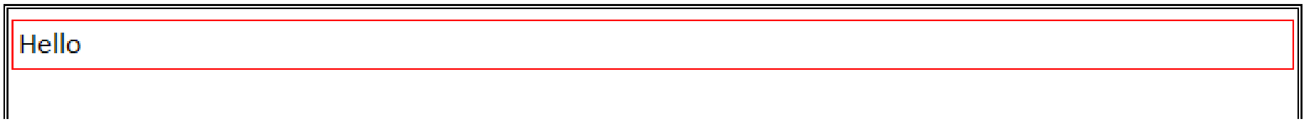
If this element is omitted, then borders shall not be automatically adjusted to prevent gaps of less than one character width. If the page border is not measured from the text extents using a value of `text` in the offsetFrom attribute on the pgBorders element, then it can be ignored.

©ISO/IEC 2012 – All rights reserved

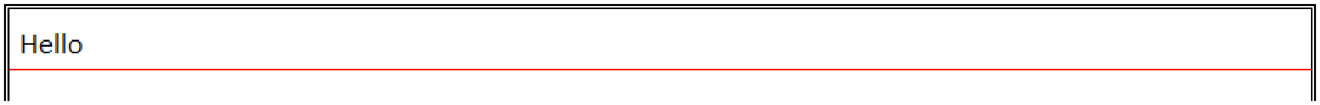
[*Example*: Consider the following WordprocessingML fragment from the document settings:

```
<w:alignBordersAndEdges w:val="true"/>
```

The alignBordersAndEdges element has a value of true specifying that borders must be adjusted to prevent gaps of less than one character width. If a document has a page border specified to appear 4 points from the text extents, and within that page a paragraph border specified to appear one point from the text extents, that would normally appear like this:



If this element is present, then those gaps (which are all of three points in width) must be adjusted to ensure that the borders align exactly and the paragraph border is suppressed:



end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.3 alwaysMergeEmptyNamespace (Do Not Mark Custom XML Elements With No Namespace As Invalid)

This element specifies whether custom XML markup specified via the customXml element which has no associated namespace shall be treated as an error and moved into a special error namespace (for the purposes of XML schema validation) when the document is opened. If this element is turned on, when an application determines that the current XML markup is in the empty namespace, those elements shall not automatically be moved into an error namespace.

If this element is not present in a WordprocessingML document than custom XML markup which has no associated namespace shall be treated as an error and moved into a special error namespace when the document is opened.

[*Example*: Consider a WordprocessingML document which should not automatically flag empty namespace XML as invalid. This requirement would be specified using the following WordprocessingML:

```
<w:alwaysMergeEmptyNamespace w:val="true"/>
```

The alwaysMergeEmptyNamespace element's val attribute has a value of true specifying that custom XML markup in the empty namespace must never be treated as an error. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.4 alwaysShowPlaceholderText (Use Custom XML Element Names as Default Placeholder Text)

This element specifies that each custom XML element specified using the customXml element within this document shall always show some form of in-document placeholder text presentation when it contains no run content. If the placeholder element (§17.5.2.25) is present in the custom XML element's properties, then this is the placeholder text displayed and this effect has no effect. If the placeholder element is omitted, then the application shall use the name of the element to generate default placeholder text in its place.

If this element is omitted, then custom XML markup which does not contain a placeholder element within its properties shall not display any placeholder text.

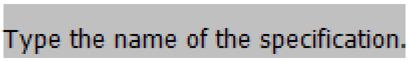
[*Example*: Consider the following WordprocessingML fragment from the document settings:

```
<w:alwaysShowPlaceholderText w:val="true" />
```

The alwaysShowPlaceholderText element has a value of true, which specifies that placeholder text must be generated using the element's name if no placeholder text is present. If two custom XML elements are defined as follows:

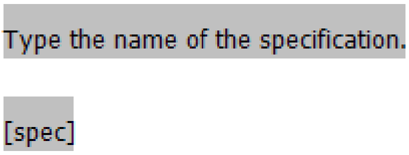
```
<w:customXml w:name="spec" … >
  <w:customXmlPr>
    <w:placeholder w:val="Type the name of the specification." />
  </w:customXmlPr>
</w:customXml>
…
<w:customXml w:name="spec" … >
</w:customXml>
```

The first custom XML element has placeholder text, and the second doesn't, so if this element is omitted, these two elements might be displayed as follows:



Type the name of the specification.

Notice that the second custom XML element has no placeholder text, and therefore is not displayed. However, when this element is present, then the application should generate default placeholder text in its place:



Type the name of the specification.

[spec]

The application generated default placeholder text from the element name, resulting in a value of `[spec]` in the document. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.5 attachedSchema (Attached Custom XML Schema)

This element specifies that the custom XML schema whose target namespace matches the value specified in the val attribute should be associated with this document when it is loaded, if such a schema is available to the hosting application. Applications can also load and utilize any additional schemas as well as those explicitly mentioned here. [*Note*: These custom XML schemas can then be used to validate the structure of the custom XML markup in the document, etc. *end note*]

If no elements of this type are present, then no custom XML schemas have been explicitly associated with the contents of this document.

[*Example*: Consider the following WordprocessingML fragment from the document settings:

```
<w:attachedSchema w:val="http://www.example.com/schema1" />
<w:attachedSchema w:val="http://www.example.com/schema2" />
```

The attachedSchema elements specify that two custom XML schemas with namespaces of `http://www.example.com/schema1` and `http://www.example.com/schema2` should be associated with the custom XML markup in the current document. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest

Attributes	Description
	ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.15.1.6 attachedTemplate (Attached Document Template)

This element specifies the location of a document template which shall be attached to the current WordprocessingML document if it is accessible and of a format supported by an application. Specifically, this element's `val` attribute shall contain the file path of the associated document template.

If this element is omitted, then the document shall not have an attached document template, and applications should use their default template in its place.

[*Example*: Consider a WordprocessingML document which is attached to a WordprocessingML template located on the local C drive within a file whose name is `c:\template.dotx`. This association is specified using the following WordprocessingML:

```
<w:attachedTemplate r:id="rId1" />
```

The attachedTemplate element contains the ID of the relationship which references the associated template. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element

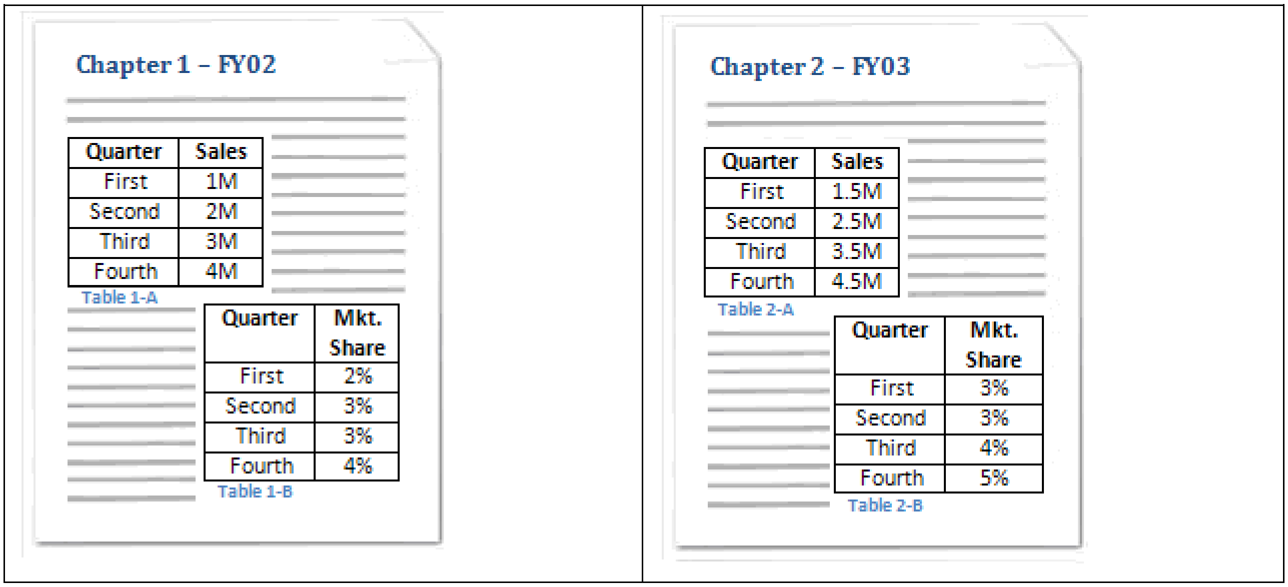
Attributes	Description
	[*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rel) is located in §A.1. *end note*]

17.15.1.7 autoCaption (Single Automatic Captioning Setting)

This element specifies what type(s) of objects shall automatically labelled with captions (§17.15.1.17), and with which captions the specified objects shall be labelled as defined in the caption element (§17.15.1.16).

[*Example*: Consider the diagram below illustrating a two page WordprocessingML document that has leveraged WordprocessingML to automatically label WordprocessingML tables with a specified caption when tables are inserted into the given document.



Chapter 1 – FY02

Quarter	Sales
First	1M
Second	2M
Third	3M
Fourth	4M

Table 1-A

Quarter	Mkt. Share
First	2%
Second	3%
Third	3%
Fourth	4%

Table 1-B

Chapter 2 – FY03

Quarter	Sales
First	1.5M
Second	2.5M
Third	3.5M
Fourth	4.5M

Table 2-A

Quarter	Mkt. Share
First	3%
Second	3%
Third	4%
Fourth	5%

Table 2-B

This type of automatic captioning is specified using the following WordprocessingML fragment:

```
<w:captions>
  <w:caption w:name="Table" w:pos="below" w:chapNum="1" w:heading="2"
w:numFmt="upperLetter" w:sep="hyphen" />
  <w:autoCaptions>
    <w:autoCaption w:name="wfwTable" w:caption="Table" />
  </w:autoCaptions>
</w:captions>
```

Here, the autoCaption element specifies through the name attribute being set equal to `wfwTable` that tables are automatically be labeled with the caption specified in the caption element whose name attribute is equal to `Table`, as the caption element's caption attribute has a value of `Table`. *end example*]

Attributes	Description

caption (Caption Used for Automatic Captioning)	Specifies the caption defined in using the caption element (§17.15.1.16) which shall be used to automatically label a given type of object inserted in a WordprocessingML document. The caption settings are linked by matching the value of this attribute with the name attribute of the corresponding caption element.
	[*Example*: Consider the WordprocessingML below
	`<w:captions>`
	`  <w:caption w:name="table" w:pos="below" w:chapNum="1" w:heading="0" w:noLabel="1" w:numFmt="upperRoman" />`
	`  <w:autoCaptions>`
	`    <w:autoCaption w:name="Paint.Picture" w:caption="table" />`
	`  </w:autoCaptions>`
	`</w:captions>`
	The autoCaption element specifies through the name attribute being set equal to `wfwTable` that tables are automatically be labeled with the caption whose name attribute is equal to `Table` (specified by the caption element's attribute name having a value of `Table`). *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

name (Identifier of Object to be Automatically Captioned)	Specifies a unique identifier which can be used to associate objects inserted into the document which are to be automatically labeled with a caption when inserted into the WordprocessingML document. [*Example*: Consider the WordprocessingML below specifying that WordprocessingML tables should be labeled with the `custom` caption: `<w:autoCaption w:name="wfwTables" w:caption="custom" />` The name attribute value of `wfwTables` specifies that WordprocessingML tables must be

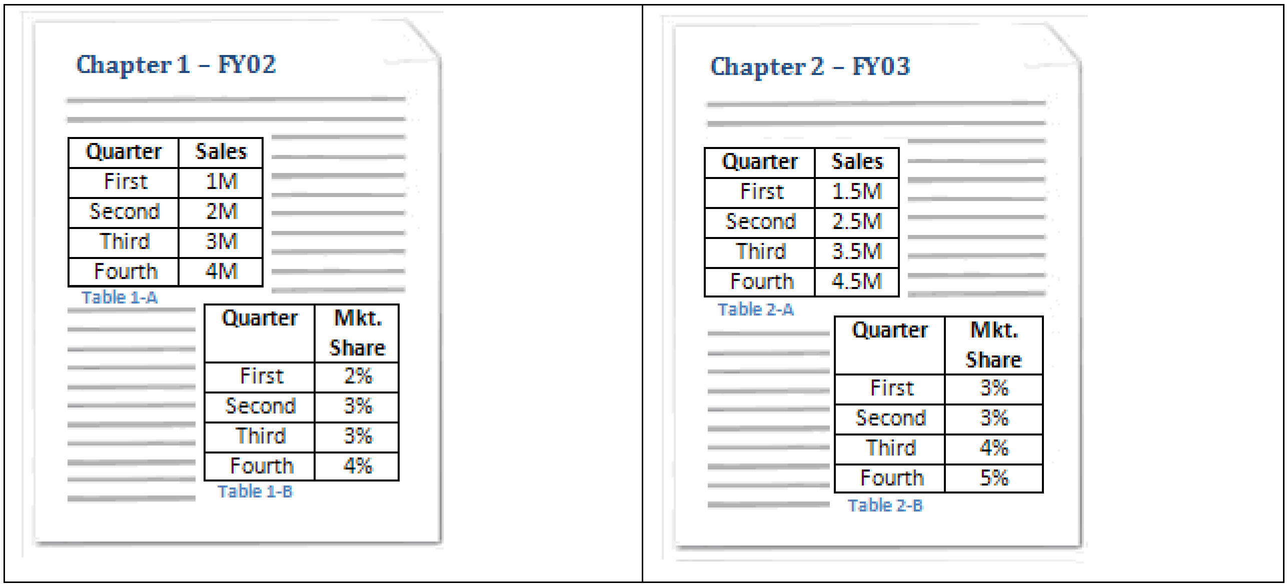
Attributes	Description
	labeled with the `custom` caption. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_AutoCaption) is located in §A.1. *end note*]

17.15.1.8 autoCaptions (Automatic Captioning Settings)

This element specifies that one or more types of objects, when inserted into a WordprocessingML document, are automatically be labeled with a specific caption defined using the caption element (§17.15.1.16).

[*Example*: Consider the following example illustrating a two page WordprocessingML document that has leveraged WordprocessingML to automatically label WordprocessingML tables with a specified caption.



Chapter 1 – FY02

Quarter	Sales
First	1M
Second	2M
Third	3M
Fourth	4M

Table 1-A

Quarter	Mkt. Share
First	2%
Second	3%
Third	3%
Fourth	4%

Table 1-B

Chapter 2 – FY03

Quarter	Sales
First	1.5M
Second	2.5M
Third	3.5M
Fourth	4.5M

Table 2-A

Quarter	Mkt. Share
First	3%
Second	3%
Third	4%
Fourth	5%

Table 2-B

This type of automatic captioning is specified using the following WordprocessingML fragment:

```
<w:captions>
  <w:caption w:name="Table" w:pos="below" w:chapNum="1" w:heading="2" 
w:numFmt="upperLetter" w:sep="hyphen" />
  <w:autoCaptions>
    <w:autoCaption w:name="wfwTable" w:caption="Table" />
  </w:autoCaptions>
</w:captions>
```

The autoCaptions element specifies set of objects that when inserted into a WordprocessingML document are automatically be labeled with a given caption. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_AutoCaptions) is located in §A.1. *end note*]

17.15.1.9 autoFormatOverride (Allow Automatic Formatting to Override Formatting Protection Settings)

This element specifies whether formatting automatically applied by an application (i.e. not explicitly applied by a use or an application) shall be allowed to override formatting protection enabled via the formatting attribute on the documentProtection element (§17.15.1.9) when those formatting operations would add formatting which has been explicitly disabled. [*Example*: Automatically adding superscript to the `st` in the string `1st`. *end example*]

If this element is omitted, then no automatic formatting rule(s) shall be allowed to override the formatting restrictions enabled for the document.

[*Example* Consider a WordprocessingML document which has been protected such that a user must not be able to directly format text within the document. Consider also that the hosting application has been constructed such that if a user enters a ampersand, then one or more alphabetical characters, then another ampersand, that the alphabetical characters are to take on italicized formatting.

If the autoFormatOverride element is omitted or set to `false` and document protection is enabled, the aforementioned series of events does not cause the English alphabetical characters to be italicized as the document protection preventing formatting of the document in question supersedes the formatting to take place after these events. If this operation should not be prevented when active formatting restrictions are used, this would be specified using the following WordprocessingML:

```
<w:autoFormatOverride w:val="true"/>
```

The autoFormatOverride element's val attribute is equal to `true` specifying that the automatic formatting behavior shall be applied regardless of the formatting restrictions in place. *End Example*]

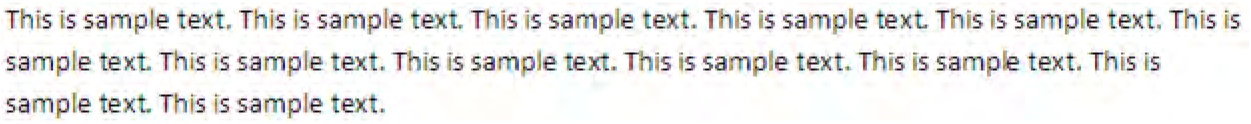
This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.10 autoHyphenation (Automatically Hyphenate Document Contents When Displayed)

This element specifies whether the content of a given WordprocessingML document should automatically be hyphenated by the hosting application before it is displayed, if the application supports this functionality.

If this element is omitted, then hyphenation shall not automatically be performed by application displaying this document.

[*Example*: Consider the images below illustrating a paragraph of text in a WordprocessingML document:

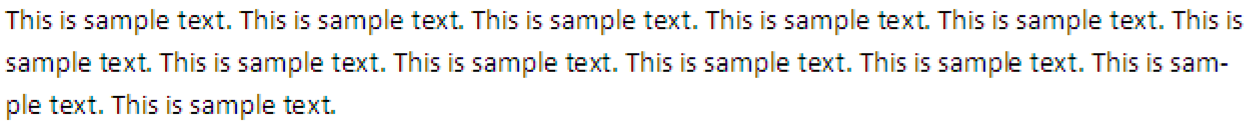


This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is
sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is
sample text. This is sample text.

If the content in this document must automatically be hyphenated when it is displayed, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:autoHyphenation w:val="true" />
```

The resulting output might look like the following (depending on the application's hyphenation algorithm and the hyphenation zone setting (§17.15.1.53):



This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is
sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sam-
ple text. This is sample text.

The autoHypehenation element has its val attribute equal to `true`, the document is automatically hyphenated and the word `sample`, beginning at the end of the second line, is hyphenated automatically and thus carried over onto the third line. Conversely, when the autoHypehnation element has its val attribute equal to `off`, the entire word `sample` is carried over to the third line as it was not hyphenated automatically and could not fit onto the second line. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.11 bookFoldPrinting (Book Fold Printing)

This element specifies if the contents of a given WordprocessingML document should be printed as signatures. *Signatures* are printed *sheets*, which depict several pages of a document that are folded and bound with other signatures to form a booklet, a set of which can be bound together to form a book like publication. Specifically, this element specifies that each page in a given WordprocessingML document should be oriented in a landscape fashion, divided in half vertically with two left margins emanating from the bisector of the page, and two right margins instantiated at the left and right side of each page.

This element is used in conjunction with the bookFoldPrintingSheets element (§17.15.1.12) to enable a WordprocessingML document to be printed such that the series of signatures printed can be folded and bound to create a booklet.

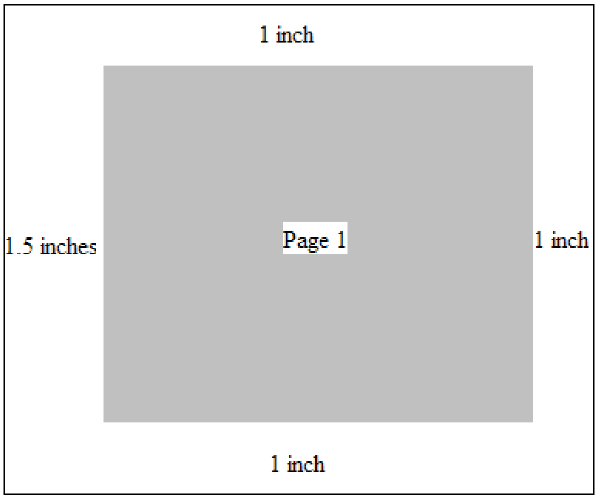
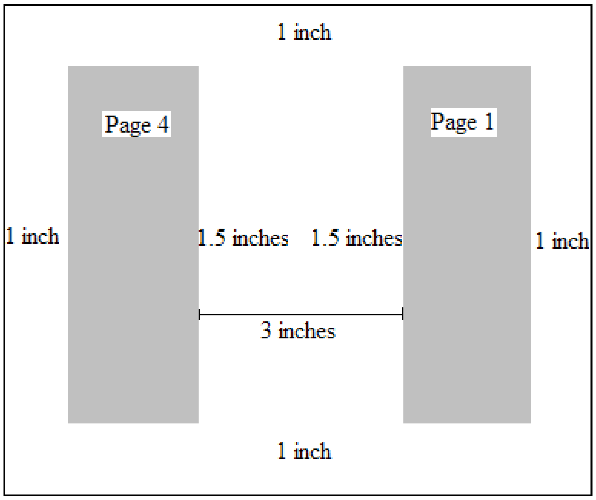
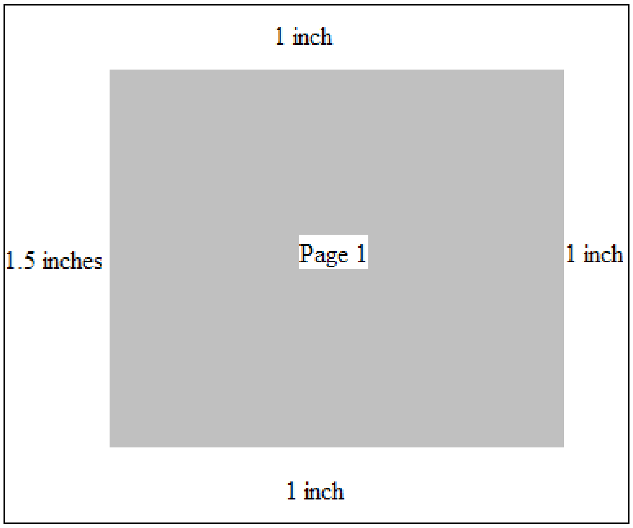
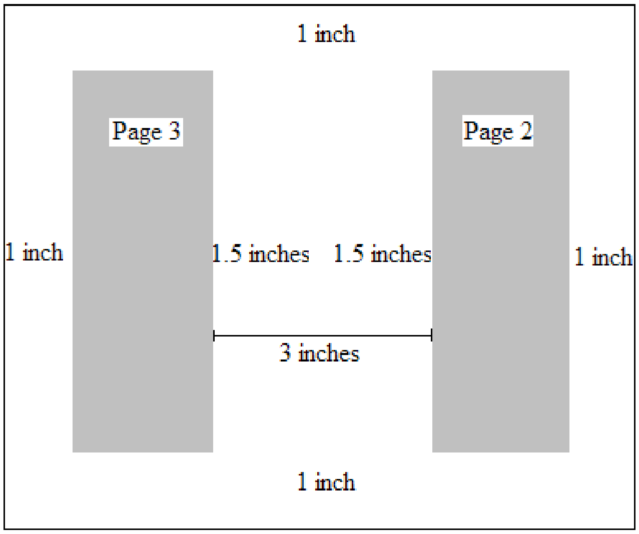
This element has no impact on the settings of printer leveraged by the hosting application. In other words, if the printer leveraged by the hosting application has been configured to print on one side of a page, including the WordprocessingML for this element has no effect.

If this element is omitted, then pages shall not be printed as signatures. If the bookFoldRevPrinting element (§17.15.1.13) is also specified, then this element shall be ignored.

[*Example*: Consider a four page WordprocessingML document with a 2,160 twentieths of a point (one and a half inch) left margin, and 1,440 twentieths of a point (one inch) bottom, right, and top margins using the pgMar element (§17.6.11) surrounding the text extents of the page (represented by the gray shaded area in diagrams below). These page margins are specified using the following WordprocessingML:

```
<w:pgMar w:header="0" w:top="1440" w:right="1440" w:bottom="1440" w:left="2160"
  w:footer="720" w:gutter="0" />
```

The necessary WordprocessingML and consequential effect of setting the bookFoldPrinting element's val attribute to `true` versus `false` and the bookFoldPrintingSheets element's val attribute to 4, is depicted graphically below—diagrams not drawn to scale:

`<w: bookFoldPrinting w:val="false"/>`	`<w: bookFoldPrinting w:val="true" />`
First Printed Sheet	First Printed Signature
	 1 inch Page 4 Page 1 1 inch 1.5 inches 1.5 inches 1 inch 3 inches 1 inch
Second Printed Sheet	Second Printed Signature
	 1 inch Page 3 Page 2 1 inch 1.5 inches 1.5 inches 1 inch 3 inches 1 inch

Assuming the page was already oriented in a landscape fashion, setting the bookFoldPrinting element's val attribute to `true` divided the page in half vertically, with two left margins emanating from the bisector of the page, and right margins instantiated at the left and right side of each page, enabling two signatures to be printed.

In addition, this element is used in conjunction with the bookFoldPrintingSheets element to enable the given WordprocessingML document to be printed such that the series of signatures printed can be folded and bound to create a booklet. Specifically, the signatures can be placed back to back, with top the bottom of each sheet aligned, and folded such that a booklet is created. *end example*]

[*Note*: This element could also be leveraged by the hosting application to notify the application to display two pages per sheets within its user interface to allow for a WYSIWYG user experience. *end note*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.12 bookFoldPrintingSheets (Number of Pages Per Booklet)

This element shall be used in conjunction with the bookFoldPrinting (§17.15.1.11) and bookFoldRevPrinting (§17.15.1.13) elements to specify the number of pages to be included in each booklet when printing a series of signatures. Signatures are printed *sheets*, which depict several pages of a document that are to be folded and bound with other signatures to form a booklet. Booklets can be bound together to form a book like publication.

If this element is omitted, then its default behavior shall be to print the contents of the content on a single sheet. A *sheet* is a single piece of paper which is folded and cut to produce a book.

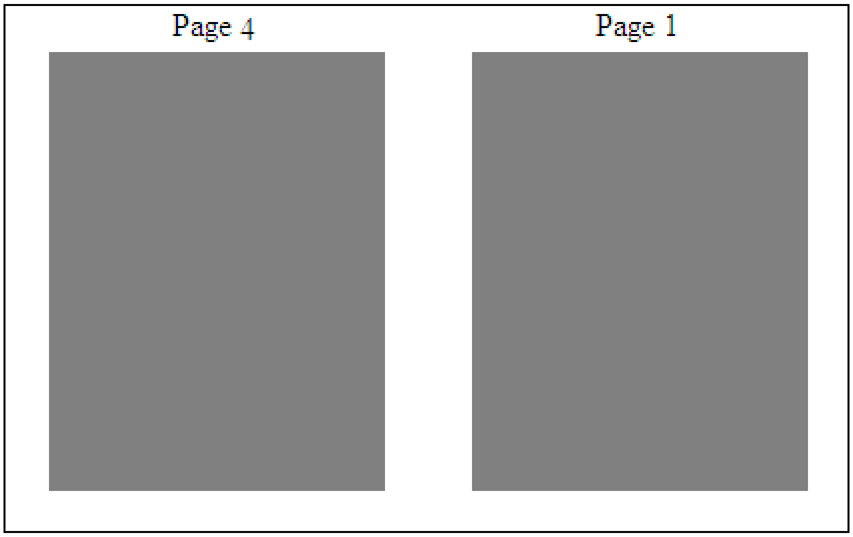
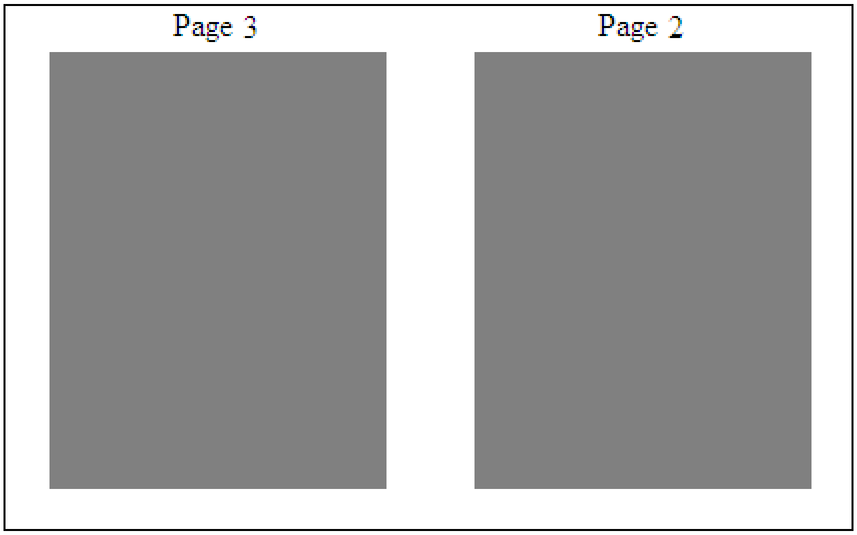
[*Example*: Consider a four page WordprocessingML document is printed as a set of two signatures to be compiled into a single booklet. This setting would be specified using the following WordprocessingML fragment in the document settings part:

```
<w:bookFoldPrinting w:val="true" />
<w:bookFoldPrintingSheets w:val="4" />
```

The bookFoldPrintingSheets element's val attribute specifies that 4 pages must be included in each booklet. Since each signature contains two pages and are printed such that the signatures can be placed back to back, with top the bottom of each sheet aligned, and folded such that the booklet is created, a booklet containing four pages distributed over two signatures can be created.

This setting is depicted visually using the illustration below (gray shading represents a page):

First Printed Signature


Second Printed Signature


end example]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.15.1.13 bookFoldRevPrinting (Reverse Book Fold Printing)

This element specifies if pages of a given WordprocessingML document are to be printed as signatures in reverse order. *Signatures* are printed *sheets*, which depict several pages of a document that are folded and bound with other signatures to form a booklet, a set of which can be bound together to form a book like publication. Specifically, this element specifies that each page in a given WordprocessingML document should be oriented in a landscape fashion and divided in half vertically, with two left margins emanating from the bisector of the page, and right margins instantiated at the left and right side of each page.

In addition, this element is used in conjunction with the bookFoldPrintingSheets element (§17.15.1.12) to enable given WordprocessingML document to be printed such that the series of signatures printed can be folded and bound to create a booklet.

This element has no impact on the settings of printer leveraged by the hosting application. In other words, if the printer leveraged by the hosting application has been configured to print on one side of a page, including the WordprocessingML for this element has no effect.

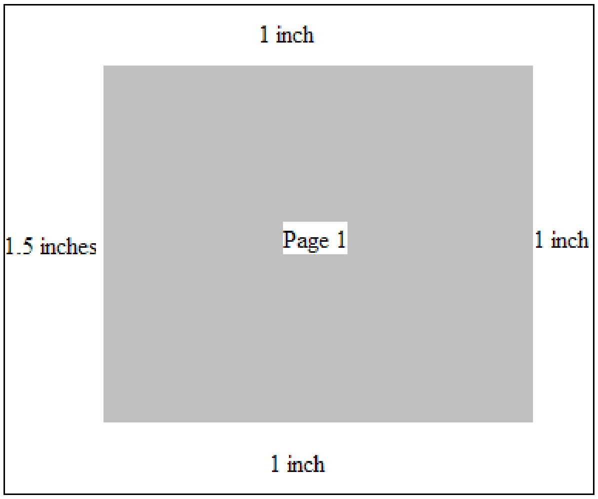
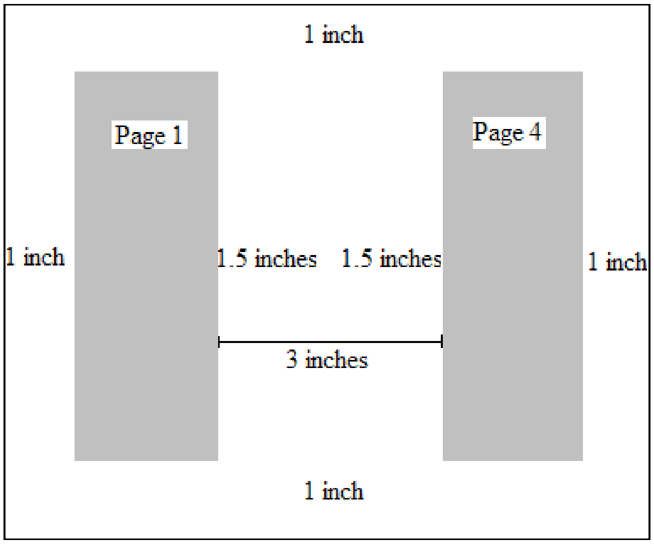
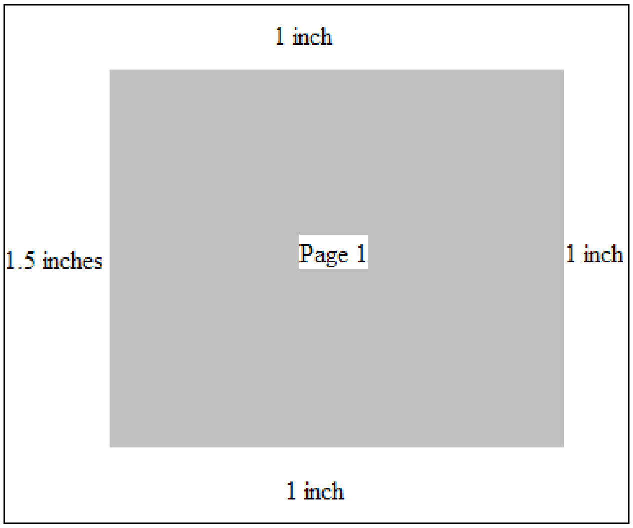
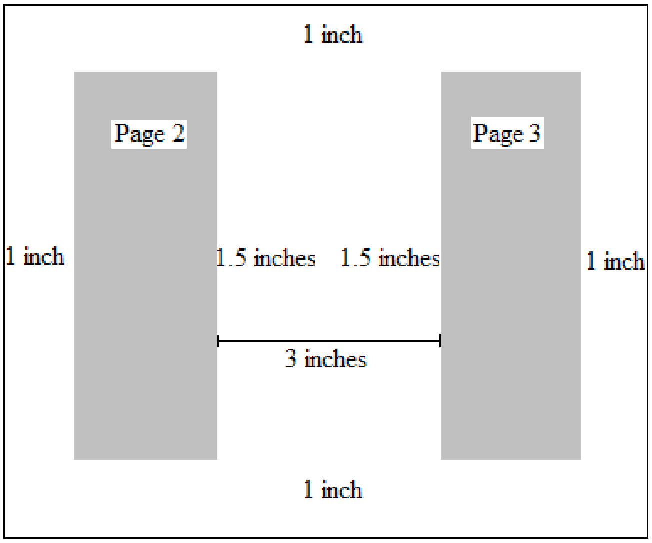
If this element is omitted, then pages shall not be printed as reverse book fold signatures. If the bookFoldPrinting element (§17.15.1.11) is also specified, then that element shall be ignored, and this element shall be used instead.

[*Example*: Consider a four page WordprocessingML document with a 2,160 twentieths of a point (one and a half inch) left margin, and 1,440 twentieths of a point (one inch) bottom, right, and top margins using the pgMar element (§17.6.11) surrounding the text extents of the page (represented by the gray shaded area in diagrams below). These page margins are specified using the following WordprocessingML:

```
<w:pgMar w:header="0" w:top="1440" w:right="1440" w:bottom="1440" w:left="2160"
  w:footer="720" w:gutter="0" />
```

The necessary WordprocessingML and consequential effect of setting the bookFoldRevPrinting element's val attribute to `true` versus `false` and the bookFoldPrintingSheets element's val attribute to 4, is depicted graphically below—diagrams not drawn to scale:

`<w: bookFoldRevPrinting w:val="false"/>`	`<w: bookFoldRevPrinting w:val="true"/>`
	`<w: bookFoldPrintingSheets w:val="4"/>`

First Printed Sheet	First Printed Signature
	 1 inch Page 1 Page 4 1 inch 1.5 inches 1.5 inches 1 inch 3 inches 1 inch
Second Printed Sheet	Second Printed Signature
	 1 inch Page 2 Page 3 1 inch 1.5 inches 1.5 inches 1 inch 3 inches 1 inch

Assuming the page was already oriented in a landscape fashion, setting the bookFoldRevPrinting element's val attribute to true divided the page in half vertically, with two left margins emanating from the bisector of the page, and right margins instantiated at the left and right side of each page, enabling two signatures to be printed.

In addition, this element is used in conjunction with the bookFoldPrintingSheets element to enable the given WordprocessingML document to be printed such that the series of signatures printed can be folded and bound to create a booklet. Specifically, the signatures can be placed back to back, with top the bottom of each sheet aligned, and folded such that a booklet is created. *end example*]

[*Note*: This element could also be leveraged by the hosting application to notify the application to display two pages per sheets within its user interface to allow for a WYSIWYG user experience. *end note*]

This element's content model is defined by the common boolean property definition in §17.17.4.

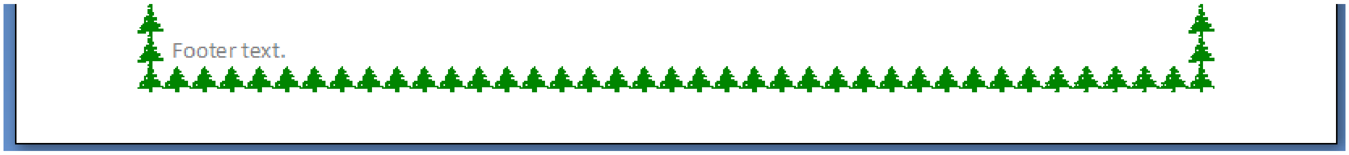
17.15.1.14 bordersDoNotSurroundFooter (Page Border Excludes Footer)

This element specifies that a given WordprocessingML document's page border specified using the pgBorders element (§17.6.10) should not surround contents of the footer.

If this element is omitted, then the page border shall not exclude the footer on the page. As well, this element shall be ignored if the pgBorders element has an offsetFrom attribute which is not equal to text.

[*Note*: If the pgBorders element has a offsetFrom attribute equal to page, the bordersDoNotSurroundFooter element is ignored as specifying the pgBorders element with a offsetFrom attribute equal to page is to specify that the positioning of borders within the document shall be calculated relative to the edge of the page and therefore irrespective of document content in the footer. *end note*]

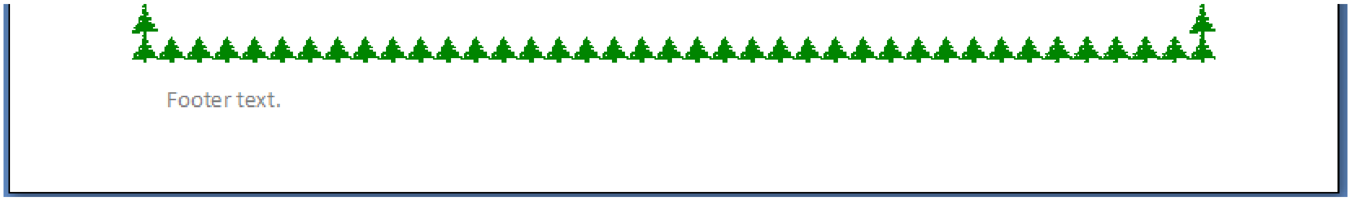
[*Example*: Consider the following page in a WordprocessingML document:



If this WordprocessingML document is modified to leverage the behavior enabled by this element, this setting would be specified using the following WordprocessingML fragment in the document settings:

```
<w:bordersDoNotSurroundFooter w:val="true"/>
```

The bordersDoNotSurroundFooter element's val attribute is equal to true specifying that the page border shall not surround the text extents of the footer, as follows:



end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

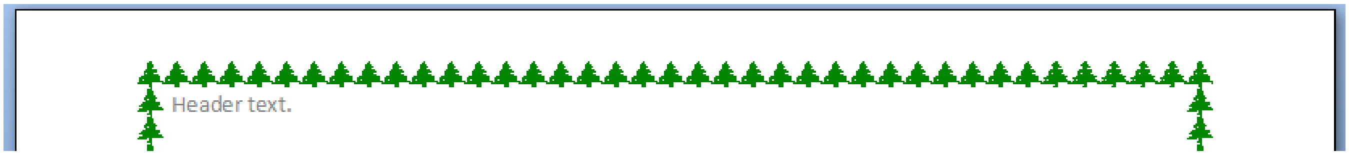
17.15.1.15 bordersDoNotSurroundHeader (Page Border Excludes Header)

This element specifies that a given WordprocessingML document's page border specified using the pgBorders element (§17.6.10) should not surround contents of the header.

If this element is omitted, then the page border shall not exclude the header on the page. As well, this element shall be ignored if the pgBorders element has a offsetFrom attribute which is not equal to `text`.

[*Note*: If the pgBorders element has a offsetFrom attribute equal to `page`, the bordersDoNotSurroundHeader element is ignored as specifying the pgBorders element with a offsetFrom attribute equal to page is to specify that the positioning of borders within the document shall be calculated relative to the edge of the page and therefore irrespective of document content in the header. *end note*]

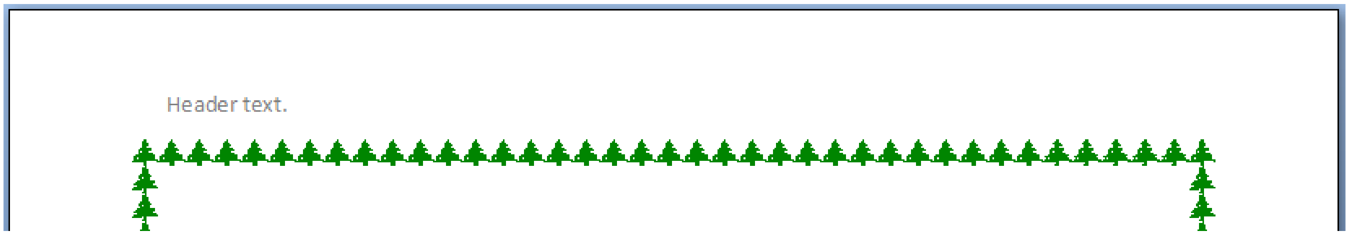
[*Example*: Consider the following page in a WordprocessingML document:



If this WordprocessingML document is modified to leverage the behavior enabled by this element, this setting would be specified using the following WordprocessingML fragment in the document settings:

```
<w:bordersDoNotSurroundHeader w:val="true"/>
```

The bordersDoNotSurroundHeader element's val attribute is equal to `true` specifying that the page border shall not surround the text extents of the header, as follows:



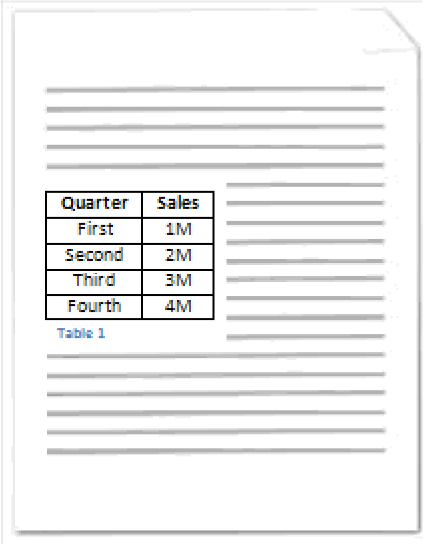
end example]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.16 caption (Single Caption Type Definition)

This element specifies the contents and positioning for captions which can be used to automatically label objects in a WordprocessingML document. A *caption* is a string that labels an object included in a WordprocessingML document, and typically consists of a string plus a field which numbers this item within a collection of similar objects.

[*Example*: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a caption:



Quarter	Sales
First	1M
Second	2M
Third	3M
Fourth	4M

In this diagram, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of the string `Table` followed by a field whose result is a decimal number. The settings which automatically produced this form of caption are specified using the following WordprocessingML fragment:

```
<w:captions>
  <w:caption w:name="Table" w:pos="below" w:numFmt="decimal" />
</w:captions>
```

The caption element specifies the parameters for the resulting caption to be used to automatically label content within the WordprocessingML document. Specifically, the name and numFmt attributes specify that captions of this caption type inserted in the given WordprocessingML document shall consist of the string `Table` followed by an incrementing decimal number field. In addition, the pos attribute specifies that these captions shall be placed below the object they are used to label.

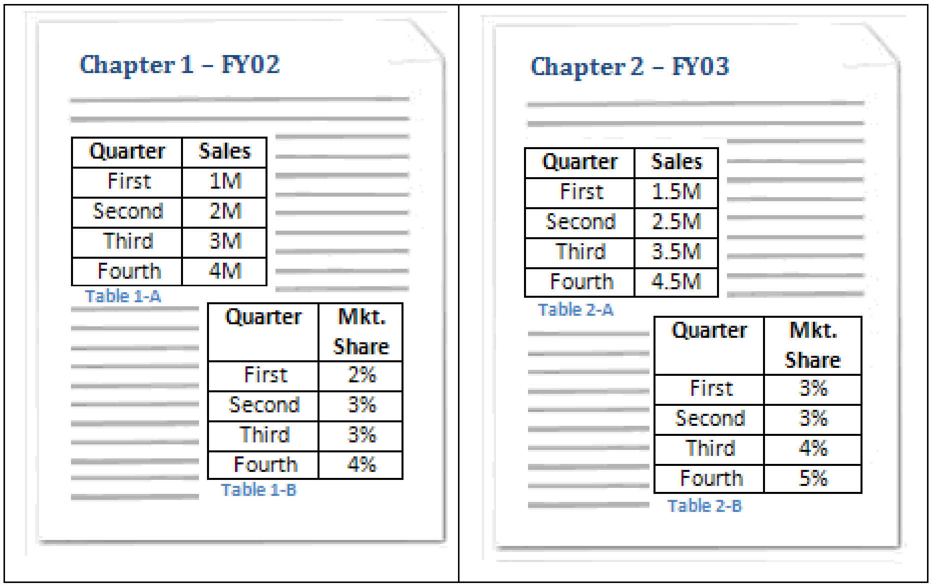
WordprocessingML is designed such that the caption element can be used in conjunction with applications to provide a dynamic captioning experience. In other words, an application can use the WordprocessingML in the example above to automatically insert a caption consisting of the string `Table` followed by an incrementing decimal number field below tables when tables are inserted into a WordprocessingML document as defined by the autoCaption element (§17.15.1.7). *end example*]

Attributes	Description
chapNum (Include Chapter Number in Field for Caption)	Specifies whether or not to display numbering associated with the most recent chapter heading in the WordprocessingML document within the caption field. A *chapter heading* is a paragraph of text within a WordprocessingML document that is formatted with a style that has been specified by the heading attribute to demarcate chapters in documents. Only a style with its styleID attribute equal to `Heading1`, `Heading2`, `Heading3`,

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
Heading4, Heading5, Heading6, Heading7, Heading8, or Heading9 can be specified as the style used to demarcate chapters in a document. The choice of which of these heading levels shall be used to determine the current chapter number is defined by the value of the corresponding heading attribute. [<i>Example</i>: Heading1 is used as the chapter heading when chapNum is true and heading is 1. <i>end example</i>]

If this attribute is omitted, then chapter numbers shall not be included in the resulting caption.

[<i>Example</i>: Consider the diagram below:



This diagram depicts a WordprocessingML document containing two chapters, each containing two tables labeled with captions. The Heading 2 style has been associated with chapter headings and applied to the strings: Chapter 1 - FY02 and Chapter 2 - FY03 in this document.

Specifically, the style used to demarcate chapters, is the style with a styleID attribute equal to Heading2 as specified by the heading attribute value of 2 in the WordprocessingML for this caption, defined as follows:

<w:caption w:name="Table" w:pos="below" w:chapNum="true"
w:heading="2" w:numFmt="upperLetter" w:sep="hyphen" />

The chapNum attribute has a value of true, specifying that the captions used to label the tables within this document contains a symbol corresponding to the one-based index of the chapter in which it is contained.

This can be seen in that the captions in Chapter 1 contain a 1, while the captions in Chapter 2 contain a 2, each corresponding with their respective chapter number. <i>end</i>
</td></tr>
</table>

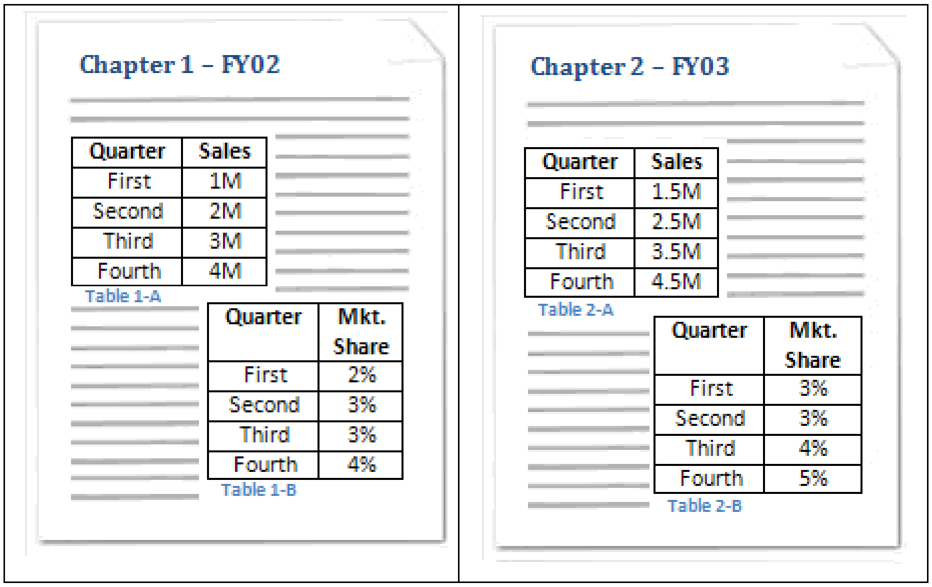
<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td><em>example</em>]

The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).</td></tr>
<tr><td>heading (Style for Chapter Headings)</td><td>Specifies the given style that is used to demarcate chapter headings in a document.

This value is used to link the chapter headings with paragraphs with a styleID attribute as follows:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>Style with styleID of Heading1</td></tr>
<tr><td>2</td><td>Style with styleID of Heading2</td></tr>
<tr><td>3</td><td>Style with styleID of Heading3</td></tr>
<tr><td>4</td><td>Style with styleID of Heading4</td></tr>
<tr><td>5</td><td>Style with styleID of Heading5</td></tr>
<tr><td>6</td><td>Style with styleID of Heading6</td></tr>
<tr><td>7</td><td>Style with styleID of Heading7</td></tr>
<tr><td>8</td><td>Style with styleID of Heading8</td></tr>
<tr><td>9</td><td>Style with styleID of Heading9</td></tr>
<tr><td>Any other value</td><td>Application-defined. Can be ignored.</td></tr>
</table>

If this attribute is omitted, then its value shall be assumed to be 1.

[<em>Example</em>: Consider the diagram below:



Chapter 1 – FY02
<table>
<tr><th>Quarter</th><th>Sales</th></tr>
<tr><td>First</td><td>1M</td></tr>
<tr><td>Second</td><td>2M</td></tr>
<tr><td>Third</td><td>3M</td></tr>
<tr><td>Fourth</td><td>4M</td></tr>
</table>
Table 1-A
<table>
<tr><th>Quarter</th><th>Mkt. Share</th></tr>
<tr><td>First</td><td>2%</td></tr>
<tr><td>Second</td><td>3%</td></tr>
<tr><td>Third</td><td>3%</td></tr>
<tr><td>Fourth</td><td>4%</td></tr>
</table>
Table 1-B

Chapter 2 – FY03
<table>
<tr><th>Quarter</th><th>Sales</th></tr>
<tr><td>First</td><td>1.5M</td></tr>
<tr><td>Second</td><td>2.5M</td></tr>
<tr><td>Third</td><td>3.5M</td></tr>
<tr><td>Fourth</td><td>4.5M</td></tr>
</table>
Table 2-A
<table>
<tr><th>Quarter</th><th>Mkt. Share</th></tr>
<tr><td>First</td><td>3%</td></tr>
<tr><td>Second</td><td>3%</td></tr>
<tr><td>Third</td><td>4%</td></tr>
<tr><td>Fourth</td><td>5%</td></tr>
</table>
Table 2-B

This diagram depicts a WordprocessingML document containing two chapters, each</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>containing two tables labeled with captions. The Heading 2 style has been associated with chapter headings and applied to the strings: <code>Chapter 1 - FY02</code> and <code>Chapter 2 - FY03</code> in this document.

Specifically, the style used to demarcate chapter headings is the style with its styleID attribute equal to <code>Heading</code>" as specified by the heading attribute value of 2 in the WordprocessingML below.

<code>&lt;w:caption w:name="Table" w:pos="below" w:chapNum="1" w:heading="2" w:numFmt="upperLetter" w:sep="hyphen" /&gt;</code>

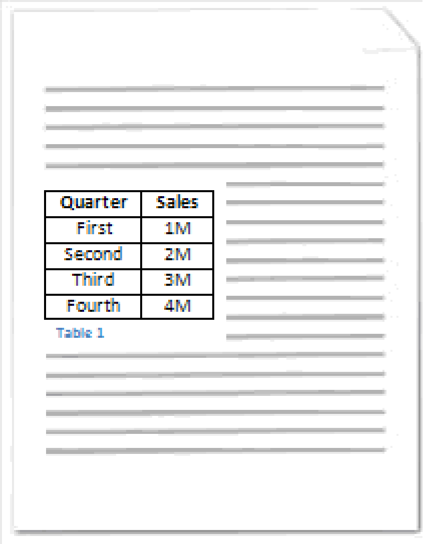
In other words, the WordprocessingML above can be used to label tables inserted in a given WordprocessingML document generated by an application with a caption consisting of: the string <code>Table</code> followed by a decimal number corresponding with the chapter number in which the table is present, a hyphen as defined in the sep attribute, and a capital English letter defined by the numFmt attribute corresponding with the given table's ordering within the current chapter. <em>end example</em>]

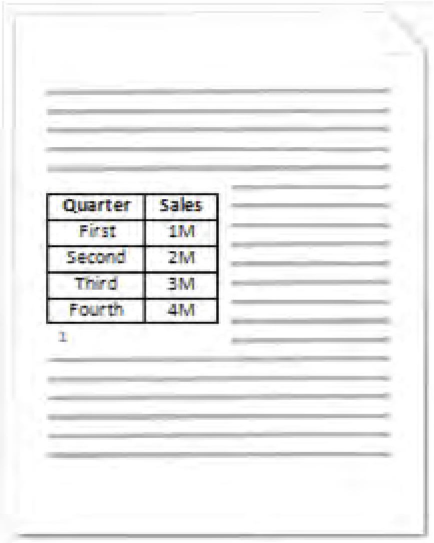
The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).</td></tr>
<tr><td>name (Caption Type Name)</td><td>Specifies the literal string component of this caption.

This value is used as follows:
• It is added to the field containing the chapter number and item number of this object when a caption is automatically added to the document.
• It is used to uniquely label this caption type, allowing it to be linked with classes of objects via the autoCaption element (§17.15.1.7)
• It can be used to label this caption type in a user interface.

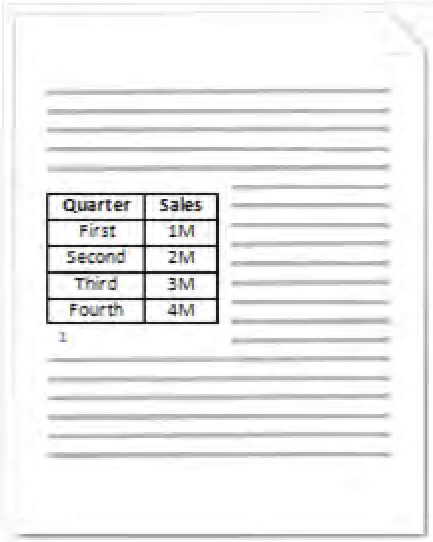
[<em>Example</em>: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a caption:</td></tr>
</table>

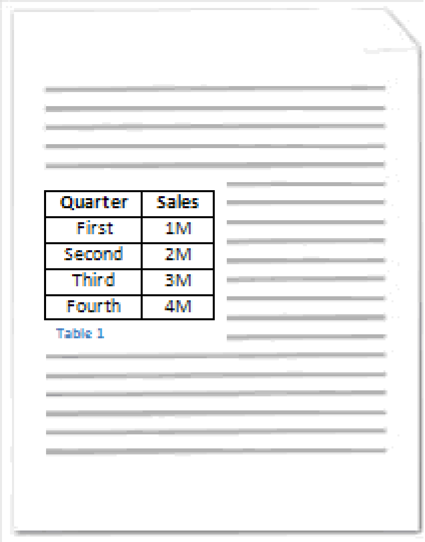
©ISO/IEC 2012 – All rights reserved

Attributes	Description
	 Quarter Sales First 1M Second 2M Third 3M Fourth 4M Table 1 In this diagram, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of the string `Table` followed by a decimal number. This caption format is specified with the following WordprocessingML: `<w:caption w:name="Table" w:pos="below" w:numFmt="decimal" />` Specifically, the name attribute specifies that the first part of the string that comprises the give caption shall consist of the string `Table`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
noLabel (Do Not Include Name In Caption)	Specifies if the string specified in the name attribute shall be included in the resulting caption when it is automatically added to the document. If set to true, then the label text in the name attribute is omitted when adding the caption. If this attribute is omitted, then the name shall be added to the caption. [*Example*: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a caption:

Attributes	Description
	 Quarter Sales First 1M Second 2M Third 3M Fourth 4M 1 In this diagram, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of only a decimal number. This caption format is specified using the following WordprocessingML: `<w:caption w:name="Custom" w:pos="below" w:noLabel="true" w:numFmt="decimal" />` Here, the noLabel attribute is equal to `true` specifying that when this caption format is automatically added, it must not include the label. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
numFmt (Caption Numbering Format)	Specifies the format of the numbering which shall be included in an automatically generated caption to specify the index of this item in that collection (within the current chapter if chapNum is specified, or within the current document story). If this attribute is omitted, then its default value shall be assumed to be `decimal`. [*Example*: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a caption:

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	 Quarter Sales First 1M Second 2M Third 3M Fourth 4M 1 In this example, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of only a decimal number. This caption format is specified using the following WordprocessingML: `<w:caption w:name="Custom" w:pos="below" w:noLabel="true" w:numFmt="decimal" />` Here, the numFmt attribute is equal to `decimal`, specifying that a decimal number shall be included in the table caption when it is automatically inserted. *End Example*] The possible values for this attribute are defined by the ST_NumberFormat simple type (§17.18.59).
pos (Automatic Caption Placement)	Specifies how an automatically inserted caption shall be positioned relative to the object that it is captioning. If this attribute is omitted, then the default value shall be `below`. [*Example*: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a *caption*.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

<table>
<tr><th>Quarter</th><th>Sales</th></tr>
<tr><td>First</td><td>1M</td></tr>
<tr><td>Second</td><td>2M</td></tr>
<tr><td>Third</td><td>3M</td></tr>
<tr><td>Fourth</td><td>4M</td></tr>
</table>

In this diagram, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of the string Table followed by a decimal number.

This caption format is specified using the following WordprocessingML:

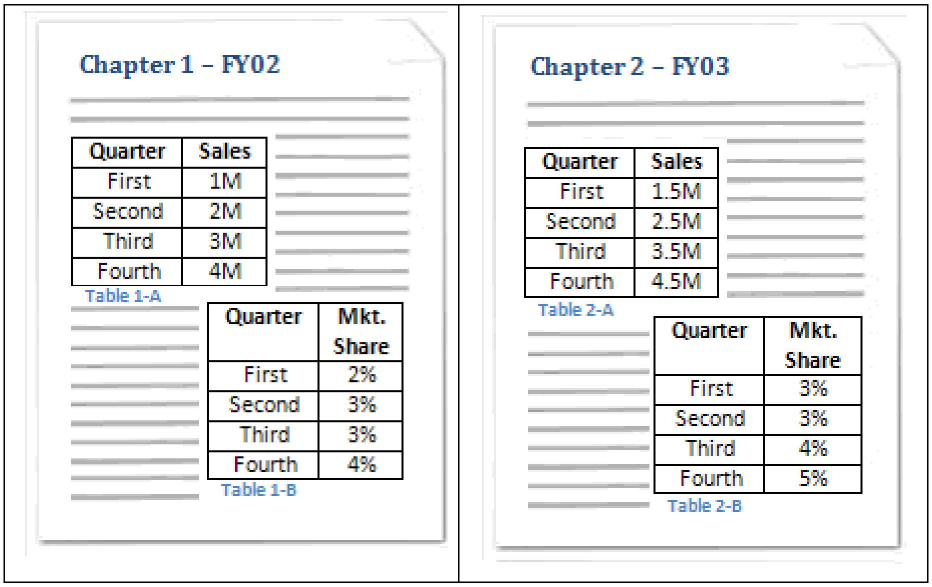
<w:caption w:name="Table" w:pos="below" w:numFmt="decimal" />

The pos attribute specifies that the given caption shall be placed below the object it is labelling. <i>end example</i>]

The possible values for this attribute are defined by the ST_CaptionPos simple type (§17.18.5).
</td></tr>
<tr><td>sep (Chapter Number/Item Index Separator)</td><td>
Specifies the character which shall be used to separate the chapter number used in this caption from the caption item numbering. A caption format consists of three components:
<ul>
<li>The (optional) literal string</li>
<li>The (optional) chapter number</li>
<li>The index of this caption within the chapter/document</li>
</ul>
When the latter two items are both present, they are delimited using the chapter separator specified by this attribute.

If this attribute is omitted, then its default value shall be hyphen. If the chapter number is not part of the caption format, then this parameter shall be ignored.

[<i>Example</i>: Consider the diagram below:
</td></tr>
</table>

Attributes	Description
	 Chapter 1 – FY02 Quarter | Sales First | 1M Second | 2M Third | 3M Fourth | 4M Table 1-A Quarter | Mkt. Share First | 2% Second | 3% Third | 3% Fourth | 4% Table 1-B Chapter 2 – FY03 Quarter | Sales First | 1.5M Second | 2.5M Third | 3.5M Fourth | 4.5M Table 2-A Quarter | Mkt. Share First | 3% Second | 3% Third | 4% Fourth | 5% Table 2-B This diagram depicts a WordprocessingML document containing two chapters, each containing two tables labeled with captions. The Heading 2 style has been associated with chapter headings and applied to the strings: `Chapter 1 - FY02` and `Chapter 2 - FY03` in this document. Specifically, the style used to demarcate chapter headings is the style with a styleID attribute equal to `Heading2` as specified by the heading attribute value of 2 in the WordprocessingML below. `<w:caption w:name="Table" w:pos="below" w:chapNum="1" w:heading="2" w:numFmt="upperLetter" w:sep="hyphen" />` The sep attribute value of hyphen specifies that the chapter number and caption index must be separated by a hyphen character when displayed in the document. *end example*] The possible values for this attribute are defined by the ST_ChapterSep simple type (§17.18.6).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Caption) is located in §A.1. *end note*]

17.15.1.17 captions (Caption Settings)

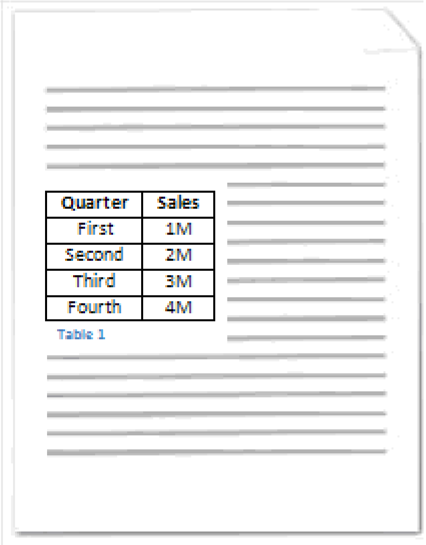
This element specifies the presence of information about captions in a given WordprocessingML document. This information is divided into two components:

- The child element caption defines the format for a single type of caption to be automatically added to the document.

- The child element autoCaptions defines the types of objects to which a caption format shall automatically be applied.

This information should be used to determine the captions which are automatically added to objects when they are inserted into a WordprocessingML document. [*Note*: This setting is typically ignored unless it is specified in an application's default template. *end note*]

[*Example*: Consider the diagram below illustrating a WordprocessingML document containing a table that has been labeled with a caption:



Quarter	Sales
First	1M
Second	2M
Third	3M
Fourth	4M

In this diagram, the table contained in the WordprocessingML document has been labeled by inserting a caption below the table consisting of the string `Table` followed by a decimal number. This automatically inserted caption format is specified using the following WordprocessingML:

```
<w:captions>
  <w:caption w:name="Table" w:pos="below" w:numFmt="decimal" />
</w:captions>
```

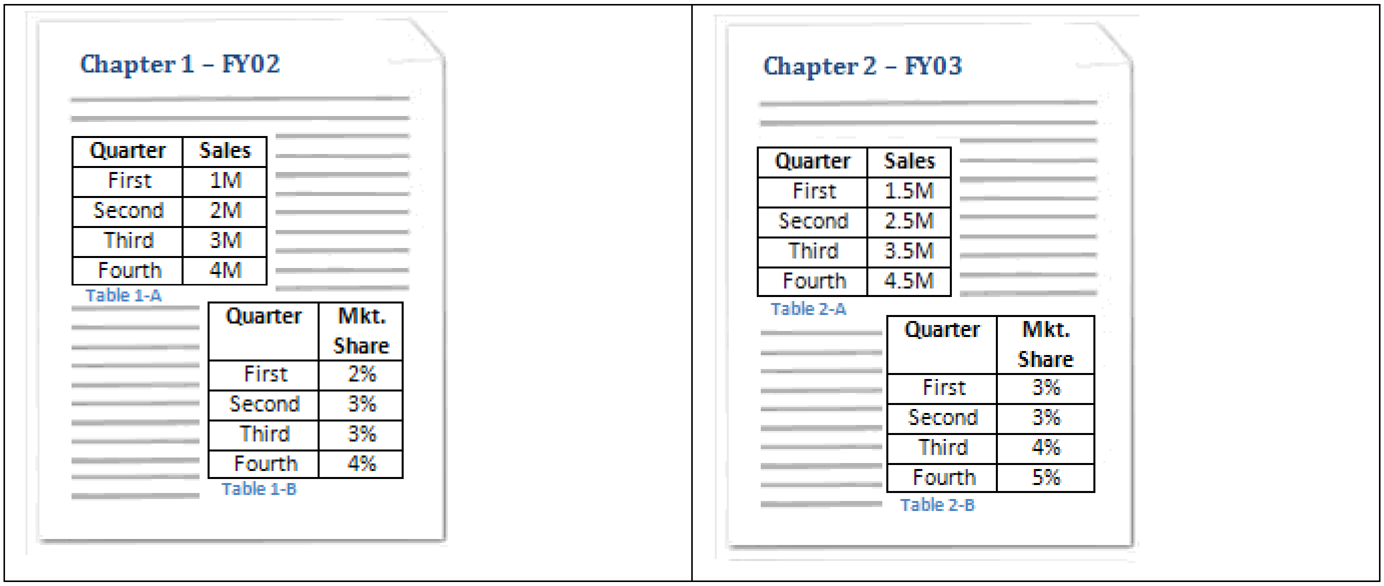
Here, the captions element specifies the presence of one or more caption formats in a given WordprocessingML document with its child element caption. Specifically, the child element caption specifies a single type of caption to be used within the WordprocessingML document. *end example*]

Captioning leverages fields (§17.16.5) to label objects with reference to either:

- Other captioned objects within a given document
- Other captioned objects within the same chapter in a given document (when chapter numbers are added by specifiying the chapNum attribute on the caption type.

  A *chapter* is a section of text within a WordprocessingML document that is preceded by content with a style that has been specified by to demarcate chapters in documents. Only one style can be specified as the style used for a single caption type to demarcate chapters in a document. A chapter ends immediately above the next instance of content with the style used to demarcate chapters.

[*Example*: Consider the diagram below:



Chapter 1 – FY02

Quarter	Sales
First	1M
Second	2M
Third	3M
Fourth	4M

Table 1-A

Quarter	Mkt. Share
First	2%
Second	3%
Third	3%
Fourth	4%

Table 1-B

Chapter 2 – FY03

Quarter	Sales
First	1.5M
Second	2.5M
Third	3.5M
Fourth	4.5M

Table 2-A

Quarter	Mkt. Share
First	3%
Second	3%
Third	4%
Fourth	5%

Table 2-B

This diagram depicts a WordprocessingML document containing two chapters, each containing two tables labeled with captions. The style associated with chapter demarcation has been applied to the strings: `Chapter 1 - FY02` and `Chapter 2 - FY03` in this document. Specifically, the style used to demarcate chapters is the style with its styleID attribute equal to `Heading2` as specified by the heading attribute value of 2 in the WordprocessingML for the caption format:

```
<w:caption w:name="Table" w:pos="below" w:chapNum="1" w:heading="2"
w:numFmt="upperLetter" w:sep="hyphen" />
```

In other words, the WordprocessingML above can be used to label objects (in this case, tables) inserted in a given WordprocessingML document generated by an application with a caption consisting of: the string Table followed by a decimal number corresponding with the chapter number in which the table is present, a hyphen, and a capital English letter corresponding with the given table's index within the given chapter. *end example*]

[*Note*: WordprocessingML is designed such that the caption element can be used in conjunction with applications to provide a dynamic captioning experience. In other words, an application can use the WordprocessingML in the example above to automatically insert a caption consisting of the string `Table` followed by an incrementing decimal number field below tables when tables are inserted into a WordprocessingML document as defined by the autoCaption element (§17.15.1.7). *end note*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Captions) is located in §A.1. *end note*]

17.15.1.18 characterSpacingControl (Character-Level Whitespace Compression)

This element specifies how full-width characters in the current WordprocessingML document should be compressed to remove additional whitespace when the contents of this document are displayed, specifically by specifying the set(s) of characters which can be compressed to remove additional whitespace. [*Note*: The behavior of this element is functionally identical to the CSS text-justify-trim property. *end note*]

If this element is omitted, then the default value shall be `dontCompress`.

[*Example*: Consider the WordprocessingML below:

```
<w:characterSpacingControl w:val="doNotCompress" />
```

The characterSpacingControl element has a val attribute value of `doNotCompress`, which specifies that no character compression shall be applied to any character when the document is displayed. *end example*]

Attributes	Description
val (Value)	Specifies the set(s) of characters which should be compressed when the contents of this document are displayed. [*Example*: Consider a WordprocessingML document for which only full-width punctuation characters must have their whitespace compression applied. This requirement would be specified using the following WordprocessingML: `<w:characterSpacingControl w:val="compressPunctuation"/>` The val attribute value of `compressPunctuation` specifies that character compression must be applied to full-width punctuation characters only when the document is displayed. *end example*] The possible values for this attribute are defined by the ST_CharacterSpacing simple type (§17.18.7).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_CharacterSpacing) is located in §A.1. *end note*]

17.15.1.19 clickAndTypeStyle (Paragraph Style Applied to Automatically Generated Paragraphs)

This element specifies the paragraph style, specified using the style element, which shall be applied to paragraphs which are automatically created when text is inserted into a WordprocessingML document in an area of the document that has no other style associated with it. This style is referenced via the val attribute, which stores the style ID of the style (stored in the styleId attribute on the style definition).

[*Guidance*: Consider a WordprocessingML document opened in an application that allows users to place their cursor anywhere within the document editing canvas and enter text. The clickAndTypeStyle element should be used to specify the paragraph style to be associated with the paragraph of text entered after a user places their cursor somewhere in the blank document that results in the generation of new paragraphs. *end guidance*]

If this element is omitted, then the default paragraph style (the paragraph style whose default attribute is set to true), shall be used for automatically generated paragraphs. If the style whose styleId is specified using the val attribute is not a paragraph style or does not exist in the document, then the default paragraph style shall be used instead.

[*Example*: Consider a WordprocessingML document that has specified that paragraphs which are automatically created when text is inserted in a given area of the document which has no other style associated with it must be associated with the paragraph style that has a styleId equal to `BalloonText`.

This is accomplished by specifying a clickAndTypeStyle element with a val attribute equal to the value of the ID of the desired style. This constraint would be specified using the following WordprocessingML:

```
<w:clickAndTypeStyle w:val="BalloonText" />
```

The corresponding style in the styles part would be defined as follows:

```
<w:style w:type="paragraph" w:styleId="BalloonText">
  …
</w:style>
```

The clickAndTypeStyle element specifies the use of the paragraph style with the style ID of `BalloonText`. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`

Attributes	Description
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.15.1.20 clrSchemeMapping (Theme Color Mappings)

This element specifies the theme color, stored in the document's Theme part to which the value of this theme color shall be mapped. This mapping enables multiple theme colors to be chained together.

[*Example*: Consider a WordprocessingML document that must have the theme color value `background1` mapped to the theme color `light1` as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:clrSchemeMapping w:bg1="light1" />
```

The clrSchemeMapping element's attribute background1 has a value of `light1`, specifying that theme color value `background1` must be mapped to the theme color light1. *end example*]

Attributes	Description
accent1 (Accent 1 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the `accent1` theme color shall be used. [*Example*: Consider a WordprocessingML document that must have references to the theme color `accent1` mapped to the theme color lt1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:accent1="light1" />` The accent1 attribute has a value of `light1`, specifying that uses of the theme color value `accent1` must be mapped to the theme color lt1. *end example*] The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).
accent2 (Accent 2 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the `accent2` theme color shall be used.

Attributes	Description
	[*Example*: Consider a WordprocessingML document that must have the references to the theme color `accent2` mapped to the theme color hlink as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:accent2="hyperlink" />` The accent2 attribute has a value of `hyperlink`, specifying that uses of the theme color value `accent2` must be mapped to the theme color hlink. *end example*] The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).
accent3 (Accent3 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the `accent3` theme color shall be used. [*Example*: Consider a WordprocessingML document that must have references to the theme color `accent3` mapped to the theme color dk1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:accent3="dark1" />` The accent3 attribute has a value of `dark1`, specifying that uses of the theme color value `accent3` must be mapped to the theme color dk1. *end example*] The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).
accent4 (Accent4 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the `accent4` theme color shall be used. [*Example*: Consider a WordprocessingML document that must have references to the theme color `accent4` mapped to the theme color dk2 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:accent4="dark2" />` The accent4 attribute has a value of `dark2`, specifying that uses of the theme color value `accent3` must be mapped to the theme color dk2. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).
accent5 (Accent5 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the accent5 theme color shall be used. [*Example*: Consider a WordprocessingML document that must have references to the theme color accent5 mapped to the theme color accent1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:accent5="accent1" />` The accent5 attribute has a value of accent1, specifying that uses of the theme color value accent5 must be mapped to the theme color accent1. *end example*] The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).
accent6 (Accent6 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the accent6 theme color shall be used. [*Example*: Consider a WordprocessingML document that must have references to the theme color accent6 mapped to the theme color accent1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:accent6="accent1" />` The accent6 attribute has a value of accent1, specifying that uses of the theme color value accent6 must be mapped to the theme color accent1. *end example*] The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).
bg1 (Background 1 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the light1 theme color shall be used. [*Example*: Consider a WordprocessingML document that must have references to the theme color bg1 mapped to the theme color lt2 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings:

Attributes	Description
	`<w:clrSchemeMapping w:bg1="light2" />` The bg1 attribute has a value of `light2`, specifying that uses of the theme color value `bg1` must be mapped to the theme color lt2. *end example*] The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).
bg2 (Background 2 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the `light2` theme color shall be used. [*Example*: Consider a WordprocessingML document that must have references to the theme color `bg2` mapped to the theme color dk1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:bg2="dark1" />` The bg2 attribute has a value of `dark1`, specifying that uses of the theme color value `bg2` must be mapped to the theme color dk1. *end example*] The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).
followedHyperlink (Followed Hyperlink Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the `followedHyperlink` theme color shall be used. [*Example*: Consider a WordprocessingML document that must have references to the theme color `followedHyperlink` mapped to the theme color hyperlink as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:followedHyperlink="hyperlink" />` The followedHyperlink attribute has a value of `hyperlink`, specifying that uses of the theme color value `followedHyperlink` must be mapped to the theme color hyperlink. *end example*] The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).
hyperlink (Hyperlink Theme	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content.

Attributes	Description
Color Mapping)	If this attribute is omitted, then the hyperlink theme color shall be used. [*Example*: Consider a WordprocessingML document that must have references to the theme color hyperlink mapped to the theme color accent1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:hyperlink="accent1" />` The hyperlink attribute has a value of accent1, specifying that uses of the theme color value hyperlink must be mapped to the theme color accent1. *end example*] The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).
t1 (Text 1 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the t1 theme color shall be used. [*Example*: Consider a WordprocessingML document that must have references to the theme color t1 mapped to the theme color lt1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:t1="light1" />` The t1 attribute has a value of light1, specifying that uses of the theme color value t1 must be mapped to the theme color lt1. *end example*] The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).
t2 (Text 2 Theme Color Mapping)	Specifies the theme color in the document's theme part which shall be used in place of this color when it is referenced by document content. If this attribute is omitted, then the t2 theme color shall be used. [*Example*: Consider a WordprocessingML document that must have references to the theme color t2 mapped to the theme color dk1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings: `<w:clrSchemeMapping w:t2="dark1" />` The t2 attribute has a value of dark1, specifying that uses of the theme color value t2

Attributes	Description
	must be mapped to the theme color dk1. *end example*] The possible values for this attribute are defined by the ST_WmlColorSchemeIndex simple type (§17.18.103).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ColorSchemeMapping) is located in §A.1. *end note*]

17.15.1.21 compat (Compatibility Settings)

This element specifies a set of optional compatibility options for the current document.

[*Example*: Consider a WordprocessingML document with a series of compatibility settings:

```
<w:settings>
  <w:compat>
    …
  </w:compat>
</w:settings>
```

The compat element specifies the set of compatibility settings for a document. *end example*]

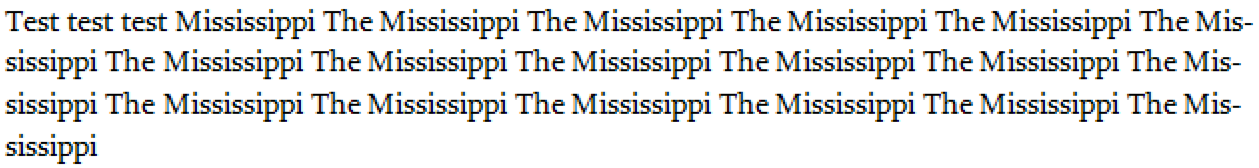
[*Note*: The W3C XML Schema definition of this element's content model (CT_Compat) is located in §A.1. *end note*]

17.15.1.22 consecutiveHyphenLimit (Maximum Number of Consecutively Hyphenated Lines)

This element specifies the maximum number of consecutive lines of text that can end with a hyphen when the contents of this document are displayed. Once this limit has been reached, the following line shall not be hyphenated regardless of whether or not it meets the criteria needed for hyphenation.

If this element is omitted or has its val attribute equal to 0, the given WordprocessingML document shall have no limit on the number of consecutive lines of text that can end with a hyphen.

[*Example*: Consider a WordprocessingML document which should automatically be hyphenated. If the contents of this document result in hyphens appearing on every line in the document, as follows:

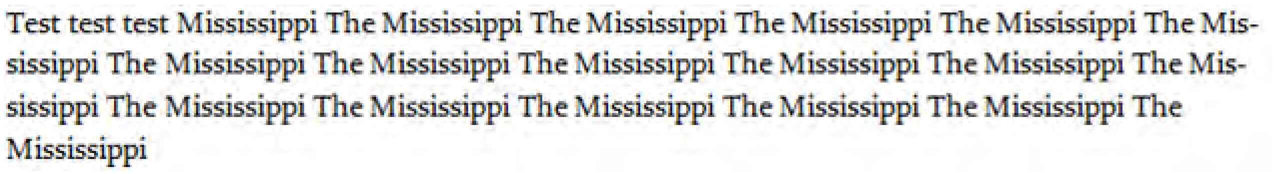


Test test test Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The Mis-
sissippi The Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The Mis-
sissippi The Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The Mis-
sissippi

This output might be undesirable. If the document shall have a maximum of two consecutive hyphens, this requirement is specified using the following WordprocessingML in the document settings:

```
<w:consecutiveHyphenLimit w:val="2" />
```

The consecutiveHyphenLimit element's val attribute has a value of 2 specifying that a maximum of two hyphens should be allowed, limiting the hyphenation output like this:



Test test test Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The Mis-
sissippi The Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The Mis-
sissippi The Mississippi The Mississippi The Mississippi The Mississippi The Mississippi The
Mississippi

end example]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.15.1.23 decimalSymbol (Radix Point for Field Code Evaluation)

This element specifies the character that shall be interpreted as the radix point when evaluating the contents of all fields in the current document.

[*Rationale*: When evaluating field instructions based on the contents of the current document, it is necessary to know the character which must be treated as the radix point in order to prevent changes to the calculation of the same field instructions based on the current user's locale. This element stores the radix point which must be used to evaluate fields in the contents of this document, irrespective of the locale of the application loading the file. *end rationale*]

If this element is omitted, the application shall use the default radix point of its current locale setting to evaluate field instructions. If this element's attribute value is more than a single character, then the document is non-conformant.

[*Example*: Consider a WordprocessingML document which should use the comma character as the radix point for all field instructions. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:decimalSymbol w:val="," />
```

The decimalSymbol element's val attribute has a value of `,` specifying that the comma character must be interpreted as the radix point.

For instance, the string `12.345,00` would be interpreted as a numeric value of twelve thousand three hundred and forty five. If the decimalSymbol was a period, the same string would be twelve and three hundred and forty five thousandths. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.15.1.24 defaultTableStyle (Default Table Style for Newly Inserted Tables)

This element specifies the table style which shall automatically be applied to the table properties of tables added to this document by an application. Note that it does not change the table style applied to tables which do not reference a style, instead, it automatically applies the style to that table via the tblStyle element (§17.4.62). This link is made by referencing the styleId attribute value of the table style which shall be used to format newly inserted tables.

If this element is omitted, then no table style shall automatically be applied to inserted tables (therefore inheriting the default table style). If the referenced style is not present or not a table style, then no table style shall automatically be applied to inserted tables.

[*Example*: Consider a WordprocessingML document which should use the LightShading-Accent3 style. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:defaultTableStyle w:val="LightShading-Accent3" />
```

The corresponding table style must therefore exist in the styles part:

```
<w:style w:type="table" w:styleId="LightShading-Accent3">
  …
</w:style>
```

The defaultTableStyle element's val attribute has a value of `LightShading-Accent3` specifying that that style is applied automatically to newly inserted tables. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.15.1.25 defaultTabStop (Distance Between Automatic Tab Stops)

This element specifies the value which shall be used as the multiplier to generate automatic tab stops in this document. *Automatic tab stops* refer to the tab stop locations which occur after all custom tab stops in the current paragraph have been surpassed.

If this element is omitted, then automatic tab stops should be generated at 720 twentieths of a point (0.5") intervals across the displayed page.

[*Example*: Consider a WordprocessingML document which should have automatic tab stops every 360 twentieths of a point (0.25 inches). This requirement is specified using the following WordprocessingML in the document settings:

```
<w:defaultTabStop w:val="360" />
```

The defaultTabStop element's val attribute has a value of 360 specifying that automatic tab stops shall occur every 1/4th of an inch across the page.

If a custom tab stop was located at 2.28", then the next three automatic tab stops would be at 2.5", 2.75" and 3.0" (the next three multiples of the default tab stop value). *end example*]

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<… w:val="720" />` The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TwipsMeasure) is located in §A.1. *end note*]

17.15.1.26 displayBackgroundShape (Display Background Objects When Displaying Document)

This element specifies whether the images and colors defined in the document's background using the background element (§17.2.1) shall be displayed when the document is displayed in print layout view as specified in the view element (§17.15.1.92).

If this element is omitted, then background shapes shall not be displayed when the document is displayed in print layout view.

[*Example*: Consider a WordprocessingML document that has a turquoise background specified for all pages and is being displayed in page layout view, as follows:

If the document's background should not be displayed, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:displayBackgroundShape w:val="true" />
```

The resulting document would display the background in page layout view:

end example]

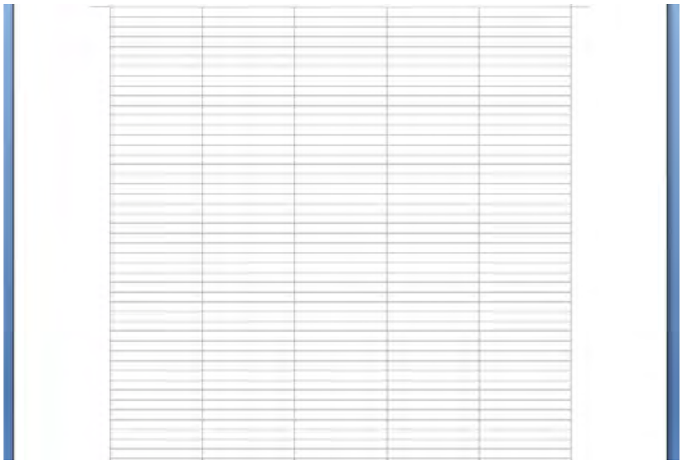
This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.27 displayHorizontalDrawingGridEvery (Distance between Horizontal Gridlines)

This element specifies the number of horizontal grid units defined using the drawingGridHorizontalSpacing element (§17.15.1.45) which shall be allowed between subsequent visible horizontal drawing grid lines in this document, if gridlines are being shown. [*Note*: The display of gridlines is an application-level setting not specified in ISO/IEC 29500. *end note*] The *drawing grid* is a grid which can be used by applications to help position floating objects in the document.

If this element is omitted, then gridlines shall be displayed for each horizontal grid unit.

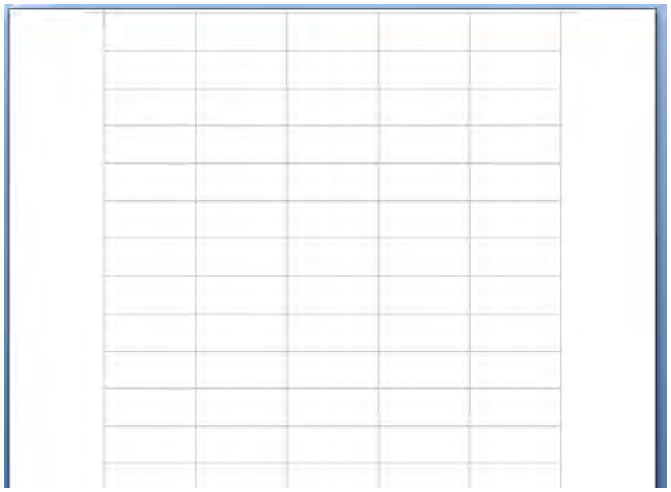
[*Example*: Consider the image below illustrating a WordprocessingML document in which all horizontal grid units are visible (the default setting):



If the gridlines in this document shall only be displayed for every 4th horizontal drawing gridline, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:displayHorizontalDrawingGridEvery w:val="4" />
```

The resulting grid would look like the following:



The displayHorizontalDrawingGridEvery element has its val attribute equal to 4, therefore every fourth gridline is displayed in the document when the drawing grid is turned on. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

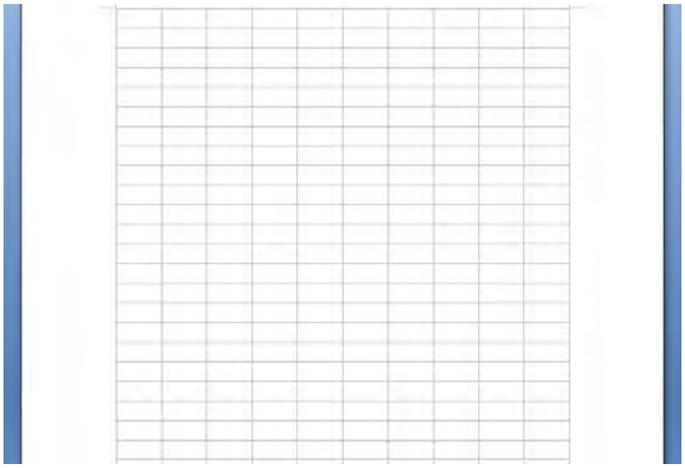
[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.15.1.28 displayVerticalDrawingGridEvery (Distance between Vertical Gridlines)

This element specifies the number of vertical grid units defined using the drawingGridVerticalSpacing element (§17.15.1.47) which shall be allowed between subsequent vertical gridlines in this document, if gridlines are being shown. [*Note*: The display of gridlines is an application-level setting not specified in ISO/IEC 29500. *end* *note*] The *drawing grid* is a grid which can be used by applications to help position floating objects in the document.

If this element is omitted, then vertical gridlines shall not be displayed.

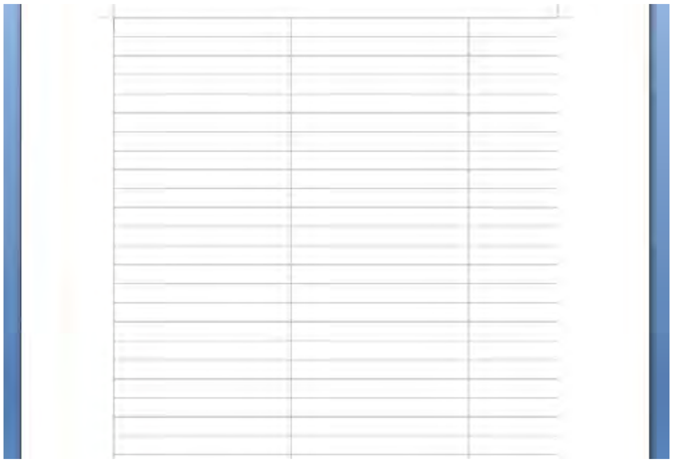
[*Example*: Consider the image below illustrating a WordprocessingML document in which all vertical grid units are visible (the default setting):



If the vertical drawing gridlines in this document must only be displayed for every 4th gridline, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:displayVerticalDrawingGridEvery w:val="4" />
```

The resulting grid would look like the following:



The displayVerticalDrawingGridEvery element has its val attribute equal to 4, therefore every fourth vertical gridline is displayed in the document when the drawing grid is turned on. *end example*]

Attributes	Description
val (Decimal	Specifies that the contents of this attribute contains a decimal number.

Attributes	Description
Number Value)	The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.15.1.29 documentProtection (Document Editing Restrictions)

This element specifies the set of document protection restrictions which have been applied to the contents of a WordprocessingML document. These restrictions should be enforced by applications editing this document when the enforcement attribute is turned on, and ignored (but persisted) otherwise. *Document protection* is a set of restrictions used to prevent unintentional changes to all or part of a WordprocessingML document. [*Note*: This protection does not encrypt the document, and malicious applications might circumvent its use. This protection is not intended as a security feature. *end note*]

If this element is omitted, then no protection shall be applied to this document. When a password is to be hashed and stored in this element, it shall be hashed as defined below, starting from a UTF-16LE encoded string value. If there is a leading BOM character (U+FEFF) in the encoded password it is removed before hash calculation.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>algorithmName (Cryptographic Algorithm Name)</td><td>Specifies the specific cryptographic hashing algorithm which shall be used along with the salt attribute and input password in order to compute the hash value.
The following values are reserved:
<table>
<tr><th>Value</th><th>Algorithm</th></tr>
<tr><td>MD2</td><td>Specifies that the MD2 algorithm, as defined by RFC 1319, shall be used.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>[<i>Note</i>: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. <i>end note</i>]</td></tr>
<tr><td><code>MD4</code></td><td>Specifies that the MD4 algorithm, as defined by RFC 1320, shall be used.

[<i>Note</i>: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. <i>end note</i>]</td></tr>
<tr><td><code>MD5</code></td><td>Specifies that the MD5 algorithm, as defined by RFC 1321, shall be used.

[<i>Note</i>: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. <i>end note</i>]</td></tr>
<tr><td><code>RIPEMD-128</code></td><td>Specifies that the RIPEMD-128 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.

[<i>Note</i>: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. <i>end note</i>]</td></tr>
<tr><td><code>RIPEMD-160</code></td><td>Specifies that the RIPEMD-160 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr>
<tr><td><code>SHA-1</code></td><td>Specifies that the SHA-1 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr>
<tr><td><code>SHA-256</code></td><td>Specifies that the SHA-256 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr>
<tr><td><code>SHA-384</code></td><td>Specifies that the SHA-384 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr>
<tr><td><code>SHA-512</code></td><td>Specifies that the SHA-512 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr>
<tr><td><code>WHIRLPOOL</code></td><td>Specifies that the WHIRLPOOL algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr>
</table>

[<i>Example</i>: Consider an Office Open XML document with the following information stored in one of its protection elements:

<code>&lt; … algorithmName="SHA-1"</code>
<code>hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" /&gt;</code>

The algorithmName attribute value of “<code>SHA-1</code>” specifies that the SHA-1 hashing algorithm must be used to generate a hash from the user-defined password. <i>end</i>
</td></tr>
</table>

Attributes	Description
	example] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
edit (Document Editing Restrictions)	Specifies the set of editing restrictions which shall be enforced on a given WordprocessingML document, as defined by the simple type referenced below If this attribute is omitted, the consumer shall behave as though there are no editing restrictions applied to this document; equivalent to an attribute value of none. [*Example*: Consider a WordprocessingML document that contains the following WordprocessingML specifying that hosting applications must enforce read-only protection for a given document: `<w:documentProtection w:edit="readOnly" w:enforcement="1" />` The edit attribute has a value of `readOnly` and a enforcement attribute with a value of `1`, specifying that read-only document protection must be enforced on the given document. e*nd example*] The possible values for this attribute are defined by the ST_DocProtect simple type (§17.18.18).
enforcement (Enforce Document Protection Settings)	Specifies if the document protection settings shall be enforced for a given WordprocessingML document. If the value of this element is `off`, `0`, or `false`, all the WordprocessingML pertaining to document protection is still preserved in the document, but is not enforced. If the value of this element is on, 1, or `true`, the document protection is enforced. If this attribute is omitted, then document protection settings shall not be enforced by applications. [*Example*: Consider a WordprocessingML document that contains the following WordprocessingML specifying that hosting applications must apply read-only protection for a given document: `<w:documentProtection w:edit="readOnly" w:enforcement="1" />` The enforcement attribute has a value of 1, specifying that the document protection specified must be enforced on the given document. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
formatting (Only Allow Formatting With Unlocked Styles)	Specifies if formatting restrictions are in effect for a given WordprocessingML document. This enables the document to restrict the types of styles that can exist in a given WordprocessingML document. Specifically, by setting this attribute's value equal to `true`, every style whose locked element (§17.7.4.7) has a value of `true` (or latent styles

Attributes	Description

	(§17.7.4.5) whose locked attribute is `true`) shall not be available for use in the application, nor should any direct formatting. Only styles with a locked value of `false` can be used.
	If this attribute is omitted, then no formatting restrictions shall be applied, even when document protection is enforced.
	[*Example*: Consider a WordprocessingML document that must apply formatting protection. This requirement would be specified using the following WordprocessingML in the document settings:
	`<w:documentProtection w:formatting="true" w:enforcement="true" />`
	If the following definition for a style was also present in the document:
	`<w:style w:type="paragraph" w:styleId="Heading1">`
	`  <w:name w:val="Heading 1" />`
	`  <w:locked w:val="1" />`
	`  …`
	`</w:style>`
	The formatting attribute has a value of `true` specifying that the applications must not allow the style above to be added to the WordprocessingML document. This does not preclude previous uses of that style (which must not be removed), but does prevent new uses of this style from being added. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

hashValue (Password Hash Value)	Specifies the hash value for the password stored with this document. This value shall be compared with the resulting hash value after hashing the user-supplied password using the algorithm specified by the preceding attributes and parent XML element, and if the two values match, the protection shall no longer be enforced.
	If this value is omitted, then the reservationPassword attribute shall contain the password hash for the workbook.
	[*Example*: Consider an Office Open XML document with the following information stored in one of its protection elements:
	`<… AlgorithmName="SHA-1"`
	`  hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	The hashValue attribute value of `9oN7nWkCAyEZib1RomSJTjmPpCY=` specifies that the user-supplied password must be hashed using the pre-processing defined by the parent element (if any) followed by the SHA-1 algorithm (specified via the algorithmName attribute value of `SHA-1`) and that the resulting has value must be

Attributes	Description
	9oN7nWkCAyEZib1RomSJTjmPpCY= for the protection to be disabled. *end example*] The possible values for this attribute are defined by the W3C XML Schema base64Binary datatype.

saltValue (Salt Value for Password Verifier)	Specifies the salt which was prepended to the user-supplied password before it was hashed using the hashing algorithm defined by the preceding attribute values to generate the hashValue attribute, and which shall also be prepended to the user-supplied password before attempting to generate a hash value for comparison. A *salt* is a random string which is added to a user-supplied password before it is hashed in order to prevent a malicious party from pre-calculating all possible password/hash combinations and simply using those pre-calculated values (often referred to as a "dictionary attack").
	If this attribute is omitted, then no salt shall be prepended to the user-supplied password before it is hashed for comparison with the stored hash value.
	[*Example*: Consider an Office Open XML document with the following information stored in one of its protection elements:
	`<… saltValue="ZUdHa+D8F/OAKP3I7ssUnQ=="`
	`  hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	The saltValue attribute value of ZUdHa+D8F/OAKP3I7ssUnQ== specifies that the user-supplied password must have this value prepended before it is run through the specified hashing algorithm to generate a resulting hash value for comparison. *end example*]
	The possible values for this attribute are defined by the W3C XML Schema base64Binary datatype.

spinCount (Iterations to Run Hashing Algorithm)	Specifies the number of times the hashing function shall be iteratively run (runs using each iteration's result plus a 4 byte value (0-based, little endian) containing the number of the iteration as the input for the next iteration) when attempting to compare a user-supplied password with the value stored in the hashValue attribute.
	[*Rationale*: Running the algorithm many times increases the cost of exhaustive search attacks correspondingly. Storing this value allows for the number of iterations to be increased over time to accommodate faster hardware (and hence the ability to run more iterations in less time). *end rationale*]
	[*Example*: Consider an Office Open XML document with the following information stored in one of its protection elements:
	`<… spinCount="100000"`
	`  hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	The spinCount attribute value of 100000 specifies that the hashing function must be run one hundred thousand times to generate a hash value for comparison with the

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	hashValue attribute. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocProtect) is located in §A.1. *end note*]

17.15.1.30 documentType (Document Classification)

This element specifies the classification of a given WordprocessingML document.

[*Note*: This element can be used by hosting applications to facilitate customized user interface and/or automatic formatting behaviors based on the 'type' of a given WordprocessingML document. *end note*]

If this element is omitted, then the document shall be classified as a general document.

[*Example*: Consider a set of WordprocessingML documents which should be classified as 'letters'. This classification would be specified using the following WordprocessingML in the document settings of these documents:

```
<w:documentType w:val="letter" />
```

The documentType element's val attribute is equal to `letter`, specifying that the hosting application must apply the behaviors it has specified for letters to the given WordprocessingML document. *end example*]

Attributes	Description
val (Document Classification Value)	Specifies the classification of the document based on the types defined in the referenced simple type definition. [*Example*: Consider a WordprocessingML document which should be classified as an e-mail message. This classification would be specified using the following WordprocessingML in the document settings: `<w:documentType w:val="eMail" />` The val attribute is equal to `eMail`, specifying that the hosting application can apply e-mail behaviors (if any) to this document. *end example*] The possible values for this attribute are defined by the ST_DocType simple type (§17.18.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocType) is located in §A.1. *end note*]

17.15.1.31 docVar (Single Document Variable)

This element specifies the parameters of a single document variable. A *document variable* is a storage location for arbitrary customer data in name/value pairs that is persisted in a given WordprocessingML document. Specifically, this element specifies through its name and val attributes the name and value pair for a given document variable.

[*Note*: This mechanism is maintained for legacy compatibility only, and should be avoided in favor of the custom XML data support defined in ISO/IEC 29500. *end note*]

[*Example*: Consider the following WordprocessingML fragment specifying a document variable named `example` and containing the value `example value`:

```
<w:docVars>
  <w:docVar w:name="example" w:val="example value" />
</w:docVars>
```

The docVar element defines a single document variable, named `example` using the name attribute, and assigned the value `example value` through the val attribute. *end example*]

Attributes	Description

name (Document Variable Name)	Specifies the name of the parent document variable.
	[*Example*: Consider the following WordprocessingML fragment specifying a document variable:
	`<w:docVars>`
	`  <w:docVar w:name="example name" w:val="example value" />`
	`</w:docVars>`
	The name attribute specifies that the name of the document variable is `example name`. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

val (Document Variable Value)	Specifies the value of the parent document variable.
	[*Example*: Consider the following WordprocessingML fragment specifying a document variable:
	`<w:docVars>`
	`  <w:docVar w:name="example name" w:val="Tristan Davis" />`
	`</w:docVars>`
	The val attribute specifies that the value of the document variable is `Tristan Davis`. *end example*]

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocVar) is located in §A.1. *end note*]

17.15.1.32 docVars (Document Variables)

This element specifies the presence of document variables in a WordprocessingML. A *document variable* is a storage location for arbitrary customer data in name/value pairs that is persisted in a given WordprocessingML document.

[*Note*: This mechanism is maintained for legacy compatibility only, and should be avoided in favor of the custom XML data support defined in ISO/IEC 29500. *end note*]

[*Example*: Consider the following WordprocessingML specifying three document variables:

```
<w:docVars>
  <w:docVar … />
  <w:docVar … />
  <w:docVar … />
</w:docVars>
```

The docVars element contains three child elements each defining a single document variable in this document. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocVars) is located in §A.1. *end note*]

17.15.1.33 doNotAutoCompressPictures (Do Not Automatically Compress Images)

This element specifies that pictures in this document shall not automatically be compressed when saving the document in order to reduce the overall size of the resulting WordprocessingML document.

If this element is omitted, applications can perform basic compression on images before saving the contents of the document.

[*Example*: Consider a WordprocessingML document which should never have its images compressed before they are saved. This requirement would be specified using the following WordprocessingML:

```
<w:doNotAutoCompressPictures w:val="true"/>
```

The doNotAutoCompressPictures element's val attribute has a value of true specifying that images must not be automatically compressed when the document is saved. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.34 doNotDemarcateInvalidXml (Do Not Show Visual Indicator For Invalid Custom XML Markup)

This element specifies whether a visual cue should be displayed around content contained in a WordprocessingML document which is contained with custom XML markup specified via the customXml element when an application determines that the current XML markup (or its contents) violate the constraints of the attached XML schema(s).

If this element is not present in a WordprocessingML document visual cues shall be displayed on content contained in custom XML markup in a WordprocessingML document which is considered to be invalid based on the associated XML schema(s).

[*Example*: Consider a WordprocessingML document which should show no visual indication of invalid custom XML markup. This requirement would be specified using the following WordprocessingML:

```
<w:doNotDemarcateInvalidXml w:val="true"/>
```

The doNotDemarcateInvalidXml element's val attribute has a value of true specifying the display of any visual indication of invalid custom XML markup must be suppressed for this document. *end example*]

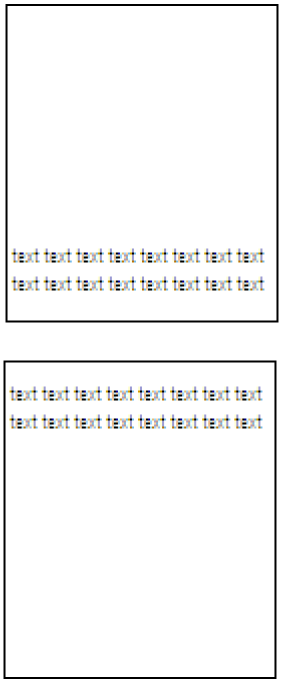
This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.35 doNotDisplayPageBoundaries (Do Not Display Visual Boundary For Header/Footer or Between Pages)

This element specifies whether applications displaying this document should display the contents of the header and footer when displaying the document in print layout view (§17.15.1.92) or should collapse those areas as well as the whitespace on all displayed pages so that the text extents are directly following one another. [*Rationale*: Collapsing the ends of pages makes it easier to read the contents of the document, since the text flows between pages without whitespace, while maintaining the WYSIWYG functionality of print layout view for the document's main content. *end rationale*]

If this element is omitted, then all pages should be shown at their full size (including whitespace and headers/footers) when they are displayed in print layout view.

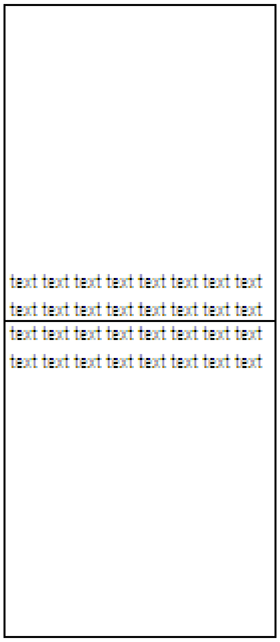
[*Example*: Consider the images below illustrating two pages in a WordprocessingML document:


text text text text text text text text
text text text text text text text text
text text text text text text text text
text text text text text text text text

If document must automatically have whitespace between pages removed when it is displayed, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:doNotDisplayPageBoundaries w:val="true" />
```

The resulting output might look like the following:


text text text text text text text text
text text text text text text text text
text text text text text text text text
text text text text text text text text

The doNotDisplayPageBoundaries element has its val attribute equal to true, therefore the document is automatically displayed with whitespace between text extents on following pages compressed, allowing the pages to be viewed more easily. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.36 doNotEmbedSmartTags (Remove Smart Tags When Saving)

This element specifies if any smart tags specified using the smartTag element shall be removed from the contents of this document before it is resaved. This setting shall also prevent the addition of new smart tags to the content of the document.

If this element is omitted, then smart tags shall not be removed from the file when it is saved.

[*Example*: Consider a WordprocessingML document which should never be saved with smartTag elements in its contents. This requirement is specified using the following WordprocessingML fragment in the document settings:

```
<w:doNotEmbedSmartTags w:val="true"/>
```

The doNotEmbedSmartTags element's val attribute has a value of true specifying that smart tags must never be saved in the contents of this document. For example, if a run formerly looked like this:

```
<w:p>
  <w:r>
    <w:t xml:space="preserve">Hello</w:t>
  </w:r>
  <w:smartTag … >
    <w:r>
      <w:t>world</w:t>
    </w:r>
  </w:smartTag>
</w:p>
```

The presence of this element specifies that the SmartTag element must be removed, and applications might then choose to combine duplicated runs as desired. *end example*]

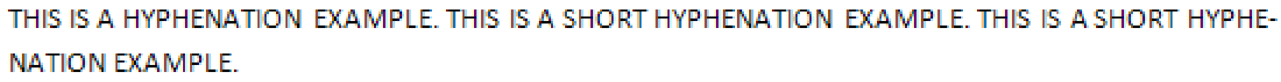
This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.37 doNotHyphenateCaps (Do Not Hyphenate Words in ALL CAPITAL LETTERS)

This element specifies whether or not words comprised of all capital letters shall be hyphenated within a given document when automatic hyphenation is specified via the autoHyphenation element (§17.15.1.10).

If this element is omitted, then words in ALL CAPITAL LETTERS shall be hyphenated when the document is hyphenated.

[*Example*: Consider a document which is automatically hyphenated containing the following paragraph of content:

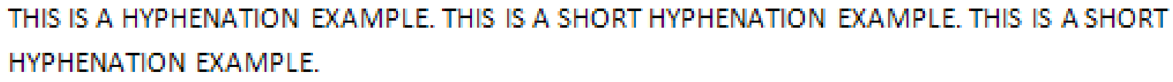


THIS IS A HYPHENATION EXAMPLE. THIS IS A SHORT HYPHENATION EXAMPLE. THIS IS A SHORT HYPHE-
NATION EXAMPLE.

If words in ALL CAPITAL LETTERS must not be hyphenated, this requirement would be specified by adding the following WordprocessingML to the document settings part:

```
<w:doNotHyphenateCaps w:val="true"/>
```

The resulting content would not be hyphenated:



THIS IS A HYPHENATION EXAMPLE. THIS IS A SHORT HYPHENATION EXAMPLE. THIS IS A SHORT
HYPHENATION EXAMPLE.

The doNotHyphenateCaps element val set to true, specifying that the first line of text to end with the word SHORT as the word HYPHENATION had to be moved to the second line since it could not fit in its entirety on the first line.

Conversely, setting the doNotHyphenateCaps element val set to off (the default) caused the first line of text to contain a hyphenated portion of the word HYPHENATION as hyphenation of words comprised of all capital letters is permitted. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.38 doNotIncludeSubdocsInStats (Do Not Include Content in Text Boxes, Footnotes, and Endnotes in Document Statistics)

This element specifies if document content contained in text boxes, footnotes, and endnotes shall be excluded when an application calculates a given document's statistics when these values are calculated and/or displayed by an application.

[*Note*: Some examples of document statistics that an application might chose to calculate are: number of words, number of characters, number of paragraphs, number of pages, number of lines, and so on. *end note*]

[*Example*: Consider a WordprocessingML that specifies that it must not include these document stories when its contents are used to calculate document statistics. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:doNotIncludeSubdocsInStats w:val="true"/>
```

The doNotIncludeSubdocsInStats element's val attribute has a value of true specifying that only the contents of the main document story should be used when calculating document statistics. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.39 doNotShadeFormData (Do Not Show Visual Indicator For Form Fields)

This element specifies whether a visual cue should be displayed around form fields contained in a WordprocessingML document specified via the FORMTEXT, FORMCHECKBOX, or FORMDROPDOWN fields.

If this element is not present in a WordprocessingML document visual cues should be displayed on form fields contained in the document.

[*Example*: Consider a WordprocessingML document which should no visual indication of form fields. This requirement would be specified using the following WordprocessingML:

```
<w:doNotShadeFormData w:val="true"/>
```

The doNotShadeFormData element's val attribute has a value of true specifying the display of any visual indication of form fields must be suppressed for this document. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.40 doNotTrackFormatting (Do Not Track Formatting Revisions When Tracking Revisions)

This element specifies that applications shall not track revisions made to the formatting of this WordprocessingML document when the trackRevisions element (§17.15.1.89) is turned on.

If this element is omitted, then revisions to formatting shall be generated by changes to the contents of this document when the trackRevisions element is turned on.

[*Example*: Consider a WordprocessingML document containing the text run Example that must have revisions tracked. Example WordprocessingML from Document 1 is given below:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

If the word text was added to the end of this document and bolded, the resulting WordprocessingML would be output as follows:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
      <w:ins … >
        <w:r>
          <w:rPr>
            <w:b/>
            <w:rPrChange … >
              <w:rPr/>
            </w:rPrChange>
          </w:rPr>
          <w:t>text</w:t>
        </w:r>
      </w:ins>
    </w:p>
  </w:body>
</w:document>
```

If changes to formatting were turned off using the following WordprocessingML syntax in the document settings:

```
<w:settings>
  <w:trackRevisions w:val="true" />
  <w:doNotTrackFormatting w:val="true" />
  …
</w:settings>
```

The same revision (the word `text` was added to the end of this document and bolded) would result in the following markup:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
      <w:ins … >
        <w:r>
          <w:rPr>
            <w:b/>
          </w:rPr>
          <w:t>text</w:t>
        </w:r>
      </w:ins>
    </w:p>
  </w:body>
</w:document>
```

The doNotTrackFormatting element's val attribute was set to `true`, therefore the changes to the formatting of the document were not tracked as revisions in the document's WordprocessingML. Specifically, applying bold formatting to the text was not tracked as a revision with the rPrChange (§17.13.5.31) element. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.41 doNotTrackMoves (Do Not Use Move Syntax When Tracking Revisions)

This element specifies that applications shall not track revisions made to this WordprocessingML document as moves when the trackRevisions element (§17.15.1.89) is turned on, even when that syntax is appropriate. Instead, applications should use a standard insertion and deletion annotation syntax. Existing moves shall not be modified. [*Rationale*: This element is provided to enable interoperability with earlier word processing applications which do not understand moves. *end rationale*]

If this element is omitted, then move annotations can be generated by changes to the contents of this document when the trackRevisions element is turned on as appropriate.

[*Example*: Consider a WordprocessingML that specifies that it must not have additional moves added to its contents. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:doNotTrackMoves w:val="true"/>
```

The doNotTrackMoves element's val attribute has a value of `true` specifying that insertion/deletion annotations must be used rather than moves when revisions are tracked in this document. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.42 doNotUseMarginsForDrawingGridOrigin (Do Not Use Margins for Drawing Grid Origin)

This element specifies that the top-left corner of the page shall not be used as the origin for the drawing grid. The *drawing grid* is a virtual grid which can be used by applications to specify where drawing objects shall be positioned on a page when inserted (i.e. to ensure objects are aligned, etc.). If this element is present the grid shall start at the top-left edge of the page and not the text extents.

If this element is omitted, then the gridlines shall start at the topmost edge of the text extents.

[*Example*: Consider a WordprocessingML document whose drawing grid must begin at the top left edge of the page. This requirement would be specified using the following WordprocessingML markup in the document settings:

```
<w:doNotUseMarginsForDrawingGridOrigin w:val="true" />
```

The doNotUseMarginsForDrawingGridOrigin element's val attribute is equal to `true` specifying that the document's drawing grid must begin from the top left corner of the page, rather than the top left corner of the text extents. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.43 doNotValidateAgainstSchema (Do Not Validate Custom XML Markup Against Schemas)

This element specifies that applications shall not validate the custom XML markup in this document against the applicable custom XML schema(s), even when those schemas are available. The application should silently behave as if it was unable to provide this functionality.

If this element is omitted, then applications which support this functionality should attempt to validate the custom XML contents against any available related custom XML schema(s).

[*Example*: Consider a WordprocessingML document which should not have its custom XML content validated even by applications which support this operation. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:doNotValidateAgainstSchema w:val="true" />
```

The doNotValidateAgainstSchema element's val attribute has a value of `true` specifying that the custom XML markup in this document must not be validated. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.44 drawingGridHorizontalOrigin (Drawing Grid Horizontal Origin Point)

This element specifies the distance from of the left edge of the page which shall be used as the origin for the horizontal gridlines used by the drawing grid. The *drawing grid* is a virtual grid which might be used by applications to specify where drawing objects shall be positioned on a page when inserted (i.e. to ensure objects are aligned, etc.). Since the grid always covers the entire page when the doNotUseMarginsForDrawingGridOrigin element (§17.15.1.42) is specified, this element shall only affect the starting edge of the first horizontal gridline displayed (i.e. it only adjusts the grid by the modulus of the value against the width of one grid unit).

If this element is omitted, then the gridlines shall start at the leftmost edge of the page. If the doNotUseMarginsForDrawingGridOrigin element is not specified, then this element is ignored.

[*Example*: Consider a WordprocessingML document whose drawing grid must begin three inches (4320 twentieths of a point) before the left edge of the page. This requirement would be specified using the following WordprocessingML markup in the document settings:

```
<w:settings>
  …
  <w:doNotUseMarginsForDrawingGridOrigin w:val="true" />
  <w:drawingGridHorizontalOrigin w:val="4320" />
  …
</w:settings>
```

The drawingGridHorizontalOrigin element's val attribute is equal to 4320 specifying that the horizontal edge of the document's drawing grid must begin three inches (4320 twentieths of a point) from the left edge of the page, since the doNotUseMarginsForDrawingGridOrigin element's val attribute is equal to true. *end example*]

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<… w:val="720" />` The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

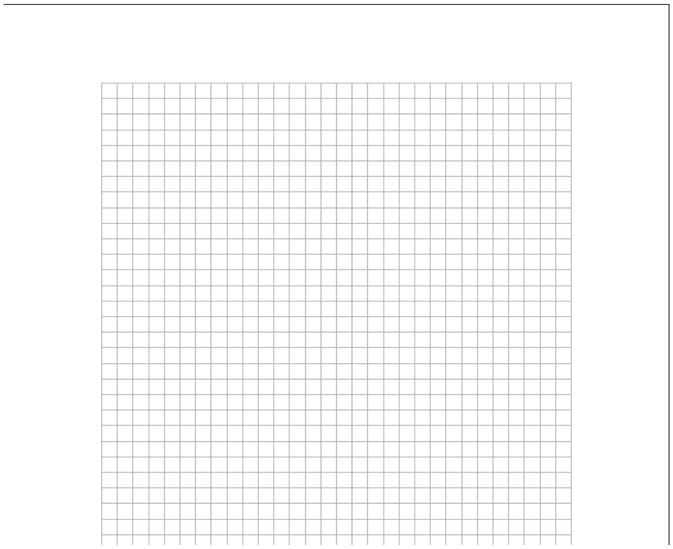
[*Note*: The W3C XML Schema definition of this element's content model (CT_TwipsMeasure) is located in §A.1. *end note*]

17.15.1.45 drawingGridHorizontalSpacing (Drawing Grid Horizontal Grid Unit Size)

This element specifies the width of horizontal grid units in this document. The *drawing grid* is a grid which can be used by applications to help position floating objects in the document.

If this element is omitted, then each horizontal grid unit shall be 180 twentieths of a point (0.125") in width.

[*Example*: Consider the image below illustrating a WordprocessingML document in which all horizontal grid units are each 144 twentieths of a point wide (and all are showing):



If the gridlines in this document must only be displayed for every half an inch, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:drawingGridHorizontalSpacing w:val="720" />
```

The resulting grid would look like the following:

The drawingGridHorizontalSpacing element has its val attribute equal to 720, therefore every horizontal gridline has a width of one half of an inch (720 twentieths of a point). *end example*]

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<… w:val="720" />` The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_TwipsMeasure) is located in §A.1. *end note*]

17.15.1.46 drawingGridVerticalOrigin (Drawing Grid Vertical Origin Point)

This element specifies the distance from of the top edge of the page which shall be used as the origin for the vertical gridlines used by the drawing grid. The *drawing grid* is a virtual grid which can be used by applications to specify where drawing objects shall be positioned on a page when inserted (i.e. to ensure objects are aligned, etc.). Since the grid always covers the entire page when the doNotUseMarginsForDrawingGridOrigin element (§17.15.1.42) is specified, this element shall only affect the starting edge of the first vertical gridline displayed (i.e. it only adjusts the grid by the modulus of the value against the width of one grid unit).

If this element is omitted, then the gridlines shall start at the topmost edge of the page. If the doNotUseMarginsForDrawingGridOrigin element is not specified, then this element is ignored.

[*Example*: Consider a WordprocessingML document whose drawing grid must begin one inch (1440 twentieths of a point) before the top edge of the page. This requirement would be specified using the following WordprocessingML markup in the document settings:

```
<w:settings>
  …
  <w:doNotUseMarginsForDrawingGridOrigin w:val="true" />
  <w:drawingGridVerticallOrigin w:val="1440" />
  …
</w:settings>
```

©ISO/IEC 2012 – All rights reserved

The drawingGridVerticalOrigin element's val attribute is equal to 1440 specifying that the vertical edge of the document's drawing grid must begin one inch (1440 twentieths of a point) from the top edge of the page, since the doNotUseMarginsForDrawingGridOrigin element's val attribute is equal to true. *end example*]

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<… w:val="720" />` The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

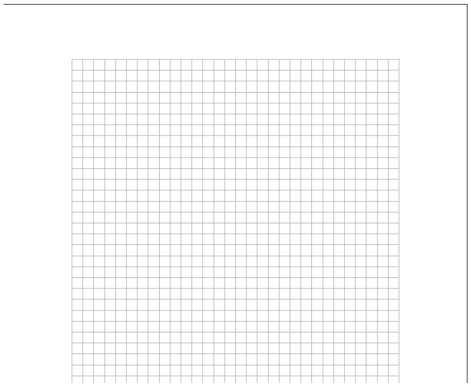
[*Note*: The W3C XML Schema definition of this element's content model (CT_TwipsMeasure) is located in §A.1. *end note*]

17.15.1.47 drawingGridVerticalSpacing (Drawing Grid Vertical Grid Unit Size)

This element specifies the width of vertical grid units in this document. The *drawing grid* is a grid which can be used by applications to help position floating objects in the document.

If this element is omitted, then each vertical grid unit shall be 180 twentieths of a point (0.125") in width.

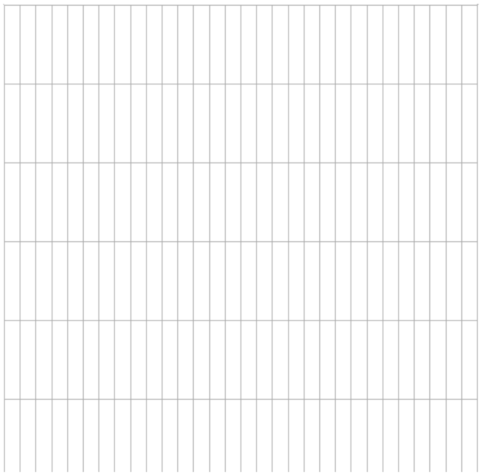
[*Example*: Consider the image below illustrating a WordprocessingML document in which all vertical grid units are each 144 twentieths of a point high (and all are showing):



If the vertical gridlines in this document must only be displayed for every half an inch, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:drawingGridVerticalSpacing w:val="720" />
```

The resulting grid would look like the following:



The drawingGridVerticalSpacing element has its val attribute equal to 720, therefore every vertical gridline has a height of one half of an inch (720 twentieths of a point). *end example*]

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<… w:val="720" />` The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TwipsMeasure) is located in §A.1. *end note*]

17.15.1.48 forceUpgrade (Upgrade Document on Open)

This element specifies that the contents of this document can be upgraded and that the resulting document shall not have its functionality limited to only those functions compatible with earlier word processing applications. The only actions required as part of upgrading the document are:

- The removal of this element. If an application does not know how to upgrade a document, this element should be ignored and persisted.
- The removal of all compatibility options (§17.15.1.21) on the document which maintain compatibility with previous word processing applications. The compatibility settings which simply affect a given behavior shall not be turned off.

[*Note*: The remaining operations which must be performed as part of upgrading the document are application-defined and outside the scope of ISO/IEC 29500. *end note*]

[*Example*: Consider a WordprocessingML document that specifies that it must automatically be upgraded when it is opened by an application. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:forceUpgrade/>
```

The forceUpgrade element's presence specifies that this document must be upgraded by any application which supports this operation. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.1. *end note*]

17.15.1.49 formsDesign (Structured Document Tag Placeholder Text Should be Resaved)

This element specifies that the document was last saved while the placeholder text of all structured document tags in this document were being edited. This means that the placeholder text currently displayed in all structured document tags which are displaying the showingPlcHdr element (§17.5.2.39) shall be committed to the corresponding glossary document entry as specified using the docPart element (§17.12.5) when this document is opened, in order to ensure that the most recent placeholder text is stored in the glossary document entry. If the current placeholder text cannot be saved as a glossary document entry, then it should be modified as needed before saving.

If this element is omitted, then the placeholder text in this document should not automatically be resaved when the document is opened.

[*Example*: Consider a WordprocessingML document that specifies that its placeholder text should be resaved to the glossary document when the file is opened. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:formsDesign w:val="true"/>
```

The formsDesign element's val attribute has a value of true specifying that this document should be resaved to its glossary document by any application which supports this operation. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.50 gutterAtTop (Position Gutter At Top of Page)

This element specifies that a given WordprocessingML document’s gutter shall be positioned at the top of the document’s pages when the document is displayed. A *gutter* is the white space formed by the inner margins of two pages facing one another; such as the white space between the text on pages of a book when the book is opened.

If this element is omitted, then the gutter shall not be positioned at the top of the page. If the mirrorMargins (§17.15.1.57), bookFoldPrinting (§17.15.1.11), bookFoldRevPrinting (§17.15.1.13), or printTwoOnOne (§17.15.1.64) elements are used within a given document, the gutterAtTop element shall not be used. Rather, the gutter shall be positioned automatically as necessary to enable the printing and page layout capabilities of these settings.

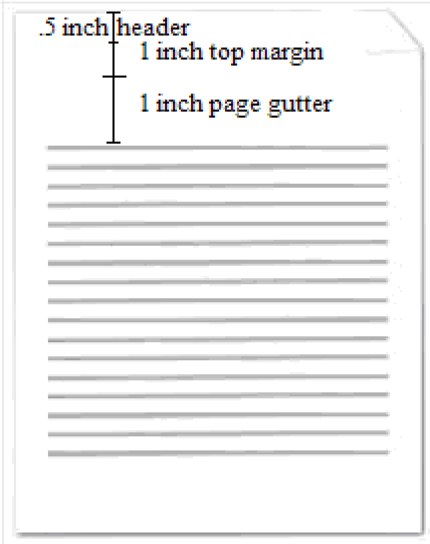
[*Example*: Consider a one page WordprocessingML document with a 1,440 twentieths of a point (one inch) top margin and gutter, and a 720 twentieths of a point (one half of an inch) header. Consider also, that the gutter must exist at the top of the document's pages. This requirement is specified using the following WordprocessingML in the section properties:

```
<w:pgMar w:top="1440" … w:header="720" … w:gutter="1440" />
```

And the following WordprocessingML in the document settings:

```
<w:gutterAtTop w:val="true" />
```

The resulting document's pages would have the gutter positioned as follows:



The gutterAtTop element's val attribute is equal to `true`, specifying that the gutter must appear at the top of each page. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.51 hideGrammaticalErrors (Do Not Display Visual Indication of Grammatical Errors)

This element specifies whether a visual cue should be displayed around run content contained in a WordprocessingML document which has been flagged as a possible grammatical error using the proofErr element (§17.13.8.1) or via the application's own grammar engine.

If this element is not present in a WordprocessingML document, visual cues shall be displayed on content contained in a WordprocessingML document which is considered to contain grammatical errors.

[*Example*: Consider a WordprocessingML document which should show no visual indication of grammatical errors. This requirement would be specified using the following WordprocessingML:

```
<w:hideGrammaticalErrors w:val="true"/>
```

The hideGrammaticalErrors element's val attribute has a value of `true` specifying the display of any visual indication of grammatical errors must be suppressed for this document. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.52 hideSpellingErrors (Do Not Display Visual Indication of Spelling Errors)

This element specifies whether a visual cue should be displayed around run content contained in a WordprocessingML document which has been flagged as a possible spelling error using the proofErr element (§17.13.8.1) or via the application's own spelling engine.

If this element is not present in a WordprocessingML document, visual cues shall be displayed on content contained in a WordprocessingML document which is considered to contain spelling errors.

[*Example*: Consider a WordprocessingML document which should show no visual indication of spelling errors. This requirement would be specified using the following WordprocessingML:

```
<w:hideSpellingErrors w:val="true"/>
```

The hideSpellingErrors element's val attribute has a value of `true` specifying the display of any visual indication of spelling errors must be suppressed for this document. *end example*]

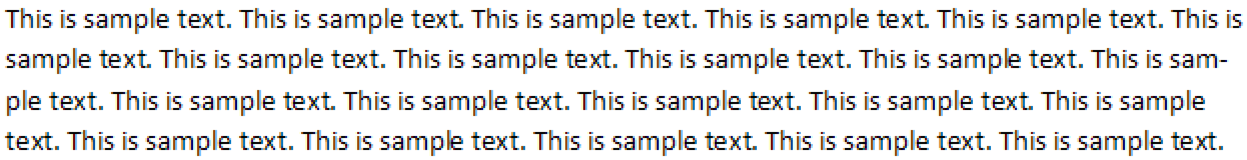
This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.53 hyphenationZone (Hyphenation Zone)

This element specifies the hyphenation zone which shall be used when automatically or manually hyphenating the contents of this document. The *hyphenation zone* is the amount of whitespace which can be left at the end of a line (or added to justified lines) before hyphenation should be attempted on the next word in the document (in order to reduce the amount of whitespace on the line). A smaller hyphenation zone should reduce the raggedness of the right edge of a given document's body text, as more words is hyphenated. Conversely, a larger hyphenation zone should increase the raggedness of the right edge of a given document's text, as fewer words is hyphenated.

If this element is omitted, then a default hyphenation zone of 360 twentieths of a point (0.25") shall be applied when performing hyphenation on this document.

[*Example*: Consider the images below illustrating a paragraph of text in a WordprocessingML document which was automatically hyphenated with the default hyphenation zone:

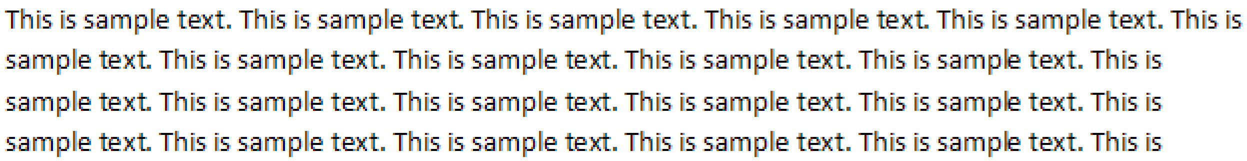


This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is
sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is sam-
ple text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample
text. This is sample text. This is sample text. This is sample text. This is sample text. This is sample text.

If document must instead be hyphenated using a hyphenation zone of one-half of an inch, that requirement would be specified using the following WordprocessingML in the document settings:

```
<w:hyphenationZone w:val="720" />
```

The resulting output would only hyphenate words when the remaining line whitespace was less than half an inch, and would look like the following:



This is sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is
sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is
sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is
sample text. This is sample text. This is sample text. This is sample text. This is sample text. This is

The hyphenationZone element has its val attribute equal to 720, therefore lines in the document are hyphenated when the remaining line spacing is less than 72o twentieths of a point (0.5"), resulting in fewer hyphens. *end example*]

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<… w:val="720" />` The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TwipsMeasure) is located in §A.1. *end note*]

17.15.1.54 ignoreMixedContent (Ignore Mixed Content When Validating Custom XML Markup)

This element specifies that applications should ignore all text content which is not contained within a leaf custom XML markup element when validating the contents of the custom XML markup in this document against one or more attached custom XML schema(s). A *leaf element* is a custom XML element which has no child custom XML elements (it is a leaf in the custom XML tree).

If this element is omitted, then text content in leaf elements shall not be ignored when validating the custom XML markup against one or more custom XML schema(s).

[*Example*: Consider a WordprocessingML document that contains the following markup:

```
<w:customXml w:element="invoice" w:uri="http://www.example.com/invoice">
  <w:p>
    <w:r>
      <w:t>Invoice #:</w:t>
    </w:r>
    <w:customXml w:element="id" w:uri="http://www.example.com/invoice">
      <w:r>
        <w:t>012345</w:t>
      </w:r>
    </w:customXml>
  </w:p>
  <w:p>
    <w:r>
      <w:t>Invoice Date:</w:t>
    </w:r>
    <w:customXml w:element="date" w:uri="http://www.example.com/invoice">
      <w:r>
        <w:t>01/29/2009</w:t>
      </w:r>
    </w:customXml>
  </w:p>
</w:customXml>
```

If all the custom markup is extracted from the document, that markup would include all content in the document, i.e.:

```
<invoice xmlns="http://www.example.com/invoice">
  Invoice #:
  <id>012345</id>
  Invoice Date
  <date>01/29/2009</date>
</invoice>
```

The content shown above is formatted for readability. In fact, there is no such space in the XML.

However, if the ignoreMixedContent element is present with a val attribute value of `true` (or equivalent) then an application should ignore all text nodes in elements with mixed content, i.e.:

```
<invoice xmlns="http://www.example.com/invoice">
  <id>012345</id>
  <date>01/29/2009</date>
</invoice>
```

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.55 linkStyles (Automatically Update Styles From Document Template)

This element specifies that styles in the given document shall be updated to match the styles in the attached template specified using the attachedTemplate element (§17.15.1.6) when the document is opened by a hosting application. This setting enables the styles contained in documents with attached templates to stay synchronized with the styles used in the attached template.

If this element is omitted, then styles shall not be updated based on the document template regardless of its availability. If the attached template cannot be located or is not a valid file, then this setting should be silently ignored.

[*Example*: Consider a WordprocessingML document which should always update its styles with those defined in the document's attached template. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:settings>
  <w:linkStyles w:val="true" />
  <w:attachedTemplate r:id="rId10" />
  …
</w:settings>
```

The linkStyles element has a val attribute value of `true`, specifying that applications should attempt to locate the document template referenced by the relationship specified in the attachedTemplate element and update the document's styles with the styles from that template. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.56 listSeparator (List Separator for Field Code Evaluation)

This element specifies the character that shall be interpreted as a list item separator when evaluating the contents of all fields in the current document.

[*Rationale*: When evaluating field instructions based on the contents of the current document, it is necessary to know the character which must be treated as the list separator in order to prevent changes to the calculation of the same field instructions based on the current user's locale. This element stores the list separator which must be used to evaluate fields in the contents of this document, irrespective of the locale of the application loading the file. *end rationale*]

If this element is omitted, the application shall use the default list separator of its current locale setting to evaluate field instructions. If this element's attribute value is more than a single character, then the document is non-conformant.

[*Example*: Consider a WordprocessingML document which should use the semicolon character as the list separator for all field instructions. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:listSeparator w:val=";" />
```

The listSeparator element's val attribute has a value of `;` specifying that the semicolon character shall be interpreted as a list item separator.

For instance, the string `10;20,5` would be interpreted as having two values - `10` and `20,5`. If the listSeparator was a comma, the same string would be interpreted as `10;20` and `5`. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`

Attributes	Description
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.15.1.57 mirrorMargins (Mirror Page Margins)

This element specifies that the left and right margins defined in the section properties shall be swapped on every second page. [*Note:* Page numbering can be set arbitrarily, so the flip might not always be on the pages with even-numbered labels. *end note*]

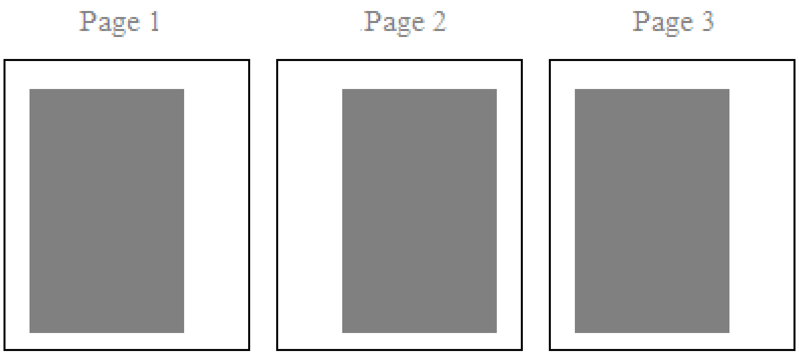
[*Guidance*: This setting is generally used when printing on both sides of pages and binding them like a book. *end guidance*]

[*Example*: Consider a graphical representation (below) of a three page WordprocessingML document with a left margin of 1" and a right margin of 2".

If the mirrorMargins element is present in the document settings with its val attribute equal to true, as follows:

```
<w:mirrorMargins w:val="true" />
```

The resulting pages has mirrored margins as follows (un this representation, the gray rectangles representing the text extents on each page):



end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

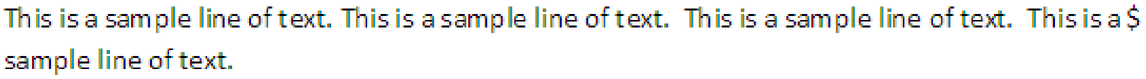
17.15.1.58 noLineBreaksAfter (Custom Set of Characters Which Cannot End a Line)

This element specifies the set of characters which shall be restricted from ending a line for runs of text which shall be subject to custom line breaking logic using the kinsoku element (§17.3.1.16) when the contents of the document are displayed. This constraint shall only apply to text which has been flagged in the language of this rule via the lang element (§17.3.2.20) or automatic detection methods outside the scope of ISO/IEC 29500.

If this element is omitted, then no custom set of characters shall be used to restrict the characters which can end a line when using the kinsoku element.

[*Example*: Consider a paragraph of WordprocessingML text displayed as follows, with the dollar symbol $ was flagged as Japanese content using the following WordprocessingML in the run properties:

```
<w:r>
  <w:rPr>
    <w:lang w:eastAsia="ja-JP" />
  </w:rPr>
  <w:t>$</w:t>
</w:r>
```

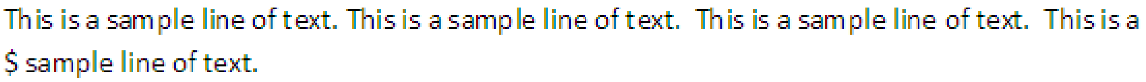


This is a sample line of text. This is a sample line of text. This is a sample line of text. This is a $
sample line of text.

This text is displayed and the resulting first line ends with the dollar sign symbol. If this character must not be used to end a line, that requirement would be specified as follows in the document settings:

```
<w:noLineBreaksAfter w:lang="ja-JP" w:val="$" />
```

The noLineBreaksAfter element's val attribute has a value of `ja-JP`, specifying that all dollar signs in this document which are marked as Japanese text must not be allowed to end a line. This means that the dollar sign character must therefore be moved to the next line as it can no longer be the last character on a line:



This is a sample line of text. This is a sample line of text. This is a sample line of text. This is a
$ sample line of text.

end example]

Attributes	Description
lang (Language For Which Custom Line Breaking Rule Applies)	Specifies the language of text for which the parent custom line breaking rule shall be applied. Applications supporting this functionality shall support custom line breaking for the following four languages: • Chinese (Traditional) • Chinese (Simplified) • Japanese • Korean

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	Applications can also support custom line breaking rules for other languages, but this is not required. [*Example*: Consider a WordprocessingML document which must have a custom line breaking rule for Japanese. That requirement would be specified as follows in the document settings: `<… w:lang="ja-JP" w:val="$" />` The lang attribute has a value of `ja-JP`, specifying that the rules must be applied to Japanese text. *end example*] The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).
val (Characters For Custom Line Breaking Rule)	Specifies the set of characters which shall be included in the custom line breaking rule. [*Example*: Consider a WordprocessingML document which must have a custom line breaking rule for Japanese. That requirement would be specified as follows in the document settings: `<… w:lang="ja-JP" w:val="$" />` The val attribute has a value of `$`, specifying that the dollar sign character is the only restricted character for Japanese text. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Kinsoku) is located in §A.1. *end note*]

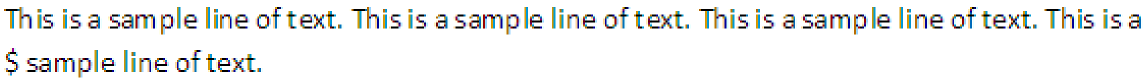
17.15.1.59 noLineBreaksBefore (Custom Set Of Characters Which Cannot Begin A Line)

This element specifies the set of characters which shall be restricted from beginning a new line for runs of text which shall be subject to custom line breaking logic using the kinsoku element (§17.3.1.16) when the contents of the document are displayed. This constraint shall only apply to text which has been flagged in the language of this rule via the lang element (§17.3.2.20) or automatic detection methods outside the scope of ISO/IEC 29500.

If this element is omitted, then no custom set of characters shall be used to restrict the characters which can end a line when using the kinsoku element.

[*Example*: Consider a paragraph of WordprocessingML text displayed as follows, with the dollar symbol $ was flagged as Korean content using the following WordprocessingML in the run properties:

```
<w:r>
  <w:rPr>
    <w:lang w:eastAsia="ko-KR" />
  </w:rPr>
  <w:t>$</w:t>
</w:r>
```

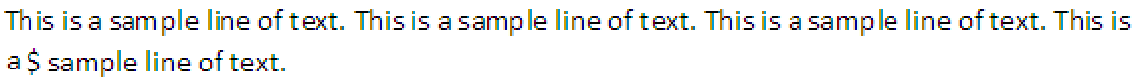


This is a sample line of text. This is a sample line of text. This is a sample line of text. This is a
$ sample line of text.

This text is displayed and the resulting second line begins with the dollar sign symbol. If this character must not be used to begin a line, that requirement would be specified as follows in the document settings:

```
<w:noLineBreaksBefore w:lang="ko-KR" w:val="$" />
```

The noLineBreaksBefore element's val attribute has a value of `ko-KR`, specifying that all dollar signs in this document which are marked as Korean text must not be allowed to begin a line. This means that the previous word character must therefore be moved to the next line as the dollar sign can no longer be the first character on a line:



This is a sample line of text. This is a sample line of text. This is a sample line of text. This is
a $ sample line of text.

end example]

Attributes	Description
lang (Language For Which Custom Line Breaking Rule Applies)	Specifies the language of text for which the parent custom line breaking rule shall be applied. Applications supporting this functionality shall support custom line breaking for the following four languages: • Chinese (Traditional) • Chinese (Simplified) • Japanese • Korean Applications can also support custom line breaking rules for other languages, but this is not required. [*Example*: Consider a WordprocessingML document which must have a custom line breaking rule for Japanese. That requirement would be specified as follows in the document settings: `<… w:lang="ja-JP" w:val="$" />` The lang attribute has a value of `ja-JP`, specifying that the rules must be applied to Japanese text. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).
val (Characters For Custom Line Breaking Rule)	Specifies the set of characters which shall be included in the custom line breaking rule. [*Example*: Consider a WordprocessingML document which must have a custom line breaking rule for Japanese. That requirement would be specified as follows in the document settings: `<… w:lang="ja-JP" w:val="$" />` The val attribute has a value of $, specifying that the dollar sign character is the only restricted character for Japanese text. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Kinsoku) is located in §A.1. *end note*]

17.15.1.60 noPunctuationKerning (Never Kern Punctuation Characters)

This element specifies that punctuation characters shall not be kerned in the current document when kerning is enabled on a run using the kern element (§17.3.2.19). *Kerning* refers to a process by which a hosting application shall reduce the spacing of adjacent characters and/or punctuation to improve the visual appearance of text. Well kerned text has a similar amount of blank space between each pair of characters and/or each set of a character and punctuation symbol. When kerning is enabled, Latin text shall always be kerned, and this option shall control whether punctuation characters are also kerned.

If this element is omitted, then punctuation characters shall be kerned when kerning is enabled on a given run.

[*Example*: Consider a WordprocessingML document that must not kern punctuation even when kerning is enabled on a given run. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:noPunctuationKerning w:val="true" />
```

The noPunctuationKerning element's val attribute has a value of true, specifying that punctuation characters must not be kerned in this document. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

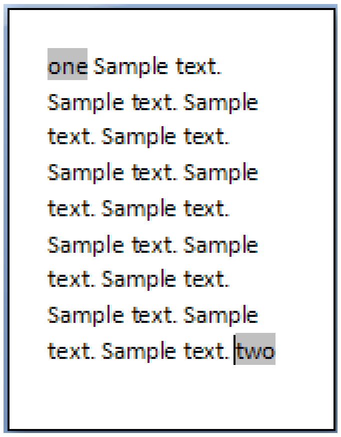
17.15.1.61 printFormsData (Only Print Form Field Content)

This element specifies that printing the contents of this document shall only print the contents of WordprocessingML form fields defined using the FORMTEXT, FORMCHECKBOX, and FORMDROPDOWN field codes in their current locations on the page - all other document contents shall be suppressed.

[*Rationale*: This setting is typically used to allow duplication of paper forms in electronic WordprocessingML document form, allowing the resulting online document to be printed into the correct locations on the existing paper form. *end rationale*]

If this element is omitted, then the contents of the entire document (not just form fields) should be printed according to the normal print settings.

[*Example*: Consider a WordprocessingML document which has form fields in the top right and bottom left corners of the first page, as follows (with the text box form fields shaded in grey):

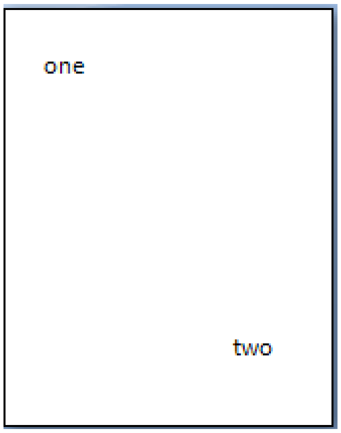


one Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. Sample text. two

If the only content which must be printed on the page are the form fields' contents, this requirement is specified using the following WordprocessingML in the document settings:

```
<w:printFormsData w:val="true" />
```

The printFormsData element's val attribute as a value of true, specifying that only form field data shall be printed, resulting in output as follows when printed:



one

two

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.62 printFractionalCharacterWidth (Print Fractional Character Widths)

This element specifies the contents of this document shall be printed with fractional character widths. *Fractional character widths* exist when the spacing between characters is not constant (i.e. a proportional font face is used).

[*Note*: Fractional character widths are generally used in conjunction with large font sizes to prevent characters from running together or having too much space between one another. *end note*]

[*Example*: Consider a WordprocessingML document which should be printed using fractional character widths as needed. This requirement is specified using the following WordprocessingML markup in the document settings:

```
<w:printFractionalCharacterWidth w:val="true"/>
```

The printFractionalCharacterWidth element's val attribute is equal to `true`, specifying that fractional character widths can be used as necessary. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.63 printPostScriptOverText (Print PostScript Codes With Document Text)

This element specifies that the PostScript codes specified in WordprocessingML documents containing `PRINT` fields shall be included in foreground (on the same Z-order as text) with the data printed in the contents of a given WordprocessingML document.

[*Note*: This setting is maintained to ensure compatibility of legacy word processing documents. The `PRINT` field should not be used in lieu of newer technologies in ISO/IEC 29500. *end note*]

If this element is omitted, then the contents of `PRINT` fields shall be printed behind text (i.e. in the background).

[*Example*: Consider a WordprocessingML document containing `PRINT` fields whose PostScript code must be printed in the foreground of the WordprocessingML document. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:printPostScriptOverText w:val="true"/>
```

The printPostScriptOverText element's val attribute is equal to `true` specifying that the PostScript codes must be treated as results for the main text level of the document (i.e. not behind that text). *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.64 printTwoOnOne (Print Two Pages Per Sheet)

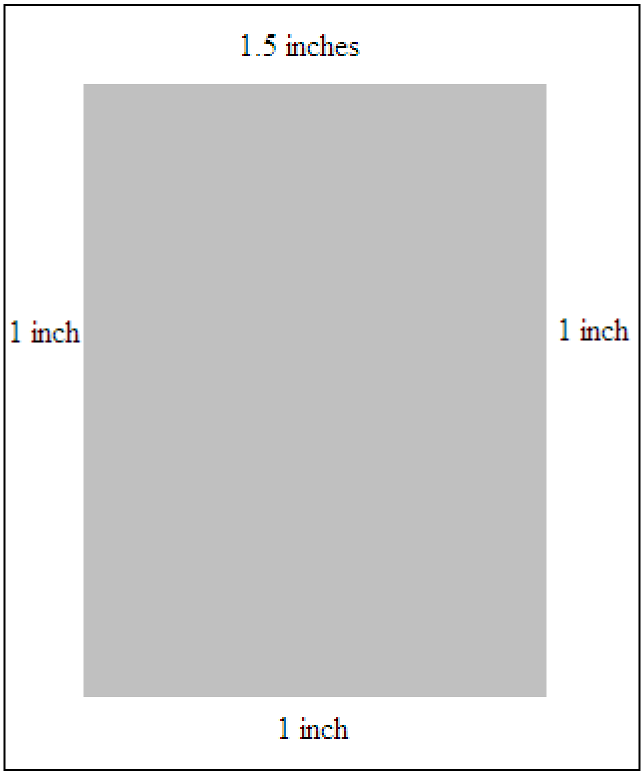
This element specifies whether two pages should be printed on one sheet of paper when this document is printed. Specifically, this element specifies that each page displayed for the contents in a given WordprocessingML document should be the page size specified in the section settings divided in half with two top margins originating from the bisector of the page, and bottom margins instantiated at the top and bottom of each page.

If this element is omitted, then pages should be displayed and printed as one per sheet.

[*Example*: Consider a one section document with a 2,160 twentieths of a point (one and a half inch) top margin, and 1,440 twentieths of a point (one inch) bottom, right, and left margins surrounding the document editing canvas (represented by the gray shaded area in diagrams below). This page setup is represented in WordprocessingML using the following fragment:

```
<w:pgMar w:header="0" w:top="2160" w:right="1440" w:bottom="1440" w:left="1440"
  w:footer="720" w:gutter="0" />
```

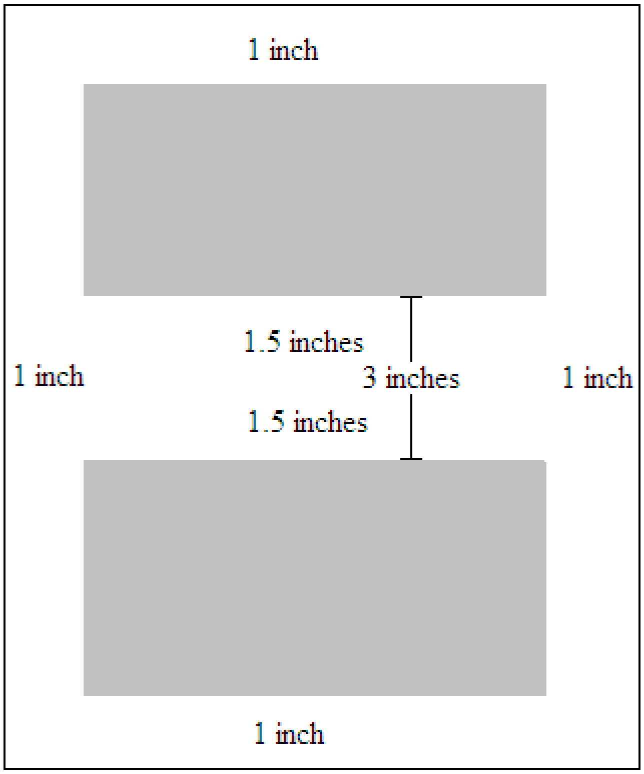
The resulting printed pages would appear as follows:



If a document should be displayed and printed as though two pages were printed on a single sheeting, this requirement would be specified using the following WordprocessingML:

```
<w:printTwoOnOne w:val="true" />
```

The printTwoOnOne element's val attribute is equal to true specifying that pages should be printed two to a sheet, resulting in the following printout given these page margins:



end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.65 proofState (Spelling and Grammatical Checking State)

This element specifies if the grammar and spell checking engines of the last application to process this document completed checking the grammar and spelling of a the document before the document was last saved. Applications which modify the document contents without checking spelling or grammar should reset these states as needed.

[*Note*: If this element specifies that an application's grammar and spell checking engines completed checking the grammar and spelling of the document when the document was last saved, then subsequent applications might elect to not run their grammar and spell checking engines when the given WordprocessingML document is loaded.

This might increase the speed with which the hosting application loads the file, and does not compromise the state of the grammar or spell checking of the document, as all errors have already been found and flagged with the proofErr element (§17.13.8.1) as the document has not been edited, only loaded, since it was last saved. *end note*]

[*Example*: Consider a WordprocessingML document that is saved by a hosting application whose spelling and grammar checking engines have completed checking grammar and spelling in the given WordprocessingML document. This state is specified using the following WordprocessingML in the document settings:

```
<w:proofState w:spelling="clean" w:grammar="clean" />
```

The proofState element's attributes spelling and grammar attribute both have the value `clean` specifying that the hosting application's grammar and spell checking engines completed checking both the grammar and spelling of the given document when it was last saved. *end example*]

Attributes	Description
grammar (Grammatical Checking State)	Specifies if an application's grammar checking engine completed checking the grammatical content of the document when it was last saved. If this attribute is omitted, then its value is assumed to be `dirty` (not complete). [*Example*: Consider a WordprocessingML document saved by a hosting application whose spelling and grammar checking engines have completed checking grammar and spelling in the given WordprocessingML document. This state is specified using the following WordprocessingML in the document settings: `<w:proofState w:spelling="clean" w:grammar="clean" />` The grammar attribute has the value `clean` specifying that the hosting application's grammar checking engine completed checking the grammar of the given document when it was last saved. *end example*] The possible values for this attribute are defined by the ST_Proof simple type (§17.18.69).
spelling (Spell Checking State)	Specifies if an application's spell checking engine completed checking the spelling of the document when it was last saved. If this attribute is omitted, then its value is assumed to be `dirty` (not complete). [*Example*: Consider a WordprocessingML document saved by a hosting application whose spelling and grammar checking engines have completed checking grammar and spelling in the given WordprocessingML document. This state is specified using the following WordprocessingML in the document settings: `<w:proofState w:spelling="clean" w:grammar="clean" />` The spelling attribute has the value `clean` specifying that the hosting application's spell checking engine completed checking the spelling of the given document when it was last saved. *end example*] The possible values for this attribute are defined by the ST_Proof simple type (§17.18.69).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Proof) is located in §A.1. *end note*]

17.15.1.66 readModeInkLockDown (Freeze Document Layout)

This element specifies the exact set of page and text sizing parameters which shall be used to display the contents of a WordprocessingML document. [*Rationale*: This setting is typically used for documents that have been annotated using ink. This setting freezes the document's presentation such that the ink annotations must exist at the same position of the WordprocessingML document irrespective of the monitor on which the WordprocessingML document is rendered. *end rationale*]

This element shall only affect the display of WordprocessingML documents as follows:

- When the actualPage attribute is specified with a value of `true`, the given WordprocessingML document's pages shall be rendered as they would normally be displayed. The resulting pages can have their magnification setting changed as desired. All other attributes shall be ignored.
- When the actualPage attribute is specified with a value of `false`, the given WordprocessingML document's pages shall be rendered as *virtual pages* when loaded by a conforming hosting application irrespective of the given WordprocessingML document's view (§17.15.1.92). *Virtual pages* are pages with no correlation with the printed layout of a given WordprocessingML document that have been scaled by a conforming hosting application to improve the readability of a given WordprocessingML document when it is displayed. Specifically, the w and h attributes specify the width and height of the virtual pages, and the fontSz attribute specifies the scaling to be applied to text within the given WordprocessingML document.

[*Example*: Consider a WordprocessingML document that must be displayed using virtual pages when its contents are displayed. This state is specified using the following WordprocessingML in the document settings:

```
<w:readModeInkLockDown w:w="692" w:h="986" w:fontSz="95" w:actualPg="0"/>
```

The readModeInkLockDown element has w and h attribute values which specify the width and height of the virtual pages to be used to render the given WordprocessingML document. Finally, the fontSz attribute specifies the scaling to be applied to text within the given WordprocessingML document. *end example*]

Attributes	Description
actualPg (Use Actual Pages, Not Virtual Pages)	Specifies if applications shall render this WordprocessingML document with actual pages, not virtual pages. A*ctual pages* are pages rendered as they is printed. A value of `true` specifies that the given WordprocessingML document's pages is rendered as they are printed, and the w, h, and fontSz attributes shall be ignored. A value of `false` specifies that the given WordprocessingML document's pages shall be rendered as virtual pages using the other attributes on this element. [*Example*: Consider a WordprocessingML document that must be displayed using virtual pages. This state is specified using the following WordprocessingML in the document settings: `<w:readModeInkLockDown w:w="692" w:h="986" w:fontSz="95" w:actualPg="0" />`

Attributes	Description
	The actualPage attribute is equal to `0` specifying that the given WordprocessingML document must be rendered by conforming hosting applications using virtual pages. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
fontSz (Font Size Scaling)	Specifies the percentage that text in a given WordprocessingML document shall be scaled by before it is displayed on a virtual page, including a trailing percent sign (U+0025). This attribute shall only be used if the actualPage attribute equals `off`, `0`, or `false.` [*Example*: Consider a WordprocessingML document that must be displayed using virtual pages. This state is specified using the following WordprocessingML in the document settings: `<w:readModeInkLockDown w:w="692" w:h="986" w:fontSz="95%" w:actualPg="0" />` The fontSz attribute is equal to `95%` specifying that the text in the WordprocessingML document must be displayed at 95% of its normal size when it is displayed on a virtual page. *end example*] The possible values for this attribute are defined by the ST_DecimalNumberOrPercent simple type (§17.18.11).
h (Virtual Page Height)	Specifies the height of the virtual pages which shall be used in this document. This value is specified in pixels. This attribute shall only be used if the actualPage attribute equals `off`, `0`, or `false.` [*Example*: Consider a WordprocessingML document that must be displayed using virtual pages. This state is specified using the following WordprocessingML in the document settings: `<w:readModeInkLockDown w:w="692" w:h="986" w:fontSz="95" w:actualPg="0" />` The h attribute is equal to `986` specifying that virtual pages in this document must be 986 pixels high. *end example*] The possible values for this attribute are defined by the ST_PixelsMeasure simple type (§17.18.67).
w (Virtual Page Width)	Specifies the width of the virtual pages which shall be used in this document. This value is specified in pixels. This attribute shall only be used if the actualPage attribute equals `off`, `0`, or `false.`

Attributes	Description
	[*Example*: Consider a WordprocessingML document that must be displayed using virtual pages. This state is specified using the following WordprocessingML in the document settings: `<w:readModeInkLockDown w:w="692" w:h="986" w:fontSz="95" w:actualPg="0" />` The w attribute is equal to 692 specifying that virtual pages in this document must be 692 pixels wide. *end example*] The possible values for this attribute are defined by the ST_PixelsMeasure simple type (§17.18.67).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ReadingModeInkLockDown) is located in §A.1. *end note*]

17.15.1.67 removeDateAndTime (Remove Date and Time from Annotations)

This element specifies that the date and time information shall be removed from all annotations which are present in the current document when it is saved. Annotations store this information in the date attribute on the annotation's XML element.

If this element is omitted, then date information shall not be removed when the document is saved. If the removePersonalInformation element is not turned on, then this setting shall be ignored.

[*Example*: Consider a WordprocessingML document that must not save date and time information on annotations in the document content. This state is specified using the following WordprocessingML in the document settings:

```
<w:settings>
  …
  <w:removePersonalInformation w:val="true" />
  <w:removeDateAndTime w:val="true" />
  …
</w:settings>
```

The removeDateAndTime element's val attribute has a value of true specifying that all annotations in the document must have and date and time information removed before they are saved by omitting their date attributes. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.68 removePersonalInformation (Remove Personal Information from Document Properties)

This element specifies that hosting applications shall remove all personal information of document authors upon saving a given WordprocessingML document. The definition and extent of personal information is not defined by ISO/IEC 29500.

If this element is omitted, then personal information shall not be removed when the document is saved.

[*Example*: Consider a WordprocessingML document that must not save personal information in the document. This state is specified using the following WordprocessingML in the document settings:

```
<w:removePersonalInformation w:val="true" />
```

The removePersonalInformation element's val attribute has a value of true specifying that applications must remove any personal information when saving this file. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.69 revisionView (Visibility of Annotation Types)

This element specifies which forms of annotations shall be visible for a WordprocessingML document when it is displayed. This setting shall not affect whether annotations are added or persisted, it shall only affect the display of the annotations which exist in the document's contents (persisted or in memory).

If this element is omitted, then all forms of annotations shall be visible.

[*Example*: Consider the WordprocessingML below specifying that only formatting and ink annotations within a given WordprocessingML document must be displayed when the document is opened:

```
<w:revisionView w:markup="false" w:comments="false" w:insDel="false" />
```

The revisionView element specifies that the visibility of the markup region, comments and content additions/deletions must be suppressed by setting a value of false. Since the formatting and inkAnnotation attributes are omitted, they inherit the default of true and must be displayed. *end example*]

Attributes	Description
comments (Display Comments)	Specifies if comments should be included when the contents of this document are displayed. If this attribute is omitted, then comments shall be displayed when annotations are visible based on application-level settings. [*Example*: Consider the WordprocessingML below specifying that comments must be displayed: `<w:revisionView w:comments="true" />`

Attributes	Description
	The comments attribute has a value of `true`, specifying that comments must be rendered when the document's annotations are displayed. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
formatting (Display Formatting Revisions)	Specifies if revisions to properties (i.e. formatting revisions) should be included when the contents of this document are displayed. If this attribute is omitted, then formatting revisions shall be displayed when annotations are visible based on application-level settings. [*Example*: Consider the WordprocessingML below specifying that formatting revisions must be displayed: `<w:revisionView w:formatting="true" />` The formatting attribute has a value of `true`, specifying that formatting revisions must be rendered when the document's annotations are displayed. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
inkAnnotations (Display Ink Annotations)	Specifies if ink annotations should be included when the contents of this document are displayed. If this attribute is omitted, then ink annotations shall be displayed when annotations are visible based on application-level settings. [*Example*: Consider the WordprocessingML below specifying that ink annotations must be displayed: `<w:revisionView w:inkAnnotations="true" />` The inkAnnotations attribute has a value of `true`, specifying that ink annotations must be rendered when the document's annotations are displayed. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
insDel (Display Content Revisions)	Specifies if revisions to content (i.e. insertions, deletions, and moves) should be included when the contents of this document are displayed. If this attribute is omitted, then insertions, deletions, and moves shall be displayed when annotations are visible based on application-level settings. [*Example*: Consider the WordprocessingML below specifying that insertions, deletions, and moves must be displayed: `<w:revisionView w:insDel="true" />`

Attributes	Description
	The insDel attribute has a value of true, specifying that insertions, deletions, and moves must be rendered when the document's annotations are displayed. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
markup (Display Visual Indicator Of Markup Area)	Specifies if the application shall visually indicate any additional non-printing area used to display annotations when the annotations in this document are displayed. If this attribute is omitted, then any additional non-printing area shall be indicated when they are visible based on application-level settings. [*Example*: Consider the WordprocessingML below specifying that no visual indicator must be displayed for non-printing regions holding annotations: `<w:revisionView w:markup="false" />` The markup attribute has a value of false, specifying that nothing must be rendered indicating when a non-printing region is added when the document's annotations are displayed. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_TrackChangesView) is located in §A.1. *end note*]

17.15.1.70 rsid (Single Session Revision Save ID)

This element specifies the revision save ID that was associated with a single editing session for a document. An editing session is a span of time that begins and ends with any event that produces an editable file, such as a save or an e-mail send, and contains no such event. When revision save IDs are added to a document, they shall follow these rules:

- Every editing session shall be assigned a revision save ID that is larger than all earlier ones in the same file
- Revision save IDs should be randomly generated based on the current time (to minimize the chance that two disparate editing sessions starting with the same immediate predecessor are assigned the same revision save ID)
- Changes to document content in an editing session shall be stamped with the current revision save ID using the appropriate rsid* attributes
- An identical rsid value between two documents with the same rsidRoot (§17.15.1.71) shall indicate the same editing sessions

[*Note*: A revision save ID should be treated as unique within the context of all documents with the same rsidRoot value. Although in practice it is possible for two independent sessions to result in the same value, this outcome is extremely rare as the values are based on the current time. However, the meaning of two revision save IDs is not defined for documents with a different rsidRoot. Applications can use this information as desired. *end note*]

[*Example*: Consider the following fragments from two WordprocessingML documents' document settings:

Document 1	Document 2

`<w:rsids>`
`  <w:rsidRoot w:val="00464813"/>`
`  <w:rsid w:val="00455AAB" />`
`  <w:rsid w:val="00464813" />`
`  <w:rsid w:val="00996E03" />`
`</w:rsids>`
	`<w:rsids>`
	`  <w:rsidRoot w:val="00464813"/>`
	`  <w:rsid w:val="00455AAB" />`
	`  <w:rsid w:val="00464813" />`
	`  <w:rsid w:val="00473403" />`
	`  <w:rsid w:val="0048414E" />`
	`</w:rsids>`

The rsid elements are identical for the first three editing sessions for both documents, indicating that these documents, although they are now separate, originated from the same document. The documents were then separated and the first was saved once afterwards; and the second, twice. *end example*]

Attributes	Description
val (Long Hexadecimal Number Value)	Specifies a number value specified as a four digit hexadecimal number), whose contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following value for an attribute of simple type ST_LongHexNumber: `00BE2C6C`. This value is permitted, as it contains four hexadecimal digits, each an encoding of an octet of the actual decimal number value. It can therefore be interpreted as desired in the context of the parent XML element, *end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element's content model (CT_LongHexNumber) is located in §A.1. *end note*]

17.15.1.71 rsidRoot (Original Document Revision Save ID)

This element specifies the revision save ID which was associated with the first editing session for this document. [*Note*: This information must be identical between any number of copies of the same document, as they all originate from the same original editing session. Applications can use this information as desired. *end note*]

If this element is omitted, then the original document revision save ID is unknown.

[*Example*: Consider the following fragments from two WordprocessingML documents' document settings:

Document 1	Document 2

`<w:rsids>`
`  <w:rsidRoot w:val="00464813"/>`
`  <w:rsid w:val="00455AAB" />`
`  <w:rsid w:val="00464813" />`
`  <w:rsid w:val="00996E03" />`
`</w:rsids>`
	`<w:rsids>`
	`  <w:rsidRoot w:val="00464813"/>`
	`    <w:rsid w:val="00455AAB" />`
	`    <w:rsid w:val="00464813" />`
	`    <w:rsid w:val="00473403" />`
	`    <w:rsid w:val="0048414E" />`
	`</w:rsids>`

The rsidRoot element's val attribute has a value of `00464813` for both documents, indicating that these documents, although they are now separate, originated from the same document. This information can be used as desired. *end example*]

Attributes	Description
val (Long Hexadecimal Number Value)	Specifies a number value specified as a four digit hexadecimal number), whose contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following value for an attribute of simple type ST_LongHexNumber: `00BE2C6C`. This value is permitted, as it contains four hexadecimal digits, each an encoding of an octet of the actual decimal number value. It can therefore be interpreted as desired in the context of the parent XML element, *end example*] The possible values for this attribute are defined by the ST_LongHexNumber simple type (§17.18.50).

[*Note*: The W3C XML Schema definition of this element's content model (CT_LongHexNumber) is located in §A.1. *end note*]

17.15.1.72 rsids (Listing of All Revision Save ID Values)

This element specifies the set of revision save ID values for the current document. *Revision save ID values* refer to four digit hexadecimal values which uniquely identify an editing session in the life of the current document. An *editing session* is the period of time between two subsequent save operations by an application.

[*Guidance*: The set of revision save IDs stored with a document only supplies information about the editing session in which document components were last saved, which can be used by applications in any manner desired. *end guidance*]

If this element is omitted, then no information is available about the set of revision save ID values for this document.

[*Example*: Consider a WordprocessingML document with the following information present in its document settings:

```
<w:rsids>
  <w:rsidRoot w:val="00464813" />
  <w:rsid w:val="00455AAB" />
  <w:rsid w:val="00464813" />
  <w:rsid w:val="00473403" />
</w:rsids>
```

The rsids element contains four child elements, specifying that the document was edited over four distinct editing sessions (i.e. it was saved three times). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_DocRsids) is located in §A.1. *end note*]

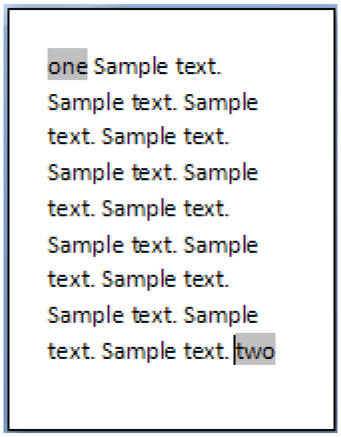
17.15.1.73 saveFormsData (Only Save Form Field Content)

This element specifies that saving the contents of this document shall only save the contents of WordprocessingML form fields defined using the FORMTEXT, FORMCHECKBOX, and FORMDROPDOWN field codes in a comma-delimited text format which does not conform to ISO/IEC 29500 (i.e. it is a one-way export from a WordprocessingML document).

[*Rationale*: This setting is typically used to allow duplication of paper forms in electronic WordprocessingML document form, allowing the resulting content to be extracted as a comma-delimited text file. *end rationale*]

If this element is omitted, then the contents of the entire document (not just form fields) should be saved according to the definition of WordprocessingML in ISO/IEC 29500.

[*Example*: Consider a WordprocessingML document which has form fields in the top right and bottom left corners of the first page, as follows (with the text box form fields shaded in grey):



one Sample text.
Sample text. Sample
text. Sample text.
Sample text. Sample
text. Sample text.
Sample text. Sample
text. Sample text.
Sample text. Sample
text. Sample text. two

If the only content which must be saved are the form fields' contents, this requirement is specified using the following WordprocessingML in the document settings:

```
<w:saveFormsData w:val="true" />
```

The saveFormsData element's val attribute as a value of `true`, specifying that only form field data must be saved, resulting in output as follows in a text file:

```
one,two
```

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.74 saveInvalidXml (Allow Saving Document As XML File When Custom XML Markup Is Invalid)

This element specifies that this document should be capable of being saved into a format consisting of a single XML file (not defined by ISO/IEC 29500) even when its contents are invalid based XML schema validation of the custom XML markup contained in the document. This setting has no effect on documents that do not contain custom XML markup, or that do contain custom XML markup but do not have a schema attached. [*Guidance*: Because this setting specifies behavior when saving to an alternative file format not defined by ISO/IEC 29500, this behavior is optional. *end guidance*]

If this element is omitted, then applications should not allow this document to be saved into a single XML file when its contents are invalid based on the custom XML markup contained in the document. If the doNotValidateAgainstSchema element (§17.15.1.43) is set, then the XML is never "invalid" and this property is ignored.

[*Example*: Consider a WordprocessingML document which should be saved into a single XML file even when its custom XML content is marked invalid by applications which support this operation. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:saveInvalidXml w:val="true" />
```

The saveInvalidXml element's val attribute has a value of `true` specifying that the content in this document can be saved regardless of its validation status. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.75 savePreviewPicture (Generate Thumbnail For Document On Save)

This element specifies if a document's Thumbnail part should be generated for the contents of the first page of this document when saved by application which support document thumbnail generation.

If this element is omitted, then applications can choose to save a thumbnail, however, that behavior is not required. If this element is specified, a thumbnail shall be produced if that functionality is supported.

[*Example*: Consider a WordprocessingML document that specifies that a document thumbnail must always be created when it is saved. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:savePreviewPicture w:val="true"/>
```

The savePreviewPicture element's val attribute has a value of `true` specifying that a document thumbnail should be generated each time this document is saved. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.76 saveThroughXslt (Custom XSL Transform To Use When Saving As XML File)

This element specifies the location of a custom XSL transform which shall be used when this document is saved as a single XML file (in a format not defined by ISO/IEC 29500). [*Guidance*: Because this setting specifies behavior when saving to an alternative file format not defined by ISO/IEC 29500, this behavior is optional. *end guidance*]

If this element is omitted, then no custom XSL transform shall be used when saving this file as a single XML file. If the useXSLTWhenSaving element (§17.15.1.91) is omitted or set to `false`, then this transform shall not be applied when the document is saved as a single XML file.

[*Example*: Consider a XML document that must have the XSL transform applied when the document is saved as a single XML file. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:useXSLTWhenSaving w:val="on"/>
<w:saveThroughXslt r:id="rId5" />
```

The useXSLTWhenSaving element's val is set to `true` indicating that applications must apply the XSLT specified by the relationship targeted by the id attribute of the saveThroughXslt element, located at `rId5`, when saving as a single XML file. *end example*]

Attributes	Description
id (XSL Transformation Location) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies an explicit relationship to the location of the XSL Transformation which shall be applied. The relationship targeted by this element shall be of type http://purl.oclc.org/ooxml/officeDocument/relationships/transform, or this document shall be declared non-conformant. [*Example*: Consider a XML document that must have the XSL transform located at `c:\Example Transform.xslt` applied when the document is saved as a single XML file. This requirement would be specified using the following WordprocessingML in the document settings: `<w:saveThroughXslt r:id="rId5" />` The saveThroughXslt element specifies that the relationship located at `rId5` must be used when saving as a single XML file in this case, that relationship must target `c:\Example Transform.xslt`. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
solutionID (Local Identifier for XSL Transform)	Specifies a string identifier which can be used to locate the XSL transform to be applied. The semantics of this attribute are not defined by ISO/IEC 29500 - applications can use this information in any application-defined manner to resolve the location of the XSL transform to apply. If this attribute is omitted, then no local identifier is specified for the XSL transform. If both this and the xslt attributes are present, then this data shall be used first, and the latter shall only be used if this information cannot be used successfully. [*Example*: Consider a XML document that must have the XSL transform identified by `mySolution` applied to when the document is saved as a single XML file. This requirement would be specified using the following WordprocessingML in the document settings: `<w:saveThroughXslt w:solutionID="mySolution" />` The solutionID attribute has a value of `mySolution` indicating that applications must apply the XSLT identified by this value (if known) when saving as a single XML file. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SaveThroughXslt) is located in §A.1. *end note*]

17.15.1.77 saveXmlDataOnly (Only Save Custom XML Markup)

This element specifies that the contents of this document shall be saved as an XML file containing only the custom XML markup in this document in its regular form. The resulting document does not conform to ISO/IEC 29500 (i.e. this is an export-only save option for a WordprocessingML document).

[*Rationale*: This setting is typically used to extract custom XML markup from a WordprocessingML document for further processing by XML-enabled applications. *end rationale*]

If this element is omitted, then the contents of the entire document (not just custom XML markup) should be saved according to the definition of WordprocessingML in ISO/IEC 29500.

[*Example*: Consider a WordprocessingML document which should be saved as an XML file containing only its custom XML markup. This requirement is specified using the following WordprocessingML fragment in the document settings:

```
<w:saveXmlDataOnly w:val="true"/>
```

The saveXmlDataOnly element's val attribute has a value of true specifying that only custom XML must be saved into a regular XML file when saving this document. For example, the document body formerly looked like this:

```
<w:body>
  <w:p>
    <w:customXml w:element="root" w:uri="urn:example">
      <w:r>
        <w:t>Hello, world</w:t>
      </w:r>
    </w:customXml>
  </w:p>
</w:body>
```

The presence of this element specifies that the resulting document only contains the custom Xml markup, resulting in the following:

```
<ns0:root xmlns:ns0="urn:example">Hello, world</ns0:root>
```

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.78 settings (Document Settings)

This element specifies the settings that are applied to a WordprocessingML document. This element is the root element of the Document Settings part in a WordprocessingML document.

[*Example*: Consider the following WordprocessingML fragment for the settings part of a document:

```
<w:settings>
  <w:defaultTabStop w:val="720" />
  <w:characterSpacingControl w:val="doNotCompress" />
</w:settings>
```

The settings element contains all of the settings for this document. In this case, the two settings applied are automatic tab stop increments of 0.5" using the defaultTabStop element, and no character level whitespace compression using the characterSpacingControl element. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Settings) is located in §A.1. *end note*]

17.15.1.79 showEnvelope (Show E-Mail Message Header)

This element specifies that an e-mail message header shall be displayed when this document is opened, if an e-mail header is supported by the application opening the file.

If this element is omitted, then applications shall not display the e-mail message header automatically when this file is opened, even if one is available in the application opening the file.

[*Example*: Consider a WordprocessingML document which should show an e-mail message header when opened. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:showEnvelope w:val="true" />
```

The showEnvelope element's val attribute has a value of `true` specifying that an e-mail message header must be displayed when the document is viewed, whenever such functionality is available. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.80 showXMLTags (Show Visual Indicators for Custom XML Markup Start/End Locations)

This element specifies that some visual indicator shall be provided for the start and end locations of custom XML markup present in this document, if any.

If this element is omitted, then applications should not provide any visual indicator of the locations of custom XML markup start/end tags.

[*Example*: Consider a WordprocessingML document which should show a visual indicator to the location of custom XML markup elements. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:showXMLTags w:val="true" />
```

The showXMLTags element's val attribute has a value of `true` specifying that custom XML markup should have a visual indicator in the document when displayed. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.1.81 smartTagType (Supplementary Smart Tag Information)

This element specifies optional supplementary information about one or more smart tags (§17.5.1.9) used in the current WordprocessingML document. This supplementary data is linked to the smart tag to which it applies via its name and namespaceuri attributes.

[*Example*: Consider a smart tag which has supplementary information defined as using the following WordprocessingML:

```
<w:smartTagType w:name="companyName" w:namespaceuri="urn:smartTagExample"
w:url="http://www.example.com/smartTag"/>
```

The name and namespaceuri attributes specify that the smart tag to which this data must be `companyName` in the `urn:smartTagExample` namespace. The supplementary data is an associated URL of `http://www.example.com/smartTag`. *end example*]

Attributes	Description
name (Smart Tag Name)	Specifies the name of the smart tag within the document for which supplementary data is provided. [*Example*: Consider a smart tag which has a name of `companyName`. This name would be referenced using the following WordprocessingML: `<w:smartTagType w:name="companyName" … />` The name attribute specifies that the name for this smart tag must be `companyName`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
namespaceuri (Smart Tag Namespace)	Specifies the namespace URI of the smart tag for which supplementary data is provided. If this attribute is omitted, the URI shall be assumed to be null (no associated URI). [*Example*: Consider a smart tag which must have a namespace URI of `urn:smartTagExample`. This namespace would be referenced using the following WordprocessingML: `<w:smartTagType w:namespaceuri="urn:smartTagExample" />` The namespaceuri attribute specifies that the namespace for the smart tag to which this data applies must be `urn:smartTagExample`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
url (Smart Tag Supplementary URL)	Specifies a URL provided for a particular smart tag type in this document. [*Note*: This URL is typically used to provide access to a URL for additional updates to this smart tag type as requested by the smart tag provider. *end note*] If this attribute is omitted, then no supplementary URL is provided for this type. [*Example*: Consider a smart tag which must have a supplementary URL of `http://www.example.com/smartTag`. This URL would be specified using the following WordprocessingML: `<w:smartTagType … w:url="http://www.example.com/smartTag" />` The url attribute specifies that the supplementary data for the smart tag to which this data applies must be `http://www.example.com/smartTag`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SmartTagType) is located in §A.1. *end note*]

17.15.1.82 strictFirstAndLastChars (Use Strict Kinsoku Rules for Japanese Text)

This element specifies that the strict set of Kinsoku rules shall be applied to Japanese text in this document when the kinsoku element (§17.3.1.16) is applied to that text. The resulting line breaking rules are provided on the kinsoku element.

If this element is omitted, then standard rules shall apply to Japanese text when the kinsoku element is applied to that text.

[*Example*: Consider a WordprocessingML document that specifies that strict Kinsoku rules must be applied to Japanese text. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:strictFirstAndLastChars w:val="true"/>
```

The strictFirstAndLastChars element's val attribute has a value of `true` specifying that a document must apply the strict set of disallowed characters for the start and end of a line. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.83 styleLockQFSet (Prevent Replacement of Styles Part)

This element specifies whether applications shall prevent the replacement of the complete set of styles stored in the Styles part when editing this document. This setting should not preclude the editing or removal of individual styles, instead, it should only prevent the removal and replacement of the entire styles part in a single operation (either through a user interface or a programmatic operation).

If this element is omitted, then applications can allow the replacement of the entire styles part in this document.

[*Example*: Consider a WordprocessingML document that specifies that applications must prevent the replacement of the entire styles part. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:styleLockQFSet w:val="true"/>
```

The styleLockQFSet element's val attribute has a value of `true` specifying that individual style changes should be allowed, but the styles data must not be replaced as a whole via a single operation. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.84 styleLockTheme (Prevent Modification of Themes Part)

This element specifies whether applications shall prevent the modification of the document's theme information stored in the Theme part when editing this document. This setting should not preclude the use of the theme information, instead, it should only prevent the modification of the theme part in a single operation (either through a user interface or a programmatic operation).

If this element is omitted, then applications can allow the replacement or modification of the theme part in this document.

[*Example*: Consider a WordprocessingML document that specifies that applications must prevent the modification of the theme part. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:styleLockTheme w:val="true"/>
```

The styleLockTheme element's val attribute has a value of `true` specifying that theme data must not be modified when modifying the contents of this document. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.85 stylePaneFormatFilter (Suggested Filtering for List of Document Styles)

This element specifies a set of suggested filters which should be applied to the list of document styles in this application if the styles are displayed in a user interface.

If this element is omitted, then all settings defined by this element are turned off.

[*Example*: Consider a document with the following value in its document settings:

```
<w:stylePaneFormatFilter w:customStyles="true" w:top3HeadingStyles="true" />
```

The stylePaneFormatFilter element's settings specify two suggested filter options for the list of document styles:

- Only custom styles should be shown
- Heading styles with a styleId of `Heading1` to `Heading3` should always be displayed in the list

end example]

Attributes	Description
allStyles (Display All Styles)	Specifies that all styles present in the Styles part should be displayed in the list of document styles. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
alternateStyleNames (Use the Alternate Style Name)	Specifies that primary names for styles should not be shown if an alternate name using the name element (§17.7.4.9) exists. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
clearFormatting (Display Styles to	Specifies that a style should be present which removes all formatting and styles from text.

Attributes	Description
Remove Formatting)	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
customStyles (Display Only Custom Styles)	Specifies that only styles with the customStyle attribute should be displayed in the list of document styles. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
directFormattingOnNumbering (Display Direct Formatting on Numbering Data)	Specifies that all unique forms of direct formatting of numbering data should be displayed in the list of document styles as though they were each a unique style. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
directFormattingOnParagraphs (Display Paragraph Level Direct Formatting)	Specifies that all unique forms of paragraph-level direct formatting should be displayed in the list of document styles as though they were each a unique style. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
directFormattingOnRuns (Display Run Level Direct Formatting)	Specifies that all unique forms of run-level direct formatting should be displayed in the list of document styles as though they were each a unique style. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
directFormattingOnTables (Display Direct Formatting on Tables)	Specifies that all unique forms of direct formatting of tables should be displayed in the list of document styles as though they were each a unique style. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
headingStyles (Display Heading Styles)	Specifies that heading styles (styles with a styleId of Heading1 to Heading9) should be displayed in the list of document styles when the previous style is used in the document and/or is present in the Styles part. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
latentStyles (Display Latent Styles)	Specifies that all latent styles should be displayed in the list of document styles. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
numberingStyles (Display Numbering Styles)	Specifies that numbering styles should be displayed in the list of document styles. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
stylesInUse (Display Styles in Use)	Specifies that only styles used in the document should be displayed in the list of document styles. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
tableStyles (Display Table Styles)	Specifies that table styles should be displayed in the list of document styles. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

Attributes	Description
top3HeadingStyles (Display Heading 1 through 3)	Specifies that heading styles with a styleId of `Heading1` to `Heading3` should always be displayed in the list of document styles. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).
visibleStyles (Only Show Visible Styles)	Specifies that styles should only be shown if the semiHidden element (§17.7.4.16) is false and the hidden element (§17.7.4.4) is false. The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_StylePaneFilter) is located in §A.1. *end note*]

17.15.1.86 stylePaneSortMethod (Suggested Sorting for List of Document Styles)

This element specifies a sorting which should be applied to the list of styles in this document if the styles are displayed in a user interface.

If this element is omitted, then styles which are visible should be sorted by the default sorting of the host application.

[*Example*: Consider a document with the following value in its document settings:

```
<w:stylePaneSortMethod w:val="type" />
```

The stylePaneFormatFilter element's val attribute specifies that styles which are visible should be sorted by their style types (i.e. character, linked, paragraph) via a value of `type`. *end example*]

Attributes	Description
val (Style Sorting)	Specifies a sort order which should be applied to the list of document styles when they are displayed in a user interface. The possible values for this attribute are defined by the ST_StyleSort simple type (§17.18.82).

[*Note*: The W3C XML Schema definition of this element's content model (CT_StyleSort) is located in §A.1. *end note*]

17.15.1.87 summaryLength (Percentage of Document to Use When Generating Summary)

This element specifies the size for automatic document summaries performed on the content of a WordprocessingML document. An *automatic document summary* is a subset of text contained in a document deemed by the hosting application to summarize the content of the WordprocessingML document. The val attribute of this element specifies the size of an automatic document summary to be performed on a given

WordprocessingML document as a percentage of the total size of the given WordprocessingML document. Performing an automatic document summary is a runtime operation outside the scope of ISO/IEC 29500.

If this element is omitted, then applications can summarize this document to any desired size.

[*Example*: Consider a WordprocessingML document whose automatic document summary is to be ten percent of the size of the given WordprocessingML document. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:summaryLength w:val="10%" />
```

The summaryLength element's val attribute is equal to 10% specifying that any automatic document summary is to be ten percent of the size of the document. *end example*]

Attributes	Description
val (Value in Percent)	Specifies a measurement in whole percentage points, with a trailing percent sign (U+0025). [*Example*: Consider the following WordprocessingML fragment: `<… w:val="50.125%" />` This value specifies a value of fifty and one-eighth percent. *end example*] The possible values for this attribute are defined by the ST_DecimalNumberOrPercent simple type (§17.18.11).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumberOrPrecent) is located in §A.1. *end note*]

17.15.1.88 themeFontLang (Theme Font Languages)

This element specifies the language which shall be used to determine the appropriate theme fonts in the document's Theme part which map to the major/minor theme fonts. Specifically, the bidi attribute is used to determine the theme font applied to complex script text, the eastAsia attribute is used to determine the theme font applied to East Asian text, and the val attribute is used to determine the theme font applied to all other text.

These mappings are performed as follows:

- For majorAscii/majorHAnsi, locate the font element (§20.1.4.1.16) in the majorFont element (§20.1.4.1.24) in the theme part for the language specified by the val attribute
- For majorBidi, locate the font element in the majorFont element in the theme part for the language specified by the bidi attribute

- For `majorEastAsia`, locate the font element in the majorFont element in the theme part for the language specified by the eastAsia attribute
- For `minorAscii`/`minorHAnsi`, locate the font element in the minorFont element (§20.1.4.1.25) in the theme part for the language specified by the val attribute
- For `minorBidi`, locate the font element in the minorFont element in the theme part for the language specified by the bidi attribute
- For `minorEastAsia`, locate the font element in the minorFont element in the theme part for the language specified by the eastAsia attribute

If this element is omitted, then the default fonts for each region as specified by the latin, ea, and cs elements (§21.1.2.3.7; §21.1.2.3.3; §21.1.2.3.1) should be used.

[*Example*: Consider a document with the following WordprocessingML in its document settings:

```
<w:themeFontLang w:val="ja-JP" />
```

The themeFontLang element's val attribute has a value of `ja-JP`, specifying that the theme fonts used for Latin text must be the theme fonts for Japanese. If the following content was present in the theme part:

```
…
<a:majorFont>
  …
  <a:font script="Jpan" typeface="MS Mincho"/>
  …
</a:majorFont>
…
```

Then this setting would specify that uses of the `majorAscii` and `majorHAnsi` theme font enumerations must be mapped to the `MS Mincho` font. *end example*]

Attributes	Description

bidi (Complex Script Language)	Specifies the language which shall be used when processing the contents of this run which use complex script characters, as determined by the Unicode character values of the run content.
	If this attribute is omitted, then the languages for the contents of this run using complex script characters shall be automatically determined based on their contents using any appropriate method.
	[*Example*: Consider a run which contains complex script characters in its contents. If those contents should be interpreted as Hebrew, that requirement would be specified as follows in the resulting WordprocessingML:
	`<w:r>`
	`  <w:rPr>`
	`    <w:lang w:bidi="he-IL" />`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
    </w:rPr>
  </w:r>
</pre>
<p>The resulting run specifies that any complex script contents must be spell and grammar checked using a Hebrew dictionary and grammar engine, if one is available. <em>end example</em>]</p>
<p>The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).</p>
</td></tr>
<tr><td>eastAsia (East Asian Language)</td><td>
<p>Specifies the language which shall be used when processing the contents of this run which use East Asian characters, as determined by the Unicode character values of the run content.</p>
<p>If this attribute is omitted, then the languages for the contents of this run using East Asian characters shall be automatically determined based on their contents using any appropriate method.</p>
<p>[<em>Example</em>: Consider a run which contains East Asian characters in its contents. If those contents should be interpreted as Korean, that requirement would be specified as follows in the resulting WordprocessingML:</p>
<pre>
  <w:r>
    <w:rPr>
      <w:lang w:eastAsia="ko-KR" />
    </w:rPr>
  </w:r>
</pre>
<p>The resulting run specifies that any complex script contents must be spell and grammar checked using a Korean dictionary and grammar engine, if one is available. <em>end example</em>]</p>
<p>The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).</p>
</td></tr>
<tr><td>val (Latin Language)</td><td>
<p>Specifies the language which shall be used to check spelling and grammar (if requested) when processing the contents of this run which use Latin characters, as determined by the Unicode character values of the run content.</p>
<p>If this attribute is omitted, then the languages for the contents of this run using Latin characters shall be automatically determined based on their contents using any appropriate method.</p>
<p>[<em>Example</em>: Consider a run which contains Latin characters in its contents. If those contents should be interpreted as English (Canada), that requirement would be specified as follows in the resulting WordprocessingML:</p>
<pre>
  <w:r>
    <w:rPr>
      <w:lang w:val="en-CA" />
    </w:rPr>
  </w:r>
</pre>
</td></tr>
</table>

Attributes	Description
	The resulting run specifies that any complex script contents must be spell and grammar checked using a English (Canada) dictionary and grammar engine, if one is available. *end example*] The possible values for this attribute are defined by the ST_Lang simple type (§22.9.2.6).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Language) is located in §A.1. *end note*]

17.15.1.89 trackRevisions (Track Revisions to Document)

This element specifies that applications shall track revisions made to the WordprocessingML document. *Revisions* are changes to a WordprocessingML document which are recorded such that they can be viewed independently, accepted or removed, and reverted if needed. When revisions are tracked, the resulting WordprocessingML markup in the Revisions subclause of this document describes the necessary syntax.

If this element is omitted, then revisions shall not be generated by changes to the contents of this document.

[*Example*: Consider a WordprocessingML document containing the text run `Example` that must not have revisions tracked. Example WordprocessingML from Document 1 is given below:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

And the corresponding document settings:

```
<w:settings>
  <w:trackRevisions w:val="false"/>
  …
</w:settings>
```

If the word `text` was added to the end of this document and bolded without revisions tracked, the resulting WordprocessingML would be output as follows:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
      <w:r>
        <w:rPr>
          <w:b/>
        </w:rPr>
        <w:t>text</w:t>
      </w:r>
    </w:p>
  </w:body>
</w:document>
```

And the corresponding document settings:

```
<w:settings>
  <w:trackRevisions w:val="false"/>
  …
</w:settings>
```

Finally, assume the same insertion and formatting took place when the trackRevisions element's val attribute was set to true, the resulting WordprocessingML would be output as follows:

```
<w:document>
  <w:body>
    <w:p>
      <w:r>
        <w:t>Example</w:t>
      </w:r>
```

```
        <w:ins … >
          <w:r>
            <w:rPr>
              <w:b/>
              <w:rPrChange … >
                <w:rPr/>
              </w:rPrChange>
            </w:rPr>
            <w:t>text</w:t>
          </w:r>
        </w:ins>
      </w:p>
    </w:body>
  </w:document>
```

And the corresponding document settings:

```
  <w:settings>
    <w:trackRevisions w:val="true"/>
    …
  </w:settings>
```

The trackRevisions element's val attribute was set to true, therefore the changes to the content of the document were inserted using the appropriate annotation elements in the document's WordprocessingML. Specifically, inserting the text Text to the right of the existing text was tracked as a revision with the ins element. In addition, applying bold formatting to the text was tracked as a revision with the rPrChange element. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.90 updateFields (Automatically Recalculate Fields on Open)

This element specifies whether the fields contained in this document should automatically have their field result recalculated from the field codes when this document is opened by an application which supports field calculations. [*Note*: Some fields are always recalculated (e.g. the page numbering), therefore this element only affects fields which are typically not automatically recalculated on opening the document. Also note that this setting must not supersede any document protection (§17.15.1.29) or write protection (§17.15.1.93) settings. *end note*]

If this element is omitted, then fields should not automatically be recalculated on opening this document.

[*Example*: Consider a WordprocessingML document that specifies that applications should attempt to automatically recalculate fields from their field codes upon opening this document. This requirement would be specified using the following WordprocessingML in the document settings part:

```
  <w:updateFields w:val="true"/>
```

The updateFields element's val attribute has a value of `true` specifying that all fields should automatically be recalculated when opening this document. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.91 useXSLTWhenSaving (Save Document as XML File through Custom XSL Transform)

This element specifies that this document should be saved through the custom XSLT transform defined by the saveThroughXslt element (§17.15.1.76) in this document when it is saved as a single XML file (not defined by ISO/IEC 29500). [*Guidance*: Because this setting specifies behavior when saving to an alternative file format not defined by ISO/IEC 29500, this behavior is optional. *end guidance*]

If the saveXmlDataOnly element (§17.15.1.77) is specified, then the single XML file to be transformed is the custom XML markup of the document, otherwise, it a format outside the scope of ISO/IEC 29500. If the XSL transform specified by the saveThroughXslt element is not present, then this setting should be ignored.

If this element is omitted, then this document should not be saved through a custom XSL transform when it is saved as a single XML file.

[*Example*: Consider a WordprocessingML document which should be saved through a custom XSL transform when it is saved as a single XML file. This requirement is specified using the following WordprocessingML in the document settings:

```
<w:useXSLTWhenSaving w:val="true" />
```

The useXSLTWhenSaving element's val attribute has a value of `true` specifying that the content in this document should be saved as a single XML file through the custom XSLT specified by the saveThroughXslt element. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.1.92 view (Document View Setting)

This element specifies the manner in which the contents of this document should be displayed when opened by an application.

If this element is omitted, then an application can view the document in any desired default state.

[*Example*: Consider a WordprocessingML document that must be displayed on the screen in the same form as it is printed. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:view w:val="print" />
```

The view element's val attribute is equal to `print` specifying that the given WordprocessingML document must be rendered as it is printed. *end example*]

Attributes	Description
val (Document View Setting Value)	Specifies the view which shall be used to render the contents of a WordprocessingML document. Applications can omit support for one or more of the views defined by the ST_View simple type (referenced below). If a WordprocessingML document containing an unsupported view is loaded by an application, it shall fall back to its default view (equivalent to use of the enumeration value none). [*Example*: Consider a WordprocessingML document that must be rendered in a view meant to mimic how the document would look in a web browser (i.e. without a fixed page width). This requirement would be specified using the following WordprocessingML in the document settings: `<w:view w:val="web" />` The val attribute is equal to web specifying that the given WordprocessingML document must be rendered in a view mimicking web page display. *end example*] The possible values for this attribute are defined by the ST_View simple type (§17.18.102).

[*Note*: The W3C XML Schema definition of this element's content model (CT_View) is located in §A.1. *end note*]

17.15.1.93 writeProtection (Write Protection)

This element specifies the write protection settings which have been applied to a WordprocessingML document. *Write protection* refers to a mode in which the document's contents cannot be edited, and the document cannot be resaved using the same file name. This setting is independent of the documentProtection (§17.15.1.29) element, but like document protection, this setting is not intended as a security feature and can be ignored.

When present, the write protection shall result in one of two write protection behaviors:

- If the password attribute is present, or both attributes are omitted, then the application shall prompt for a password to exit write protection. If the supplied password does not match the hash value in this attribute, then write protection shall be enabled.
- If only the recommended attribute is present, the application should provide user interface recommending that the user open this document in write protected state. If the user chooses to do so, the document shall be write protected, otherwise, it shall be opened fully editable.

If this element is omitted, then no write protection shall be applied to the current document.

[*Example*: Consider a WordprocessingML document that can be opened but only in a write protected state unless a password is provided, in which case the file would be opened in an editable state. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:writeProtection w:hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />
```

If the attributes specified in the password attribute group AG_Password are present, then the application shall require a password to exit write protection. If the hash value derived from supplied password does not match the hash value in the attribute hashValue, then write protection shall be enabled. *end example*]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>algorithmName (Cryptographic Algorithm Name)</td><td>Specifies the specific cryptographic hashing algorithm which shall be used along with the salt attribute and input password in order to compute the hash value.
The following values are reserved:
<table>
<tr><th>Value</th><th>Algorithm</th></tr>
<tr><td>MD2</td><td>Specifies that the MD2 algorithm, as defined by RFC 1319, shall be used.
[<i>Note</i>: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. <i>end note</i>]</td></tr>
<tr><td>MD4</td><td>Specifies that the MD4 algorithm, as defined by RFC 1320, shall be used.
[<i>Note</i>: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. <i>end note</i>]</td></tr>
<tr><td>MD5</td><td>Specifies that the MD5 algorithm, as defined by RFC 1321, shall be used.
[<i>Note</i>: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. <i>end note</i>]</td></tr>
<tr><td>RIPEMD-128</td><td>Specifies that the RIPEMD-128 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.
[<i>Note</i>: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. <i>end note</i>]</td></tr>
<tr><td>RIPEMD-160</td><td>Specifies that the RIPEMD-160 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr>
<tr><td>SHA-1</td><td>Specifies that the SHA-1 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr>
<tr><td>SHA-256</td><td>Specifies that the SHA-256 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr>
<tr><td>SHA-384</td><td>Specifies that the SHA-384 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td><table><tr><td>SHA-512</td><td>Specifies that the SHA-512 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr><tr><td>WHIRLPOOL</td><td>Specifies that the WHIRLPOOL algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.</td></tr></table>
[*Example*: Consider an Office Open XML document with the following information stored in one of its protection elements:
`< … algorithmName="SHA-1"`
`  hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
The algorithmName attribute value of "`SHA-1`" specifies that the SHA-1 hashing algorithm must be used to generate a hash from the user-defined password. *end example*]
The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).</td></tr>
<tr><td>hashValue (Password Hash Value)</td><td>Specifies the hash value for the password stored with this document. This value shall be compared with the resulting hash value after hashing the user-supplied password using the algorithm specified by the preceding attributes and parent XML element, and if the two values match, the protection shall no longer be enforced.
If this value is omitted, then the reservationPassword attribute shall contain the password hash for the workbook.
[*Example*: Consider an Office Open XML document with the following information stored in one of its protection elements:
`<… AlgorithmName="SHA-1"`
`  hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
The hashValue attribute value of `9oN7nWkCAyEZib1RomSJTjmPpCY=` specifies that the user-supplied password must be hashed using the pre-processing defined by the parent element (if any) followed by the SHA-1 algorithm (specified via the algorithmName attribute value of `SHA-1`) and that the resulting has value must be `9oN7nWkCAyEZib1RomSJTjmPpCY=` for the protection to be disabled. *end example*]
The possible values for this attribute are defined by the W3C XML Schema base64Binary datatype.</td></tr>
<tr><td>recommended (Recommend Write Protection in User Interface)</td><td>Specifies that applications should provide user interface recommending that the user open this document in write protected state. If the user chooses to do so, the document shall be write protected, otherwise, it shall be opened fully editable.
If this attribute is omitted, then user interface recommending that the user open this document in write protected state should not be provided. If the password attribute is</td></tr>
</table>

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	also specified, then this setting shall be ignored. [*Example*: Consider a WordprocessingML document which specifies that applications must recommend write protection to this document. This requirement would be specified using the following WordprocessingML in the document settings: `<w:writeProtection w:recommended="true" />` The recommended attribute has a value of `true` specifying that the applications must hash any password provided, and if it matches this hash value, can only then halt enforcement of write protection. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

saltValue (Salt Value for Password Verifier)	Specifies the salt which was prepended to the user-supplied password before it was hashed using the hashing algorithm defined by the preceding attribute values to generate the hashValue attribute, and which shall also be prepended to the user-supplied password before attempting to generate a hash value for comparison. A *salt* is a random string which is added to a user-supplied password before it is hashed in order to prevent a malicious party from pre-calculating all possible password/hash combinations and simply using those pre-calculated values (often referred to as a "dictionary attack").
	If this attribute is omitted, then no salt shall be prepended to the user-supplied password before it is hashed for comparison with the stored hash value.
	[*Example*: Consider an Office Open XML document with the following information stored in one of its protection elements:
	`<… saltValue="ZUdHa+D8F/OAKP3I7ssUnQ=="`
	`  hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />`
	The saltValue attribute value of `ZUdHa+D8F/OAKP3I7ssUnQ==` specifies that the user-supplied password must have this value prepended before it is run through the specified hashing algorithm to generate a resulting hash value for comparison. *end example*]
	The possible values for this attribute are defined by the W3C XML Schema base64Binary datatype.

spinCount (Iterations to Run Hashing Algorithm)	Specifies the number of times the hashing function shall be iteratively run (runs using each iteration's result plus a 4 byte value (0-based, little endian) containing the number of the iteration as the input for the next iteration) when attempting to compare a user-supplied password with the value stored in the hashValue attribute. [*Rationale*: Running the algorithm many times increases the cost of exhaustive search attacks correspondingly. Storing this value allows for the number of iterations to be increased over time to accommodate faster hardware (and hence the ability to run more iterations in less time). *end rationale*]

Attributes	Description
	[*Example*: Consider an Office Open XML document with the following information stored in one of its protection elements: `<… spinCount="100000" hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The spinCount attribute value of `100000` specifies that the hashing function must be run one hundred thousand times to generate a hash value for comparison with the hashValue attribute. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_WriteProtection) is located in §A.1. *end note*]

17.15.1.94 zoom (Magnification Setting)

This element specifies the magnification level which should be applied to a document when it is displayed by an application. The zoom level is specified with the use of two attributes stored on this element:

- val, which stores the type of zoom applied to the document
- percent, which stores the zoom percentage to be used when rendering the document

If both attributes are present, then the percent attribute shall be treated as a 'cached' value and only used when the value none is specified for the val attribute.

If this element is omitted, then applications can display the document in any desired magnification setting.

[*Example*: Consider a WordprocessingML document that is to have its zoom level at seventy one percent when it is displayed. This requirement would be specified using the following WordprocessingML fragment in the document settings:

```
<w:zoom w:percent="71%" />
```

The zoom element's percent attribute has a value of `71%`, specifying that the given document must have its zoom level set to seventy one percent when it is displayed. *end example*]

Attributes	Description
percent (Zoom Percentage)	Specifies the zoom percentage that should be applied when a given WordprocessingML document is rendered by conforming hosting applications. This value is the zoom percentage specified as an integer with a trailing percent sign (U+0025). If this attribute is omitted, then applications can use any desired default percentage for

Attributes	Description
	the magnification. If the val attribute instantiated in addition to the percent attribute, then the percent attribute shall be treated as a cached value and only used when the value none is specified for the val attribute. If the value specified exceeds the maximum zoom level available in a conforming hosting application, the conforming hosting application shall display the document using its maximum zoom level. Correspondingly, if the value specified is less than the minimum zoom level available in the conforming hosting application, the conforming hosting application shall display the document using its minimum zoom level. [*Example*: Consider a WordprocessingML document that is to have its zoom level at fifty percent when rendered by conforming hosting applications. This requirement would be specified using the following WordprocessingML: `<w:zoom w:percent="50%" />` The percent attribute has a value of `50%`, specifying that the given WordprocessingML document must to have its zoom level set to fifty percent when it is displayed. *end example*] The possible values for this attribute are defined by the ST_DecimalNumberOrPercent simple type (§17.18.11).
val (Zoom Type)	Specifies the type of zoom which shall be applied to a given document on open. If this attribute is not present, then the document shall be displayed as though the value had been set to none, and should rely on the value of the percent attribute for the actual zoom percentage. [*Example*: Consider a WordprocessingML document that should be visible without any horizontal scrolling when it is displayed. This requirement would be specified using the following WordprocessingML: `<w:zoom w:val="bestFit" w:percent="90%" />` The val attribute is equal to the value `bestFit` specifying that an application must dynamically calculate the magnification needed such that the given document must be visible on the horizontal plane of the document with no horizontal scrolling required to see any part of the WordprocessingML document's pages. Since both attributes are present, the percent attribute must be treated as a 'cached' value and ignored. *end example*] The possible values for this attribute are defined by the ST_Zoom simple type (§17.18.105).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Zoom) is located in §A.1. *end note*]

17.15.2 Web Page Settings

The next group of settings stored in WordprocessingML is web page settings. These settings specify two categories of settings:

- Settings which are related to HTML documents (i.e. frameset definitions) that can be used in WordprocessingML documents as well
- All settings which affect how this document shall be handled when it is saved as HTML. Actually saving a document as HTML is outside of the scope of ISO/IEC 29500, but in order to ensure the maximum interoperability between a WordprocessingML document and an HTML document, settings not explicitly stored elsewhere are stored in these settings.

[*Example*: Consider the following WordprocessingML fragment for the web page settings in a WordprocessingML document:

```
<w:webSettings>
  <w:frameset>
    …
  </w:frameset>
  <w:doNotUseLongFileNames w:val="true" />
</w:webSettings>
```

The webSettings element contains all of the web page settings for this document. In this case, the web page settings specified for this document are: a frameset defined using the frameset element (§17.15.2.19); and a setting specifying that when this file is saved as a web page, all resulting files must not exceed 8 octets with 3 octet extension using the doNotUseLongFileNames element (§17.15.2.13). *end example*]

17.15.2.1 allowPNG (Allow PNG as Graphic Format)

This element specifies that applications shall allow use of the PNG file format when the contents of this WordprocessingML document are saved as a web page. This includes all supporting images used as part of this HTML web page.

If this element is omitted from the document, then the PNG file format shall not be allowed when this document is saved as a web page, and that another suitable file format (such as the JPEG file format) should be utilized in its place.

[*Note*: This setting is intended for applications to save web pages which can be supported by legacy web browsers which do not support the reading of PNG images. However, although PNG utilizes a lossless compression algorithm, JPEG uses 'lossy' compression and can in some cases result in lower fidelity images. *end note*]

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:allowPNG w:val="true" />
</w:webSettings>
```

The allowPNG element has a val attribute value of true, which specifies that applications can use the PNG graphic format as needed when saving this WordprocessingML document as a web page. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.2.2 blockQuote (Data for HTML blockquote Element)

This element specifies that the current div element does not represent an HTML div element, but rather represents an HTML blockquote element. This element shall specify that this container shall be written out using the blockquote element if this document is subsequently saved as HTML.

If this element is omitted, then the current div element does not represent an HTML blockquote element. If both this element and the bodyDiv element (§17.15.2.3) are specified, then this element shall take precedence in all cases.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body style="margin-left:200px;margin-top:50px">
    <p>Paragraph one.</p>
    <blockquote style="border: 5px solid #00FFFF">
      <p>Paragraph in a blockquote.</p>
    </blockquote>
    <p>Paragraph two.</p>
  </body>
</html>
```

When this document is saved in the WordprocessingML format, the information stored on the div, blockquote, and body elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1626542603">
    …
    <w:divsChild>
      <w:div w:id="313534916">
        <w:blockQuote w:val="true" />
        …
      </w:div>
    </w:divsChild>
  </w:div>
</w:divs>
```

The blockQuote element has a val attribute value of true, which specifies that the nested div element actually represents a nested HTML blockquote when this document is resaved as HTML. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.2.3 bodyDiv (Data for HTML body Element)

This element specifies that the current div element does not represent an HTML div element, but rather represents formatting properties on the HTML body element. This element shall specify that the properties specified by this container shall be written out onto the body element if this document is subsequently saved as HTML.

If this element is omitted, then the current div element does not represent an HTML body element. If both this element and the blockQuote element (§17.15.2.2) are specified, then this element shall be ignored. If this element is specified on any div which is not the main div element for the document, then this element shall be ignored.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body style="margin-left:200px;margin-top:50px">
    <p>Paragraph one.</p>
    <blockquote style="border: 5px solid #00FFFF">
      <p>Paragraph in a blockquote.</p>
    </blockquote>
    <p>Paragraph two.</p>
  </body>
</html>
```

When this document is saved in the WordprocessingML format, the information stored on the div, blockquote, and body elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1626542603">
    <w:bodyDiv w:val="true" />
    …
    <w:divsChild>
      …
    </w:divsChild>
  </w:div>
</w:divs>
```

The bodyDiv element has a val attribute value of true, which specifies that the div element actually represents properties on the HTML body when this document is resaved as HTML. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.2.4 bottom (Bottom Border for HTML div)

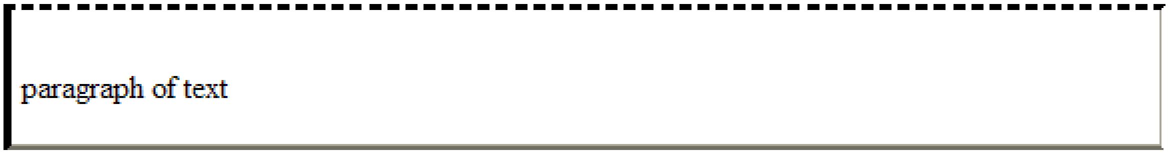
This element specifies the border which shall be displayed at the bottom of the boundaries of the current HTML div object.

If this element is omitted, then this HTML div object shall not have a bottom border.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" border-left-style:solid; border-right-style:groove; border-right-width:1px; border-top-style:dashed; border-top-width:3px; border-bottom-style:outset; border-bottom-width:3px">
      <p>paragraph of text</p>
    </div>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



Now, when this document is saved in the WordprocessingML format, the information stored on the div elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    …
    <w:divBdr>
      <w:top w:val="dashed" w:sz="18" w:space="7" w:color="auto" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="auto" />
      <w:bottom w:val="outset" w:sz="18" w:color="auto" />
      <w:right w:val="threeDEngrave" w:sz="6" w:color="auto" />
    </w:divBdr>
  </w:div>
</w:divs>
```

The bottom element specifies border information about the bottom border for the single HTML div structure in the document; in this case, a 2.25 point bottom border of type outset. The initial 3 pixel border was converted to 2.25 points using the following logic:

$$3\text{px} * \frac{1\ \text{inch}}{96\ \text{px}} * \frac{576\ \text{eighth points}}{1\ \text{inch}} = 18\ \text{eighth points (2.25 points)}$$

end example]

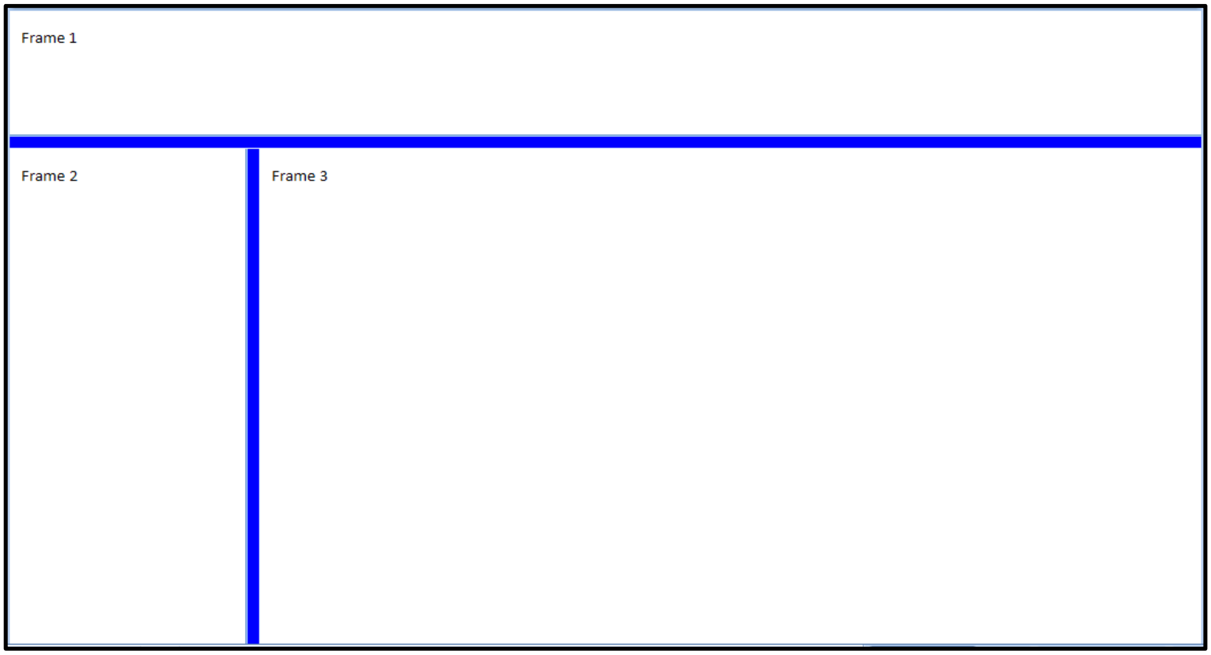
This element’s content model is defined by the common border properties definition in §17.3.4.

17.15.2.5 color (Frameset Splitter Color)

This element specifies the color of the splitters within the frameset in this WordprocessingML document. This element shall only be honored on the root frameset for this document, and can be ignored for all nested framesets in this document.

If this element is omitted, then the default color of the splitter can be automatically determined by the application displaying this WordprocessingML document (equivalent to a val attribute value of auto).

[*Example*: Consider a frameset consisting of the following three frames:



The following properties define the presentation of the splitter bars within this frameset:

```
<w:frameset>
  <w:framesetSplitbar>
    <w:w w:val="200" />
    <w:color w:val="0000FF" />
  </w:framesetSplitbar>
  …
</w:frameset>
```

The color element's val attribute specifies that the splitters must be displayed in the RGB color 0000FF (blue) when the contents of this document are displayed. *end example*]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>themeColor (Run Content Theme Color)</td><td>Specifies a theme color which should be applied to the current run.

The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part, which allows for color information to be set centrally in the document.

If the themeColor attribute is specified, then the val attribute is ignored for this run.

[<i>Example</i>: Consider a run of text which should be displayed using the accent3 theme color from the document’s Theme part. This requirement would be specified as follows in the resulting WordprocessingML:

<pre>&lt;w:rPr>
  <w:color w:themeColor="accent3" />
</w:rPr&gt;</pre>
The color attribute specifies that the run must use the accent3 theme color. <i>end example</i>]

The possible values for this attribute are defined by the ST_ThemeColor simple type (§17.18.97).</td></tr>
<tr><td>themeShade (Run Content Theme Color Shade)</td><td>Specifies the shade value applied to the supplied theme color (if any) for this run’s contents.

If the themeShade is supplied, then it is applied to the RGB value of the theme color to determine the final color applied to this run.

The themeShade value is stored as a hex encoding of the shade value (from 0 to 255) applied to the current border.

[<i>Example</i>: Consider a shade of 40% applied to a run in a document. This shade is calculated as follows:

$$\begin{aligned} S_{xml} &= 0.4 * 255 \\ &= 102 \\ &= 66(hex) \end{aligned}$$
Te resulting themeShade value in the file format would be 66. <i>end example</i>]

Given a input red, green, or blue color value C (from 0-255), an output color value of C' (from 0-255), and a shade value S (from 0-100), the shade is applied as follows:</td></tr>
</table>

Attributes	Description
	$$C' = \left(1 - \frac{S}{100}\right)\mathrm{C}$$ [*Example*: Consider a document with a run using the `accent6` theme color, whose RGB value (in RRGGBB hex format) is `F79646`. The hex value for the green component is 96 - 150 in decimal. Applying the shade formula with shade of 50%, the output decimal value of the green component is 75, or a hex value of 4B. This transformed value can be seen in the resulting run color WordprocessingML's val attribute: `<w:color w:val="7B4B23" w:themeColor="accent6" w:themeShade="80" />` *end example*] The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).
themeTint (Run Content Theme Color Tint)	Specifies the tint value applied to the supplied theme color (if any) for this run's contents. If the themeTint is supplied, then it is applied to the RGB value of the theme color to determine the final color applied to this run. The themeTint value is stored as a hex encoding of the tint value (from 0 to 255) applied to the current border. [*Example*: Consider a tint of 60% applied to a run in a document. This tint is calculated as follows: $$\begin{aligned} T_{xml} &= 0.6 * 255 \\ &= 153 \\ &= 99(hex) \end{aligned}$$ The resulting themeTint value in the file format would be 99. *end example*] Given a input red, green, or blue color value C (from 0-255), an output color value of C' (from 0-255), and a tint value T (from 0-100), the tint is applied as follows: $$C' = \left(1 - \frac{T}{100}\right)(255 - \mathrm{C}) + \mathrm{C}$$ [*Example*: Consider a document with a run using the accent1 theme color, whose RGB value (in RRGGBB hex format) is `C0504D`. The hex value for the green component is `50` - 80 in decimal. Applying the tint formula with tint of 60%, the output decimal value of the green component is 150, or a hex value

©ISO/IEC 2012 – All rights reserved

Attributes	Description

	of 96. This transformed value can be seen in the resulting run color's WordprocessingML val attribute:
	`<w:color w:val="D99694" w:themeColor="accent1" w:themeTint="99" />`
	end example]
	The possible values for this attribute are defined by the ST_UcharHexNumber simple type (§17.18.98).

val (Run Content Color)	Specifies the color for this run.
	This color can either be presented as a hex value (in RRGGBB format), or `auto` to allow a consumer to automatically determine the run color as appropriate.
	If the run specifies the use of a theme color via the themeColor attribute, then this value is superseded by the theme color value.
	[*Example*: Consider a run color with value `auto`, as follows:
	`<w:rPr>`
	`  <w:color … w:val="auto" />`
	`</w:rPr>`
	This color therefore can be automatically be modified by a consumer as appropriate, for example, in order to ensure that the run contents can be distinguished against the page's background color. *end example*]
	The possible values for this attribute are defined by the ST_HexColor simple type (§17.18.38).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Color) is located in §A.1. *end note*]

17.15.2.6 div (Information About Single HTML div Element)

This element specifies information about a single HTML `div`, `body`, or `blockquote` element which was included in this document, so that that information (which is stored on a logical structure with no direct analog in WordprocessingML) can be maintained when an HTML document is stored in the WordprocessingML format.

The div element stores the following information about these structures:

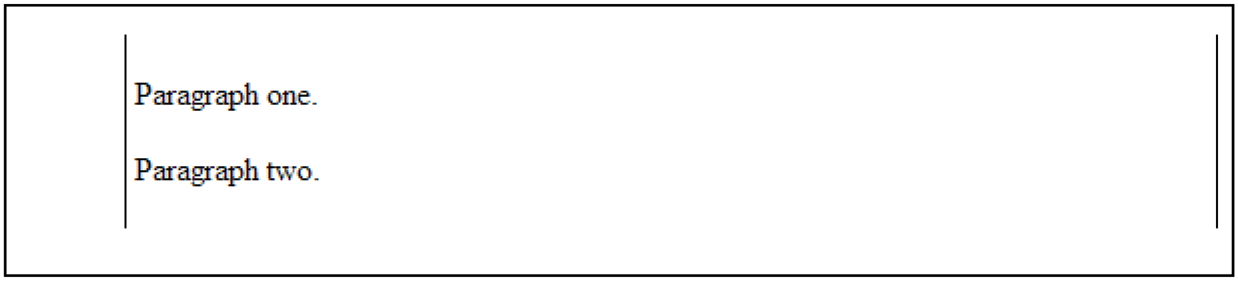
- The child HTML `div`, and `blockquote` elements
- The borders for the element
- The margins for the element

When the resulting WordprocessingML document is displayed by an application, the settings specified by this information shall be reflected in the formatting of the resulting paragraphs (i.e. this information shall not only be used when the document is resaved in the HTML format).

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style="border-left-style: solid; border-left-width: 1px; border-right-
style: solid; border-right-width: 1px; padding-left: 4px; padding-right: 4px;
padding-top: 1px; padding-bottom: 1px; margin-left: 50px">
      <p>Paragraph one.</p>
      <p>Paragraph two.</p>
    </div>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



Paragraph one.

Paragraph two.

Now, when this document is saved in the WordprocessingML format, the information stored on the `div`, `blockquote`, and body elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:marLeft w:val="750" />
    …
    <w:divBdr>
      <w:left w:val="single" w:sz="6" w:color="auto" />
      <w:right w:val="single" w:sz="6" w:color="auto" />
    </w:divBdr>
  </w:div>
</w:divs>
```

The div element specifies all margin and border information about the single HTML `div` structures in the document; in this case, the left indentation and the left and right borders. *end example*]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>id (div Data ID)</td><td>

Specifies a unique decimal number which shall be used to associate one or more structures in the WordprocessingML content with this HTML div information.

When a WordprocessingML structure (a paragraph or a table row) is associated with div information, it shall be associated with the set of information which most immediately contains the current object.

[*Example*: If a paragraph is wrapped within two HTML div elements, like this:

```
<div>
  <div>
    <p>Paragraph</p>
  </div>
</div>
```

The resulting WordprocessingML paragraph must reference the div Data ID associated with the inner HTML div element - the fact that it is also contained within the outer HTML div must be implied by the nesting of the corresponding WordprocessingML div elements in the web settings part. *end example*]

The ID specified by this attribute is then referenced by the divId element for all structures which are immediately contained within the specified HTML div.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body style=" margin-top:50px">
    <p>Paragraph one.</p>
    <div style="margin-left:50px">
      <p>Paragraph two.</p>
    </div>
  </body>
</html>
```

If the outer and inner body and div elements were assigned id attributes as follows:

```
<w:divs>
  <w:div w:id="1626542603">
    <w:bodyDiv w:val="1" />
    …
    <w:divsChild>
      <w:div w:id="313534916">
        …
      </w:div>
    </w:divsChild>
  </w:div>
</w:divs>
```

</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>

Then the first paragraph would reference the div ID of the outer div (since it is contained by the HTML body element) and the second paragraph would reference the div ID of the inner div (since it is contained within the child HTML `div` element), as follows:

```
<w:p>
  <w:pPr>
    <w:divId w:val="1626542603" />
  </w:pPr>
  <w:r>
    <w:t>Paragraph one.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:divId w:val="313534916" />
  </w:pPr>
  <w:r>
    <w:t>Paragraph one.</w:t>
  </w:r>
</w:p>
```

The id attributes on the div elements link each paragraph with the corresponding container div element. *end example*]

The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

</td></tr>
</table>

[*Note*: The W3C XML Schema definition of this element's content model (CT_Div) is located in §A.1. *end note*]

17.15.2.7 divBdr (Set of Borders for HTML div)

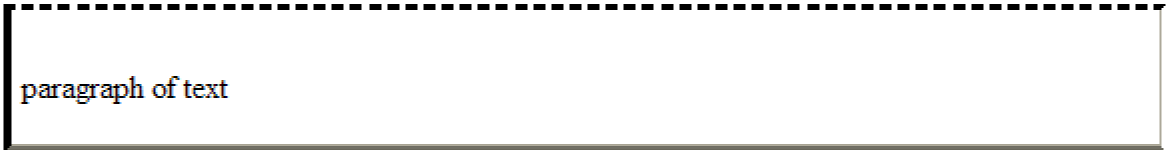
This element specifies the set of borders for the boundaries of the current HTML `div`, body, or `blockquote` element, using the four border types defined by its child elements.

If this element is omitted, then there shall be no borders associated with the current HTML v, body, or `blockquote` element.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" border-left-style:solid; border-right-style:groove; border-
right-width:3px; border-top-style:dashed; border-top-width:3px; border-bottom-
style:outset; border-bottom-width:3px">
      <p>paragraph of text</p>
      </div>
    </body>
  </html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



paragraph of text

Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
  <w:divs>
    <w:div w:id="1785730240">
      …
      <w:divBdr>
        <w:top w:val="dashed" w:sz="18" w:space="7" w:color="auto" />
        <w:left w:val="single" w:sz="24" w:space="4" w:color="auto" />
        <w:bottom w:val="outset" w:sz="18" w:color="auto" />
        <w:right w:val="threeDEngrave" w:sz="6" w:color="auto" />
      </w:divBdr>
    </w:div>
  </w:divs>
```

The divBdr element specifies border information about the single HTML `div` structure in the document. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_DivBdr) is located in §A.1. *end note*]

17.15.2.8 divs (Information about HTML div Elements)

This element specifies all information about the set of HTML `div` elements (as well as the body and `blockquote` elements) which were included in this document, so that that information (which is stored on a logical structure with no direct analog in WordprocessingML) can be maintained when an HTML document is stored in the WordprocessingML format.

The divs element stores the following information about these structures:

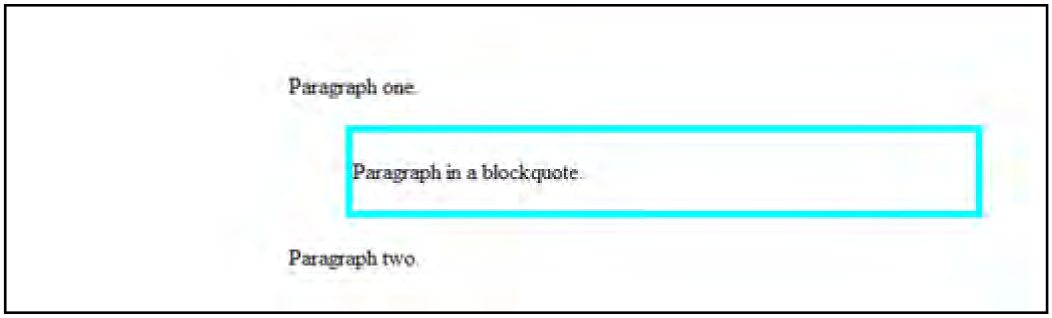
- The parent/child structure of HTML `div`, `blockquote`, and body elements
- The borders for each of these elements
- The margins for each of these elements

When the resulting WordprocessingML document is displayed by an application, the settings specified by this information shall be reflected in the formatting of the resulting paragraphs (i.e. this information shall not only be used when the document is resaved in the HTML format).

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body style="margin-left:200px;margin-top:50px">
    <p>Paragraph one.</p>
    <blockquote style="border: 5px solid #00FFFF">
      <p>Paragraph in a blockquote.</p>
    </blockquote>
    <p>Paragraph two.</p>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



Paragraph one.

Paragraph in a blockquote.

Paragraph two.

Now, when this document is saved in the WordprocessingML format, the information stored on the `div`, `blockquote`, and body elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1626542603">
    <w:bodyDiv w:val="1" />
    <w:marLeft w:val="3000" />
    <w:marTop w:val="750" />
    …    <w:divsChild>
        <w:div w:id="313534916">
          <w:blockQuote w:val="1" />
          <w:marLeft w:val="720" />
          <w:marRight w:val="720" />
          <w:marTop w:val="100" />
          <w:marBottom w:val="100" />
          <w:divBdr>
            <w:top w:val="single" w:sz="36" w:color="00FFFF" />
            <w:left w:val="single" w:sz="36" w:color="00FFFF" />
            <w:right w:val="single" w:sz="36" w:color="00FFFF" />
            <w:bottom w:val="single" w:sz="36" w:color="00FFFF" />
          </w:divBdr>
        </w:div>
      </w:divsChild>
    </w:div>
  </w:divs>
```

The divs element specifies all of the margin and border information about the necessary HTML structures in the document; in this case, the body element and the nested `blockquote`. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Divs) is located in §A.1. *end note*]

17.15.2.9 divsChild (Child div Elements Contained within Current div)

This element specifies the set of HTML `div` or `blockquote` elements which are contained within the current HTML `div`, body, or `blockquote` element, establishing the parent/child hierarchy of the original set of these elements.

When an HTML document containing these objects is saved in the WordprocessingML format, WordprocessingML objects store a reference to their most immediate parent `div`, body, or `blockquote` element using the divId element.

However, since only a single reference is stored, this information is often insufficient to determine the appropriate parent/child hierarchy for the original HTML `div` data, so it can be applied appropriately. This element allows that hierarchy to be stored, as child HTML `div` elements are stored within the childDivs element.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" margin-top:50px">
      <p>Paragraph one.</p>
      <div style="margin-left:50px">
        <p>Paragraph two.</p>
      </div>
    </div>
  </body>
</html>
```

If the outer and inner body and div elements were assigned id attributes of 1626542603 and 313534916 respectively, then the first paragraph would reference the div ID of the outer div (since it is contained within that HTML div element) and the second paragraph would reference the div ID of the inner div (since it is contained within the child HTML div element), as follows:

```
<w:p>
  <w:pPr>
    <w:divId w:val="1626542603" />
  </w:pPr>
  <w:r>
    <w:t>Paragraph one.</w:t>
  </w:r>
</w:p>
<w:p>
  <w:pPr>
    <w:divId w:val="313534916" />
  </w:pPr>
  <w:r>
    <w:t>Paragraph one.</w:t>
  </w:r>
</w:p>
```

However, this information alone is insufficient - it is unclear if the second div is contained within, or simply adjacent to, the first one.

In order to preserve this information, the correct hierarchy is stored within the web settings part:

```
<w:divs>
  <w:div w:id="1626542603">
    …
    <w:divsChild>
      <w:div w:id="313534916">
        …
      </w:div>
    </w:divsChild>
  </w:div>
</w:divs>
```

The divsChild element contains the second div as a child of the first div, specifying that the first div covers both paragraphs. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Divs) is located in §A.1. *end note*]

17.15.2.10 doNotOrganizeInFolder (Do Not Place Supporting Files in Subdirectory)

This element specifies that applications shall not automatically place all supporting files (images which are part of this HTML web page, etc.) in a subdirectory when the contents of this WordprocessingML document are saved as a web page. Typically, applications which save a document as a web page consisting of multiple files save all supporting files in a subdirectory next to the main HTML file (in order to keep those files organized). This element specifies the files shall be placed in the same directory as the actual web page.

If this element is omitted from the document, then all supporting files should be saved into a subdirectory beneath the main web page file when this document is saved as a web page.

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:doNotOrganizeInFolder w:val="true" />
</w:webSettings>
```

The doNotOrganizeInFolder element has a val attribute value of true, which specifies that applications should save all supplementary files in the same directory as the main web page HTML document when saving this WordprocessingML document as a web page. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.2.11 doNotRelyOnCSS (Do Not Rely on CSS for Font Face Formatting)

This element specifies whether applications can rely on the CSS properties for font face (the font-family property) when saving this WordprocessingML document as a web page. If this element is utilized, then the HTML font element should be used either in place of or in concert with these CSS properties in order to specify the font face formatting for the resulting web page.

If this element is omitted, then applications can choose to rely on the CSS properties for font face as desired.

[*Note*: This setting is intended for applications to save web pages which can be supported by legacy web browsers which do not support the reading of these CSS properties when attempting to read and display the resulting web page, in order to maximize the fidelity of the resulting output. *end note*]

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:doNotRelyOnCSS w:val="true" />
</w:webSettings>
```

The doNotRelyOnCSS element has a val attribute value of true, which specifies that applications should include the HTML font element when saving this WordprocessingML document as a web page. For example, this output:

```
<span style='font-family:"Courier New"'>text</span>
```

This output would instead be saved as follows:

```
<font face="Courier New"><span style='font-family:"Courier
New"'>text</span></font>
```

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.2.12 doNotSaveAsSingleFile (Recommend Web Page Format over Single File Web Page Format)

This element specifies that applications should recommend that new web page files generated using this WordprocessingML document use a multi-file web page format (HTML), rather than a single-file web page format (MHTML) when this document is saved as an HTML web page. This setting shall not prevent the use of the MHTML format; it shall only cause applications to recommend (via a default) a non single-file format when saving as a web page.

[*Note*: This setting is primarily intended for applications which explicitly support a "Save as Web Page..." action, in order to determine the default setting for the resulting web page. *end note*]

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:doNotSaveAsSingleFile w:val="true" />
</w:webSettings>
```

The doNotSaveAsSingleFile element specifies that applications should recommend a multi-file web page format when this document is subsequently saved as a web page. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.2.13 doNotUseLongFileNames (Do Not Use File Names Longer than 8.3 Characters)

This element specifies that applications shall ensure that the file names for all files generated when saving this document as a web page do not exceed eight octets with a three octet extension. This includes all supporting files (images which are part of this HTML web page, etc.). The file names generated are not case-sensitive.

[*Note*: This setting is intended for applications to save web pages which can be supported by legacy web browsers which do not support the reading of long file names when attempting to read and display the resulting web page. *end note*]

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:doNotUseLongFileNames w:val="true" />
</w:webSettings>
```

The doNotUseLongFileNames element specifies that applications should ensure that all file names generated when this document is subsequently saved as a web page do not exceed the 8.3 octet file name limitation. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.2.14 encoding (Output Encoding When Saving as Web Page)

This element specifies the encoding which shall be used for the contents of this WordprocessingML document when it is saved as an HTML web page. The set of encodings supported by this element shall be derived from the standard set of character set definitions provided at http://www.iana.org/assignments/character-sets.

If this element is omitted, then the default encoding for the current system shall be used when this document is saved as a web page. If the value of the val attribute is unknown or supported by an application, then the default encoding for the current system shall be used when this document is saved as a web page.

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:encoding w:val="utf-8" />
</w:webSettings>
```

The encoding element's val attribute has a value of `utf-8`, which specifies that this document must be encoded in the UTF-8 format when it is saved as a web page. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

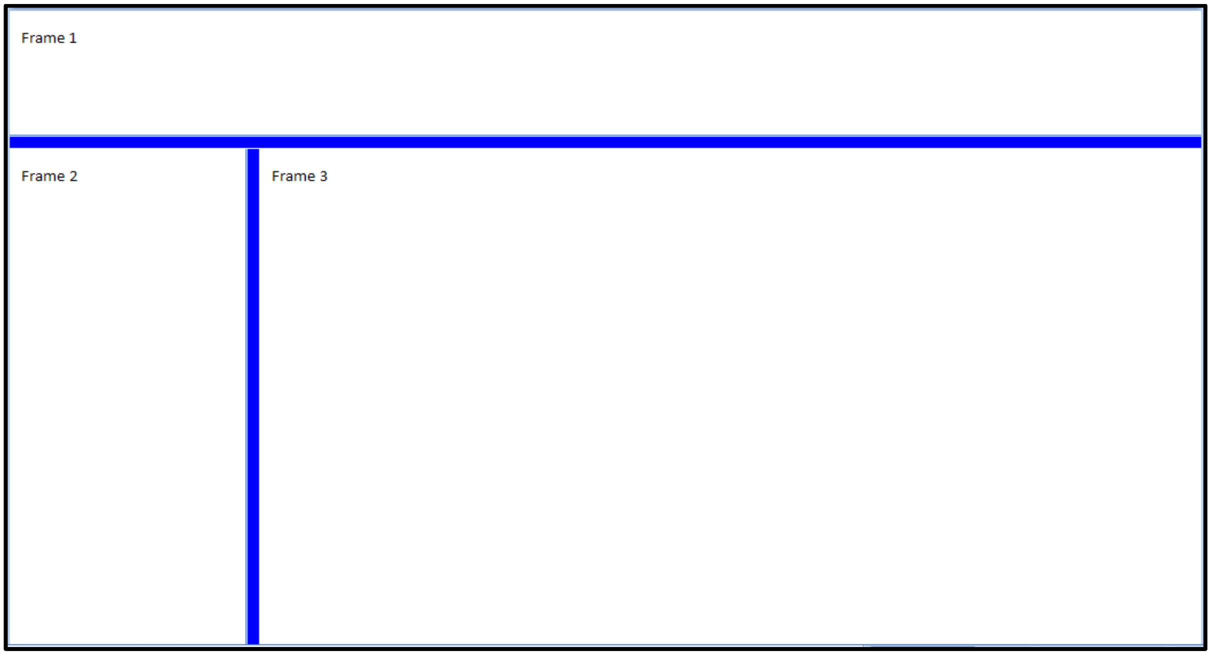
[*Note*: The W3C XML Schema definition of this element’s content model (CT_String) is located in §A.1. *end note*]

17.15.2.15 flatBorders (Frameset Splitter Border Style)

This element specifies the 3D style of the splitters within the frameset in this WordprocessingML document. This element shall only be honored on the root frameset for this document, and can be ignored for all nested framesets in this document. When this property is turned on, the borders for this frameset shall be flat (not 3D), otherwise they can be presented as 3D splitter when they are displayed.

If this element is omitted, then the default style of the splitter should be a 3D splitter.

[*Example*: Consider a frameset consisting of the following three frames:



The following properties define the presentation of the splitter bars within this frameset:

```
<w:frameset>
  <w:framesetSplitbar>
    <w:w w:val="200" />
    <w:color w:val="0000FF" />
    <w:flatBorders w:val="true" />
  </w:framesetSplitbar>
  …
</w:frameset>
```

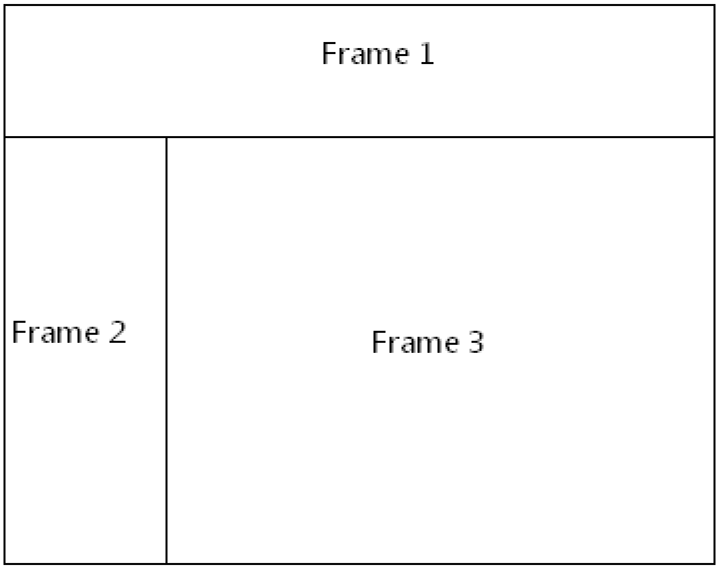
The flatBorders element's val attribute has a value of true, which specifies that the style of the splitters must be flat (the splitter might not be 3D when displayed). *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.2.16 frame (Single Frame Properties)

This element specifies the properties for a single frame within a frameset document. When a document defines a frameset using the frameset element; that frameset is composed of a set of frames, each of which is specified by a single frame element.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frame>
    <w:sz w:val="20%" />
    <w:name w:val="Frame 1" />
    <w:sourceFileName r:id="rId1" />
  </w:frame>
  <w:frameset>
    …
  </w:frameset>
</w:frameset>
```

The frame element specifies the set of properties for a single frame in the document. In this case, these properties (for the frame marked with `Frame 1` in the diagram above) specify that the frame must have the following properties:

- A height of `20%` of the height of the document
- A name of `Frame 1`
- The contents of the frame must be pulled from the document that is the target of the relationship with ID `rId1`

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Frame) is located in §A.1. *end note*]

17.15.2.17 frameLayout (Frameset Layout)

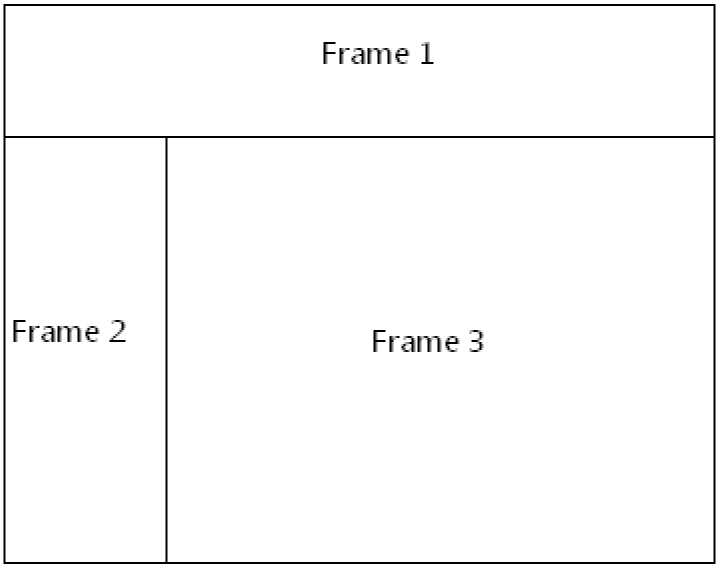
This element specifies the order in which the frames (and nested framesets) in a frameset shall be displayed. When a frameset is created, it can only contain frames which are stacked in one direction:

- Vertically (one on top of another)
- Horizontally (one next to another)

This element specifies how the frames in this frameset are stacked, which shall also be used to interpret the sizes defined by the sz element (§17.15.2.39) for each frame. In order to determine the ordering of the constituent frames within this frameset, the ordering of the child frame and frameset elements shall be used.

If this element is omitted, then the frames in this frameset shall be stacked vertically on top of one another (a row frameset).

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frameLayout w:val="rows" />
  <w:frame>
    …
  </w:frame>
  <w:frameset>
    <w:frameLayout w:val="cols" />
    <w:frame>
```

```
        …
      </w:frame>
      <w:frame>
        …
      </w:frame>
    </w:frameset>
  </w:frameset>
```

The frameLayout element specifies that the outer frameset is a consists of the single frame and the child frameset stacked vertically, and an inner nested frameset consisting of two frames stacked horizontally. *end example*]

Attributes	Description

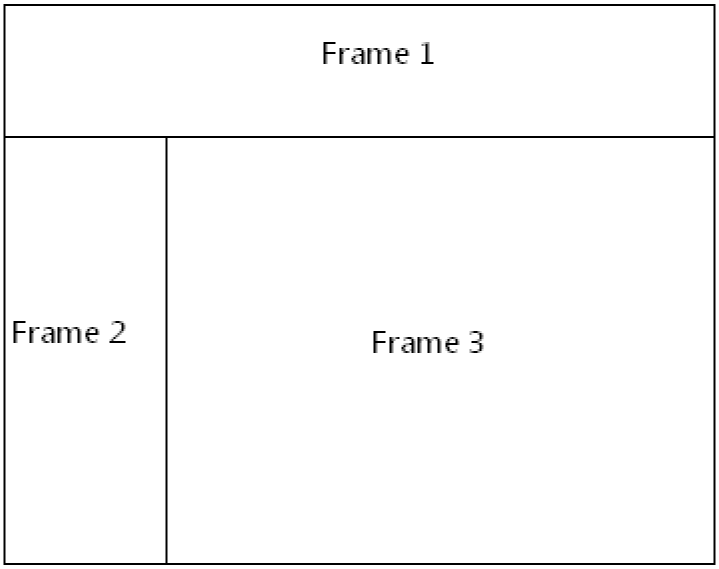
val (Frameset Layout Value)	Specifies the type of layout which shall be used to display the contents of the frames and nested framesets within this frameset, as defined by the simple type referenced below.
	[*Example*: Consider a frameset definition within a WordprocessingML document which defines the following frameset layout setting:
	`<w:frameset>`
	`  <w:frameLayout w:val="cols" />`
	`  …`
	`</w:frameset>`
	The val attribute value of `cols` specifies that the contents of this frameset must be stacked horizontally (in columns). *end example*]
	The possible values for this attribute are defined by the ST_FrameLayout simple type (§17.18.31).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FrameLayout) is located in §A.1. *end note*]

17.15.2.18 frameset (Nested Frameset Definition)

This element specifies a frameset which has been nested within another frameset within a WordprocessingML document. This WordprocessingML element is analogous to the `frameset` element in HTML (when that frameset is the child of another `frameset` element).

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  <w:frameLayout w:val="rows" />
  <w:frame>
    …
  </w:frame>
  <w:frameset>
    <w:frameLayout w:val="cols" />
    <w:frame>
      …
    </w:frame>
    <w:frame>
      …
    </w:frame>
  </w:frameset>
</w:frameset>
```

The child frameset element specifies the frameset definition for the inner frameset; that frameset consists of two frames (`Frame 2` and `Frame 3` in the image above) which have been laid out horizontally as columns. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Frameset) is located in §A.1. *end note*]

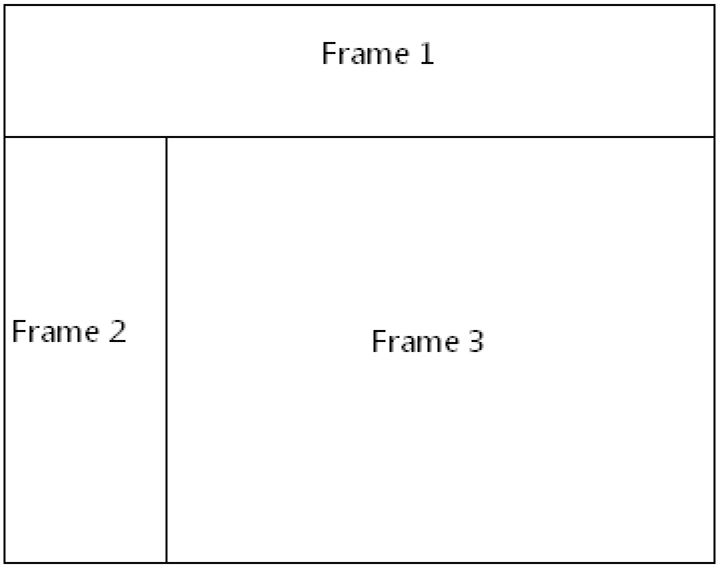
17.15.2.19 frameset (Root Frameset Definition)

This element specifies that this document is the container for a frameset. This WordprocessingML element is analogous to the `frameset` element in HTML.

When the frameset element is present within a document, that document shall serve as a frameset definition only; all of its normal document content shall therefore not be displayed as long as it contains at least one child frame or frameset element.

If this element is omitted, then the currently document shall not be treated as a frameset definition; its regular document content shall be displayed.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  <w:frameLayout w:val="rows" />
  <w:frame>
    …
  </w:frame>
  <w:frameset>
    <w:frameLayout w:val="cols" />
    <w:frame>
      …
    </w:frame>
    <w:frame>
      …
      </w:frame>
    </w:frameset>
  </w:frameset>
```

The parent frameset element specifies that the current document is a frameset definition; that frameset consists of a single frame and another nested frameset stacked vertically. *end example*]

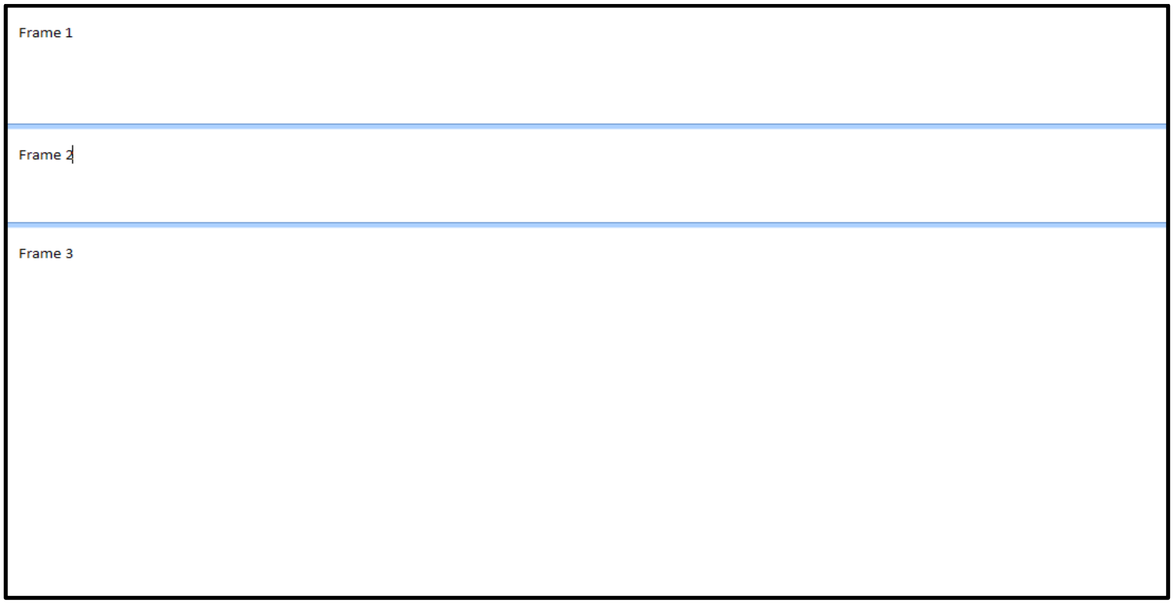
[*Note*: The W3C XML Schema definition of this element's content model (CT_Frameset) is located in §A.1. *end note*]

17.15.2.20 framesetSplitbar (Frameset Splitter Properties)

This element specifies the properties for the splitters associated with this frameset. A *splitter* is a horizontal or vertical line which visually separates the contents of one frame from another within a frameset.

If this element is omitted, then the default parameters for each of the child frameset properties shall be used for all splitters in this frameset.

[*Example*: Consider a frameset consisting of the following three frames:



The following properties define the presentation of the splitter bars within this frameset:

```
  <w:frameset>
    <w:framesetSplitbar>
      <w:w w:val="90" />
      <w:color w:val="auto" />
    </w:framesetSplitbar>
    …
  </w:frameset>
```

The framesetSplitbar element specifies the properties for all splitters in this frameset; in this case, those properties are that the splitter must be 4.5 points (90 twentieths of a point) wide, and that the color of the splitter must be automatically determined via the attribute value of auto. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_FramesetSplitbar) is located in §A.1. *end note*]

17.15.2.21 left (Left Border for HTML div)

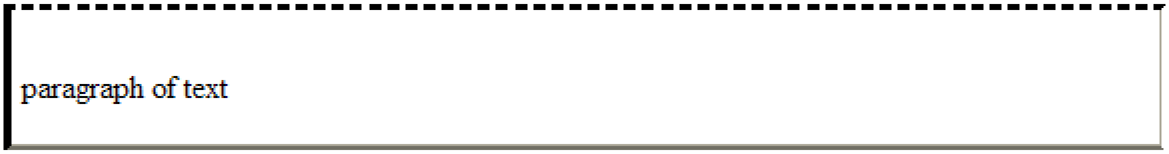
This element specifies the border which shall be displayed at the left of the boundaries of the current HTML div object.

If this element is omitted, then this HTML div object shall not have a left border.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" border-left-style:solid; border-right-style:groove; border-
right-width:1px; border-top-style:dashed; border-top-width:3px; border-bottom-
style:outset; border-bottom-width:3px">
      <p>paragraph of text</p>
    </div>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



paragraph of text

Now, when this document is saved in the WordprocessingML format, the information stored on the div elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    …
    <w:divBdr>
      <w:top w:val="dashed" w:sz="18" w:space="7" w:color="auto" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="auto" />
      <w:bottom w:val="outset" w:sz="18" w:color="auto" />
      <w:right w:val="threeDEngrave" w:sz="6" w:color="auto" />
    </w:divBdr>
  </w:div>
</w:divs>
```

The left element specifies border information about the left border for the single HTML `div` structure in the document; in this case, a 3 point bottom border of type `single`. The initial 4 pixel border was converted to 3 points using the following logic:

$$4\text{px} * \frac{1\text{ inch}}{96\text{ px}} * \frac{576\text{ eighth points}}{1\text{ inch}} = 24\text{ eighth points (3 points)}$$

end example]

This element's content model is defined by the common border properties definition in §17.3.4.

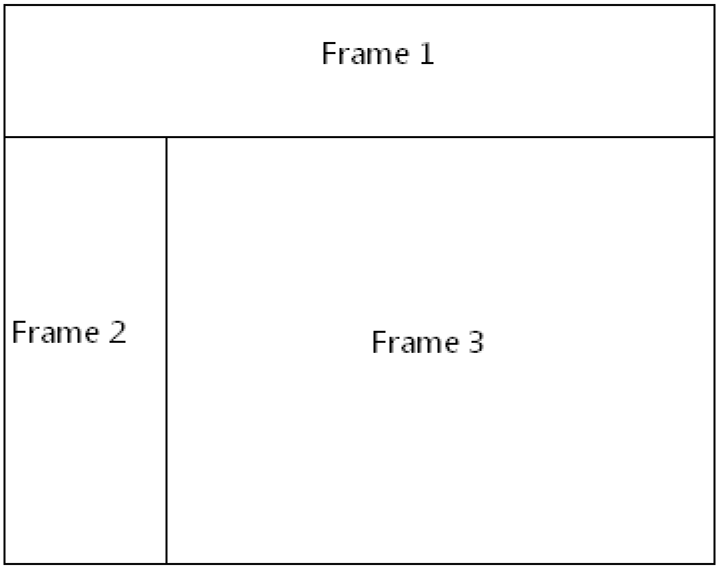
17.15.2.22 linkedToFile (Maintain Link to Existing File)

This element specifies that the file referenced by the sourceFileName element (§17.15.2.38) as the basis for the current frame shall not be changed, even when the file defined by the parent frameset is moved - i.e. the link shall remain exactly as specified.

[*Guidance*: Typically, when a document is incorporated into a frameset, a copy of that document is made such that all files encompassing the frameset are stored in a single subdirectory (so they can be moved as a single unit). However, if the link to the current file is absolute and must not be changed even when the location of the main frameset document, then this element must be set to indicate that setting. *end guidance*]

If this element is omitted, then a new file can be created as necessary when the parent frameset document is resaved to another location.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frame>
    <w:sz w:val="20%" />
    <w:name w:val="Frame 1" />
    <w:sourceFileName r:id="rId1" />
    <w:linkedToFile w:val="true" />
  </w:frame>
  <w:frameset>
    …
  </w:frameset>
</w:frameset>
```

The linkedToFile element specifies that the frame source location specified by the sourceFileName element (§17.15.2.38) must not be modified, even if the parent frameset document is resaved to another location. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

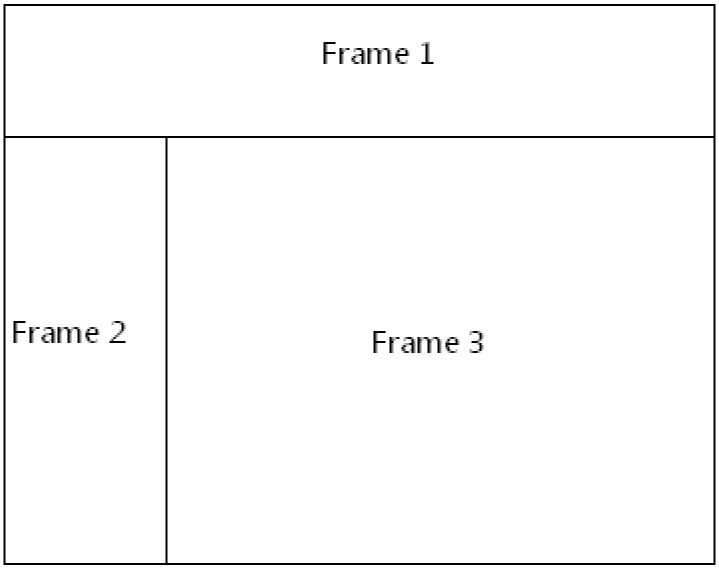
17.15.2.23 longDesc (Frame Long Description)

This element specifies an explicit relationship whose target is the long description of the frame. This description should supplement the short description provided by the title element. This property is analogous to the longdesc attribute on the frame element in HTML.

If this element is omitted, then no long description shall be associated with the given frame.

©ISO/IEC 2012 – All rights reserved

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frame>
    <w:name w:val="Frame 1" />
    <w:title w:val="Menu bar" />
    <w:longDesc r:id="rIdMenuBar" />
  </w:frame>
  <w:frameset>
    <w:title w:val="Navigation and document collection" />
    …
    <w:frame>
      <w:name w:val="Frame 2" />
      <w:title w:val="Navigation bar" />
      <w:longDesc r:id="rIdNavBar" />
    </w:frame>
    <w:frame>
      <w:name w:val="Frame 3" />
      <w:title w:val="Documents" />
      <w:longDesc r:id="rIdDocs" />
    </w:frame>
  </w:frameset>
</w:frameset>
```

The longDesc element specifies that the part targeted by the relationship with an id of `rIdMenuBar` must be used for supplementary information for Frame 1. Examining the contents of the corresponding relationship part item, we can see the targets for that relationship:

```
<Relationships … >
  …
  <Relationship Id="rIdMenuBar" TargetMode="External"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink"
Target="myexample-desc.html#menubar" />
  …
</Relationships>
```

The corresponding relationship part item shows that the long description supplementary information for Frame 1 is located at `myexample-desc.html#menubar`, where myexample-desc.html describes #menubar as "This frame provides links to the major sections of the site: Home, Resources, Links, Help." *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rel) is located in §A.1. *end note*]

17.15.2.24 marBottom (Bottom Margin for HTML div)

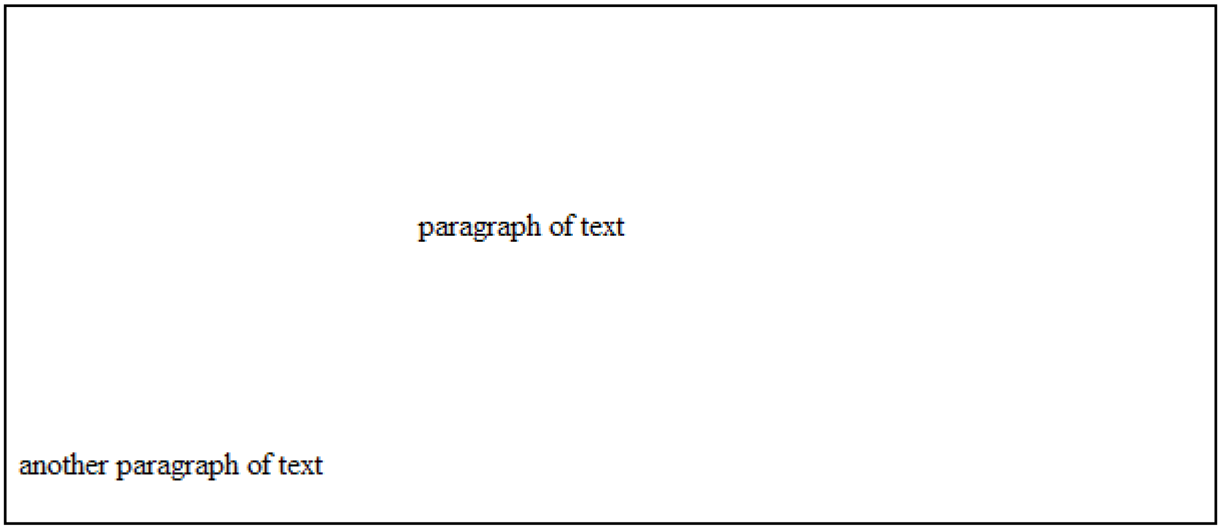
This element specifies the margin which shall be displayed at the bottom of the boundaries of the current HTML `div` object.

If this element is omitted, then this HTML `div` object shall not have a bottom margin.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style="margin-top:100px; margin-left:200px; margin-right:50px; margin-
bottom:100px">
      <p>paragraph of text</p>
    </div>
    <p>another paragraph of text</p>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



paragraph of text

another paragraph of text

Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:marLeft w:val="3000" />
    <w:marRight w:val="750" />
    <w:marTop w:val="1500" />
    <w:marBottom w:val="1500" />
  </w:div>
</w:divs>
```

The marBottom element specifies margin information about the bottom margin for the single HTML `div` structure in the document; in this case, a 75 point bottom margin. The initial 100 pixel margin was converted to 75 points using the following logic:

$$100\text{px} * \frac{1\text{ inch}}{96\text{ px}} * \frac{1440\text{ twentieth points}}{1\text{ inch}} = 1500\text{ twentieth points (75 points)}$$

end example]

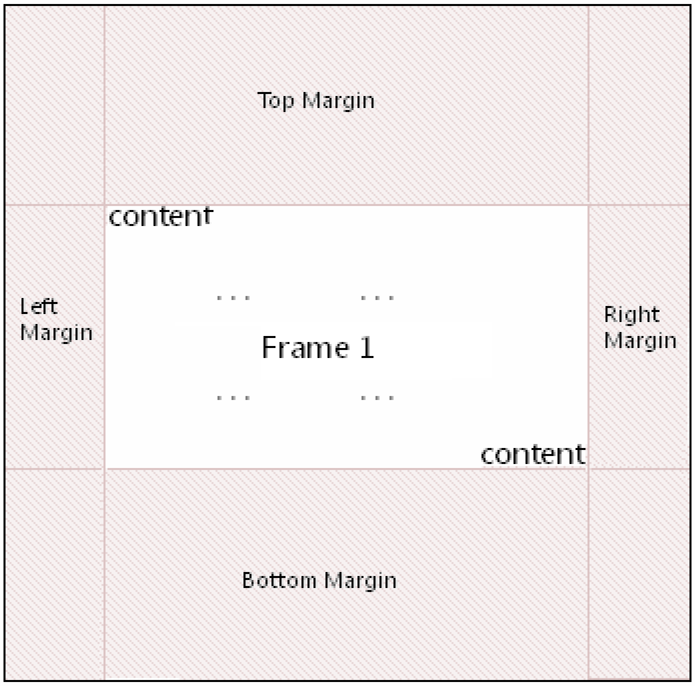
Attributes	Description
val (Positive or Negative Value in Twentieths of a Point)	Specifies a positive or negative measurement in twentieths of a point (equivalent to 1/1440th of an inch). The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of `-720` whose type is ST_SignedTwipsMeasure. This attribute value specifies a value of negative one-half of an inch or -36 points (-`720` twentieths of a point = -36 points = -0.5 inches). *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SignedTwipsMeasure) is located in §A.1. *end note*]

17.15.2.25 marH (Top and Bottom Margin for Frame)

This element specifies the top and bottom margin height for a single frame in a frameset document, as follows:

©ISO/IEC 2012 – All rights reserved



This height is expressed in pixels.

If this element is omitted, then no top or bottom margin shall be used for this frame.

[*Example*: Consider a document that has a frame, where the margin height has been specified and is represented as the following WordprocessingML:

```
<w:frame>
  <w:marH w:val="594"/>
</w:frame>
```

The marH element has a val attribute value of 594, which specifies that this frame has a top and bottom margin value of 594 pixels, resulting in 594 pixels of space between the content and the top and bottom margins of the frame. *end example*]

Attributes	Description
val (Measurement in Pixels)	Specifies a value whose contents shall contain a positive whole number, whose contents consist of a positive measurement in pixels. The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of 960 whose simple type is ST_PixelsMeasure. This attribute value specifies a value of 960 pixels. *end example*] The possible values for this attribute are defined by the ST_PixelsMeasure simple type (§17.18.67).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PixelsMeasure) is located in §A.1. *end note*]

17.15.2.26 marLeft (Left Margin for HTML div)

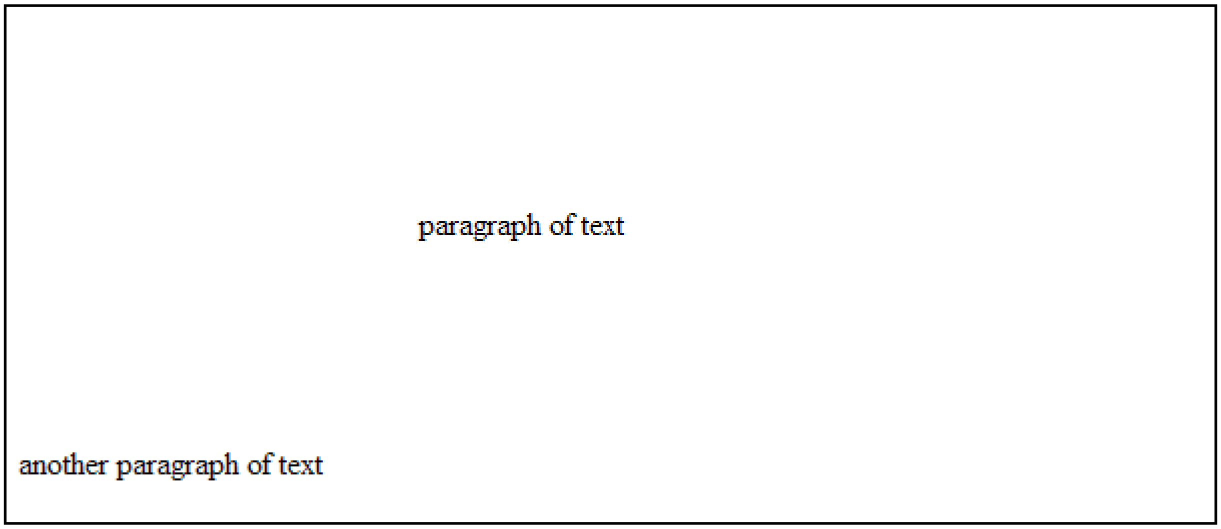
This element specifies the margin which shall be displayed at the left of the boundaries of the current HTML `div` object.

If this element is omitted, then this HTML `div` object shall not have a left margin.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style="margin-top:100px; margin-left:200px; margin-right:50px; margin-
bottom:100px">
      <p>paragraph of text</p>
    </div>
    <p>another paragraph of text</p>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



paragraph of text

another paragraph of text

Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:marLeft w:val="3000" />
    <w:marRight w:val="750" />
    <w:marTop w:val="1500" />
    <w:marBottom w:val="1500" />
  </w:div>
</w:divs>
```

The marLeft element specifies margin information about the left margin for the single HTML div structure in the document; in this case, a 150 point left margin. The initial 200 pixel margin was converted to 150 points using the following logic:

$$200\text{px} * \frac{1 \text{ inch}}{96 \text{ px}} * \frac{1440 \text{ twentieth points}}{1 \text{ inch}} = 3000 \text{ twentieth points (150 points)}$$

end example]

Attributes	Description
val (Positive or Negative Value in Twentieths of a Point)	Specifies a positive or negative measurement in twentieths of a point (equivalent to 1/1440th of an inch). The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of -720 whose type is ST_SignedTwipsMeasure. This attribute value specifies a value of negative one-half of an inch or -36 points (-720 twentieths of a point = -36 points = -0.5 inches). *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SignedTwipsMeasure) is located in §A.1. *end note*]

17.15.2.27 marRight (Right Margin for HTML div)

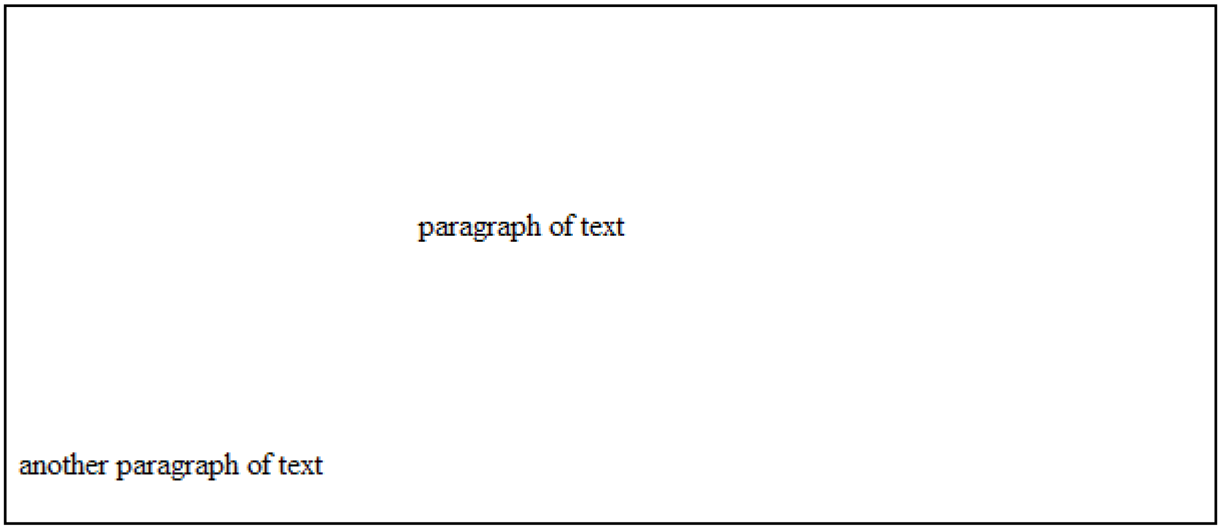
This element specifies the margin which shall be displayed at the right of the boundaries of the current HTML div object.

If this element is omitted, then this HTML div object shall not have a right margin.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style="margin-top:100px; margin-left:200px; margin-right:50px; margin-
bottom:100px">
      <p>paragraph of text</p>
    </div>
    <p>another paragraph of text</p>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



paragraph of text

another paragraph of text

Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:marLeft w:val="3000" />
    <w:marRight w:val="750" />
    <w:marTop w:val="1500" />
    <w:marBottom w:val="1500" />
  </w:div>
</w:divs>
```

The marRight element specifies margin information about the right margin for the single HTML `div` structure in the document; in this case, a 37.5 point right margin. The initial 50 pixel margin was converted to 37.5 points using the following logic:

$$50\text{px} * \frac{1\text{ inch}}{96\text{ px}} * \frac{1440\text{ twentieth points}}{1\text{ inch}} = 750\text{ twentieth points (37.5 points)}$$

end example]

Attributes	Description
val (Positive or Negative Value in Twentieths of a Point)	Specifies a positive or negative measurement in twentieths of a point (equivalent to 1/1440th of an inch). The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of `-720` whose type is ST_SignedTwipsMeasure. This attribute value specifies a value of negative one-half of an inch or -36 points (-`720` twentieths of a point = -36 points = -0.5 inches). *end example*] The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SignedTwipsMeasure) is located in §A.1. *end note*]

17.15.2.28 marTop (Top Margin for HTML div)

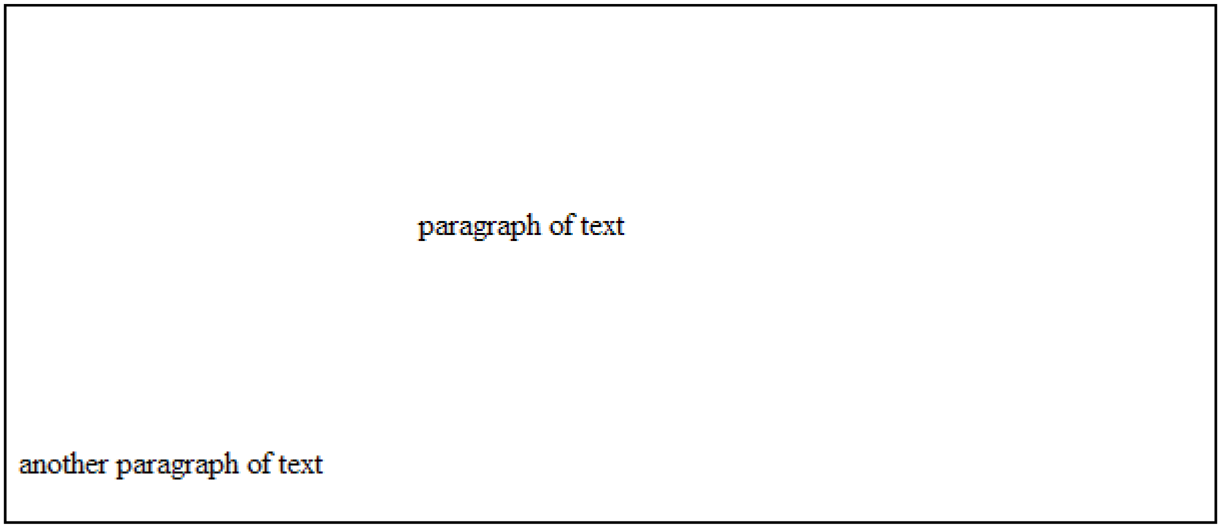
This element specifies the margin which shall be displayed at the top of the boundaries of the current HTML `div` object.

If this element is omitted, then this HTML `div` object shall not have a top margin.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style="margin-top:100px; margin-left:200px; margin-right:50px; margin-
bottom:100px">
      <p>paragraph of text</p>
    </div>
    <p>another paragraph of text</p>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



paragraph of text

another paragraph of text

Now, when this document is saved in the WordprocessingML format, the information stored on the `div` elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    <w:marLeft w:val="3000" />
    <w:marRight w:val="750" />
    <w:marTop w:val="1500" />
    <w:marBottom w:val="1500" />
  </w:div>
</w:divs>
```

The marTop element specifies margin information about the top margin for the single HTML `div` structure in the document; in this case, a 75 point top margin. The initial 100 pixel margin was converted to 75 points using the following logic:

$$100\text{px} * \frac{1\text{ inch}}{96\text{ px}} * \frac{1440\text{ twentieth points}}{1\text{ inch}} = 1500\text{ twentieth points (75 points)}$$

end example]

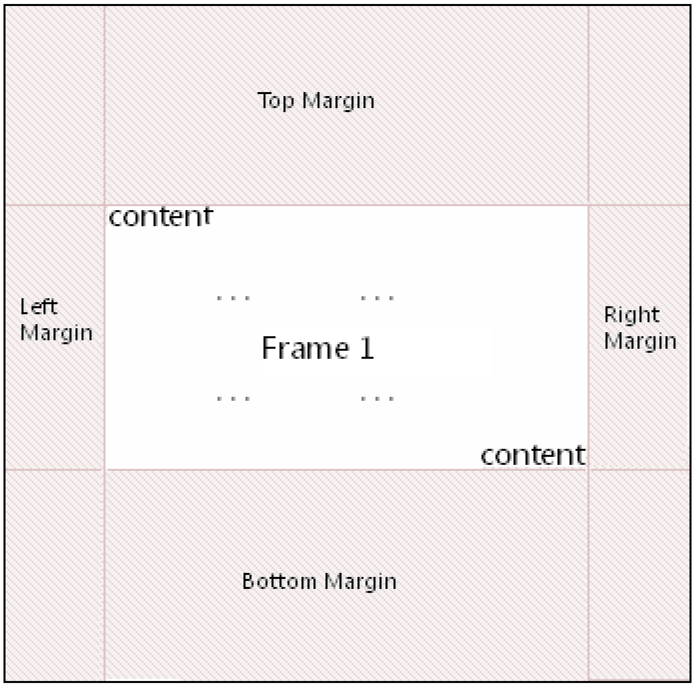
Attributes	Description
val (Positive or Negative Value in Twentieths of a Point)	Specifies a positive or negative measurement in twentieths of a point (equivalent to 1/1440th of an inch). The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of `-720` whose type is ST_SignedTwipsMeasure. This attribute value specifies a value of negative one-half of an inch or -36 points (-`720` twentieths of a point = -36 points = -0.5 inches). *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_SignedTwipsMeasure simple type (§17.18.81).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SignedTwipsMeasure) is located in §A.1. *end note*]

17.15.2.29 marW (Left and Right Margin for Frame)

This element specifies the left and right margin height for a single frame in a frameset document, as follows:



This height is expressed in pixels.

If this element is omitted, then no left or right margin shall be used for this frame.

[*Example*: Consider a document that has a frame, where the frame's margins have been specified and is represented as the following WordprocessingML:

```
<w:frame>
  <w:marW w:val="294"/>
</w:frame>
```

The marW element has a val attribute value of 294, which specifies that this frame has a left and right margin value of 294 pixels, resulting in 294 pixels of space between the content and the left and right margins of the frame. *end example*]

Attributes	Description
val (Measurement in Pixels)	Specifies a value whose contents shall contain a positive whole number, whose contents consist of a positive measurement in pixels. The contents of this measurement shall be interpreted based on the context of the parent XML element. [*Example*: Consider an attribute value of `960` whose simple type is ST_PixelsMeasure. This attribute value specifies a value of 960 pixels. *end example*] The possible values for this attribute are defined by the ST_PixelsMeasure simple type (§17.18.67).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PixelsMeasure) is located in §A.1. *end note*]

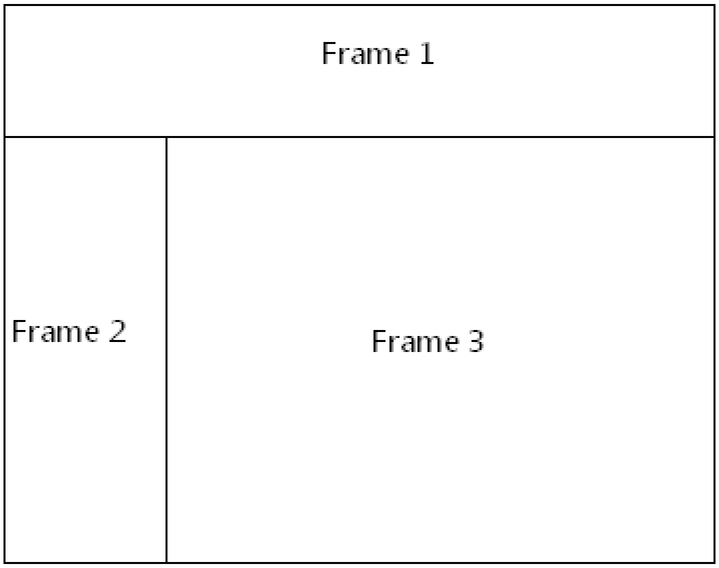
17.15.2.30 name (Frame Name)

This element specifies the name of a single frame within a frameset document. This property is analogous to the `name` attribute on the `frame` element in HTML.

[*Note*: The name of a frame can be used in web pages that reference a frame via targeted links, etc. *end note*]

If this element is omitted, then the current frame shall have no name associated with it.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frame>
    <w:name w:val="Frame 1" />
  </w:frame>
  <w:frameset>
    …
    <w:frame>
      <w:name w:val="Frame 2" />
    </w:frame>
    <w:frame>
      <w:name w:val="Frame 3" />
    </w:frame>
  </w:frameset>
</w:frameset>
```

The name element specifies the name for each frame within this frameset; in this case, the frames have names of `Frame 1`, `Frame 2`, and `Frame 3` respectively. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

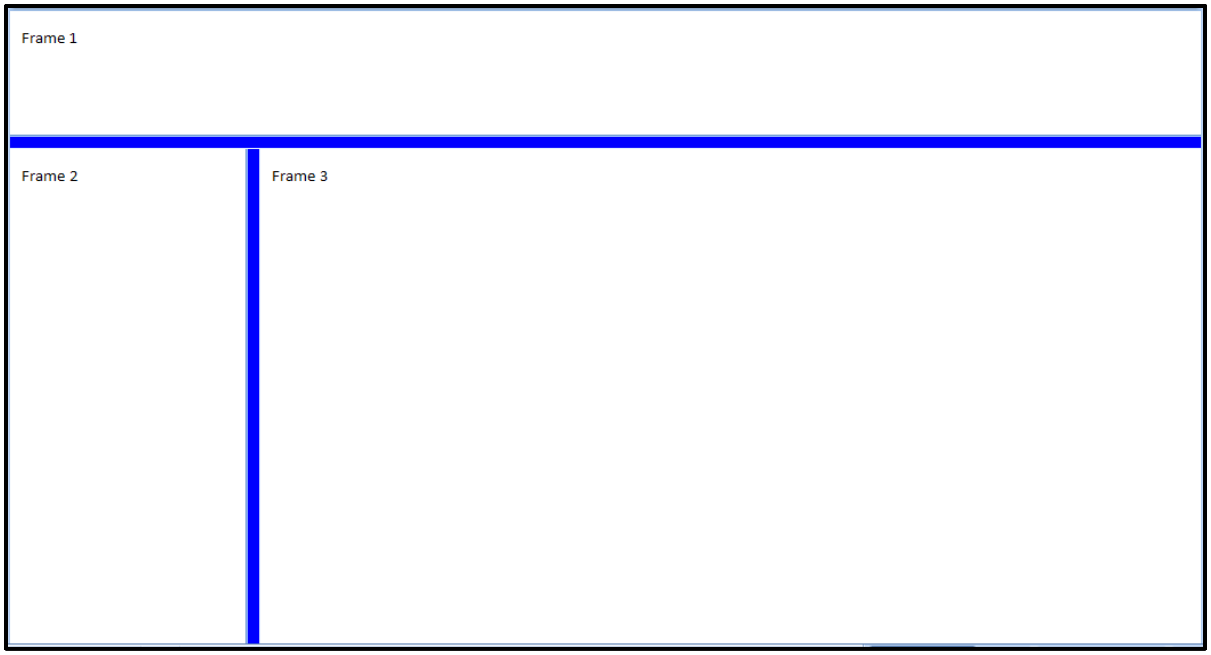
[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.15.2.31 noBorder (Do Not Display Frameset Splitters)

This element specifies whether the splitters shall be displayed for the contents of the frameset in this WordprocessingML document. This element shall only be honored on the root frameset for this document, and can be ignored for all nested framesets in this document. If this element is present, then no splitters shall be displayed, and all other frameset splitter properties can be ignored.

If this element is omitted, then the splitters in this document shall be displayed as defined by the w and color elements.

[*Example*: Consider a frameset consisting of the following three frames:



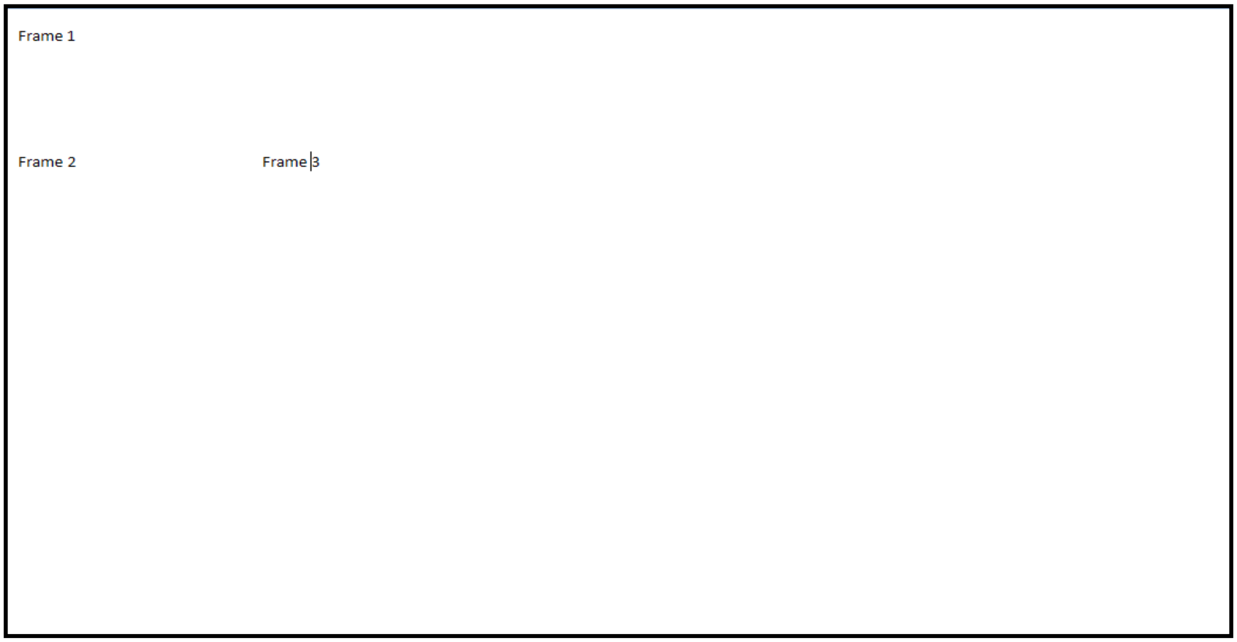
The following properties define the presentation of the splitter bars within this frameset:

```
<w:frameset>
  <w:framesetSplitbar>
    <w:w w:val="200" />
    <w:color w:val="0000FF" />
  </w:framesetSplitbar>
  …
</w:frameset>
```

If the noBorder element is also specified:

```
<w:frameset>
  <w:framesetSplitbar>
    <w:w w:val="200" />
    <w:color w:val="0000FF" />
    <w:noBorder w:val="true" />
  </w:framesetSplitbar>
  …
</w:frameset>
```

Then all frameset splitters are suppressed:



The noBorder element's val attribute has a value of `true`, which specifies that the splitters for this document must not be displayed. *end example*]

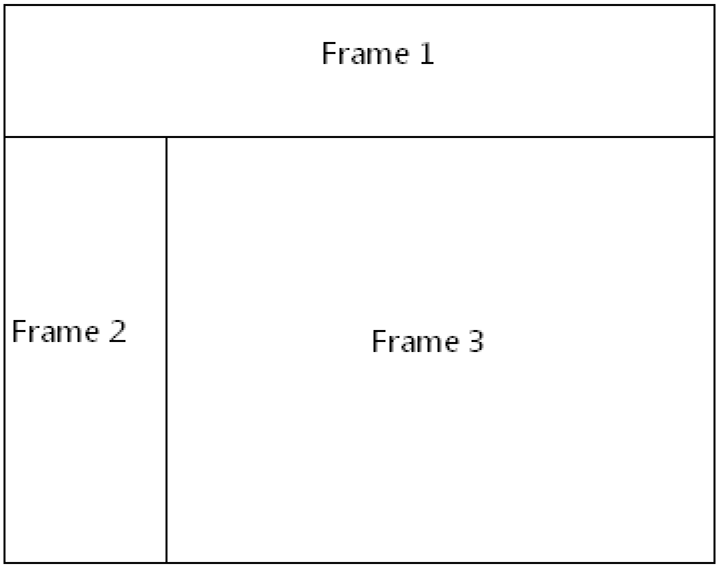
This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.2.32 noResizeAllowed (Frame Cannot Be Resized)

This element specifies whether or not the size of the current frame shall be modifiable (i.e. whether the frame can be resized) when the contents of this document are saved as HTML and displayed in a web browser. When this element is set, the size of the frame shall be set to its current values. This property is analogous to the `noresize` attribute on the `frame` element in HTML.

If this element is omitted, the size of the frame shall be modifiable (the frame can be resized when it is displayed).

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frameset>
    …
    <w:frame>
      <w:name w:val="Frame 2" />
      <w:noResizeAllowed w:val="true" />
    </w:frame>
    …
  </w:frameset>
</w:frameset>
```

The noResizeAllowed element has a val attribute of `true`, which specifies that the size of the frame specified by `Frame 2` must not be modifiable (the two borders which intersect that frame cannot be resized). *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.2.33 optimizeForBrowser (Disable Features Not Supported by Target Web Profile)

This element specifies whether applications should attempt to customize the output for any web page produced from this document, as well as the HTML output to which it should be customized. [*Example*: This might involve blocking any output which is not supported by that target output profile. *end example*] The target output profile is identified by the contents of the target attribute.

[*Example*: Consider a document whose web settings part contains the following WordprocessingML:

```
<w:webSettings>
   …
   <w:optimizeForBrowser w:target="W3C XHTML+CSS1" />
   <w:allowPNG w:val="on"/>
</w:webSettings>
```

The optimizeForBrowser element specifies that the output should be optimized for a specific output target. Since the value of the target attribute specifies a target of W3C XHTML 1.0 output using CSS 1, any HTML generated should fall within that profile. *end example*]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>target (Target Output Profile)</td><td>Specifies the version of HTML output targeted by the output of any web page produced by this document. This attribute shall only contain a string that represents an output profile defined by published standards and W3C recommendations. Product names shall not be used to define a profile.

The following reserved values and their targets are listed below:

<table>
<tr><th>Value</th><th>Target</th></tr>
<tr><td>W3C XHTML+CSS1</td><td>W3C XHTML 1.0 + CSS 1</td></tr>
<tr><td>W3C HTML4+CSS1</td><td>W3C HTML 4.01 + CSS 1</td></tr>
<tr><td>W3C XHTML+CSS2</td><td>W3C XHTML 1.0 + CSS 2</td></tr>
<tr><td>W3C HTML4+CSS2</td><td>W3C HTML 4.01 + CSS 2</td></tr>
</table>

[<i>Example</i>: For example, consider the following web publishing settings:

<code>&lt;w:optimizeForBrowser w:target="W3C XHTML+CSS2"/&gt;</code>

The target attribute explicitly declares that any web page generated from this document should target W3C XHTML 1.0 + CSS 2. <i>end example</i>]</td></tr>
</table>

Attributes	Description
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `1` or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `0` or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<… w:val="false"/>` The val attribute explicitly declares that the property is `false`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_OptimizeForBrowser) is located in §A.1. *end note*]

17.15.2.34 pixelsPerInch (Pixels per Inch for Graphics/Images)

This element specifies the number of pixels per inch (or density) that is used for the display of pictures or table cells when a WordprocessingML document is saved as a web page. The size that is specified by this element affects the size of the pictures or table cells relative to the size of text in the document. The pixels per inch (ppi) measurement is relative to the screen resolution, and the resulting physical dimensions of the resulting image or cell in pixels (which are used in web pages, but not for printed documents) are the result of the original dimensions (in inches) multiplied by the number of pixels per inch.

The range of values for this element is typically from 19 to 480 pixels per inch. The common settings for popular screen sizes are 72, 96, and 120 pixels per inch.

If this element is omitted, then a default size of 96 pixels per inch shall be used when determining the number of pixels for images and/or table cells within this document.

[*Note*: This setting is typically only specified if the target screen resolution for the web page is known, as defined by the targetScreenSz element (§17.15.2.41) to set the optimum screen size for the web page. *end note*]

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:pixelsPerInch w:val="200" />
</w:webSettings>
```

The pixelsPerInch element's val attribute has a value of 200, which specifies that all inches to pixels conversions done when saving this web page must be done assuming a transformation of 200 pixels per inch. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.15.2.35 right (Right Border for HTML div)

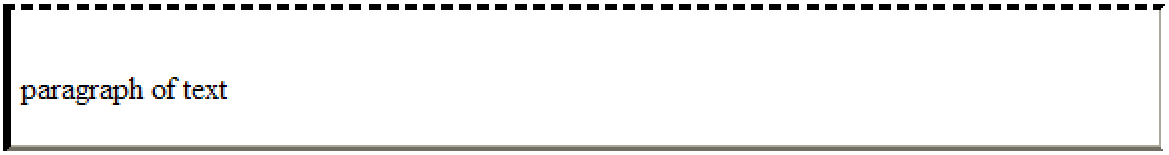
This element specifies the border which shall be displayed at the right of the boundaries of the current HTML div object.

If this element is omitted, then this HTML div object shall not have a right border.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" border-left-style:solid; border-right-style:groove; border-
right-width:1px; border-top-style:dashed; border-top-width:3px; border-bottom-
style:outset; border-bottom-width:3px">
      <p>paragraph of text</p>
    </div>
  </body>
</html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



paragraph of text

Now, when this document is saved in the WordprocessingML format, the information stored on the div elements is stored in the web setting part as follows:

```
<w:divs>
  <w:div w:id="1785730240">
    …
    <w:divBdr>
      <w:top w:val="dashed" w:sz="18" w:space="7" w:color="auto" />
      <w:left w:val="single" w:sz="24" w:space="4" w:color="auto" />
      <w:bottom w:val="outset" w:sz="18" w:color="auto" />
      <w:right w:val="threeDEngrave" w:sz="6" w:color="auto" />
    </w:divBdr>
  </w:div>
</w:divs>
```

The right element specifies border information about the right border for the single HTML div structure in the document; in this case, a 0.75 point bottom border of type threeDEngrave. The initial 1 pixel border was converted to 0.75 points using the following logic:

$$1\text{px} * \frac{1\text{ inch}}{96\text{ px}} * \frac{576\text{ eighth points}}{1\text{ inch}} = 6\text{ eighth points } (0.75\text{ points})$$

end example]

This element's content model is defined by the common border properties definition in §17.3.4.

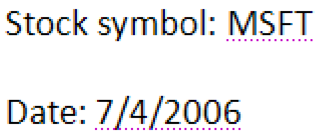
17.15.2.36 saveSmartTagsAsXml (Save Smart Tag Data in XML Property Bag)

This element specifies that the information pertaining to all smart tags () in the current document shall be saved into a separate XML-based property bag at the head of the web page when this WordprocessingML document is saved as a web page.

[*Rationale*: This setting is typically used when saving documents known to contain smart tags as web pages, in order to allow the smart tag data within the web page to be processed as a separate XML document by a separate parser, even though the actual HTML content of the resulting web page cannot be parsed by an XML-based parser. *end rationale*]

If this element is omitted, then the smart tag data of this document shall not be saved into a separate XML-compliant property bag within the HTML output when this document is saved as a web page.

[*Example*: Consider a WordprocessingML document which contains the following content:



Stock symbol: MSFT

Date: 7/4/2006

This document might typically write out the following HTML content:

```
<p>Stock symbol: <st1:stockticker>MSFT</st1:stockticker></p>
<p>Date: <st1:date ls="trans" Month="7" Day="4"
Year="2006"">7/4/2006</st1:date></p>
```

However, if the WordprocessingML document also contains the following content within the web settings part:

```
<w:webSettings>
  <w:saveSmartTagsAsXml w:val="true" />
</w:webSettings>
```

The saveSmartTagsAsXml element specifies that all smart tags in the document must also be saved into an XML property bag at the header of the file, for example:

```
<head>
  …
  <xml>
    <o:DocumentSmartTags>
      <st1:stockticker>MSFT</st1:stockticker>
      <st1:date ls="trans" Month="7" Day="4" Year="2006">7/4/2006</st1:date>
    </o:DocumentSmartTags>
  </xml>
  …
</head>
```

This header information is in addition to the normal HTML output. *end example*]

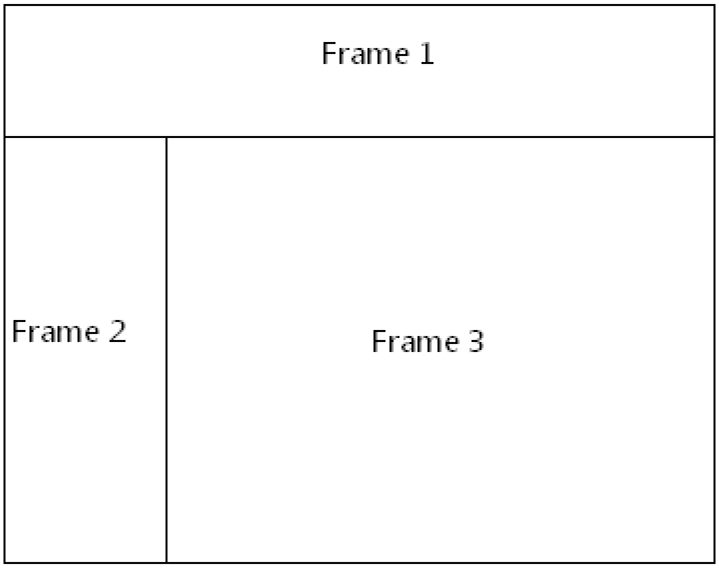
This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.2.37 scrollbar (Scrollbar Display Option)

This element specifies when a scrollbar shall be visible for the contents of the current frame. When this element is set, the val attribute determines exactly when the scrollbar shall be visible. This property is analogous to the `scrolling` attribute on the `frame` element in HTML.

If this element is omitted, the scrollbar shall only be displayed when the contents of the frame exceed the visible space for the frame (i.e. when the scrollbar is needed to display all of the content).

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frameset>
    …
    <w:frame>
      <w:name w:val="Frame 2" />
      <w:scrollbar w:val="auto" />
    </w:frame>
    …
  </w:frameset>
</w:frameset>
```

The scrollbar element has a val attribute of auto, which specifies that the frame must only display a scrollbar when it is needed to display all of its content. *end example*]

Attributes	Description
val (Scrollbar Display Option Value)	Specifies the criteria under which a scrollbar shall be displayed along with the contents of this frameset, as defined by the simple type referenced below. [*Example*: Consider a frameset definition within a WordprocessingML document which

Attributes	Description

	defines the following scrollbar visibility setting:
	`<w:frame>`
	`  <w:scrollbar w:val="on" />`
	`  …`
	`</w:frame>`
	The val attribute value of on specifies that the scrollbar must always be displayed, even when it is not needed (i.e. when it would be displayed disabled). *end example*]
	The possible values for this attribute are defined by the ST_FrameScrollbar simple type (§17.18.32).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FrameScrollbar) is located in §A.1. *end note*]

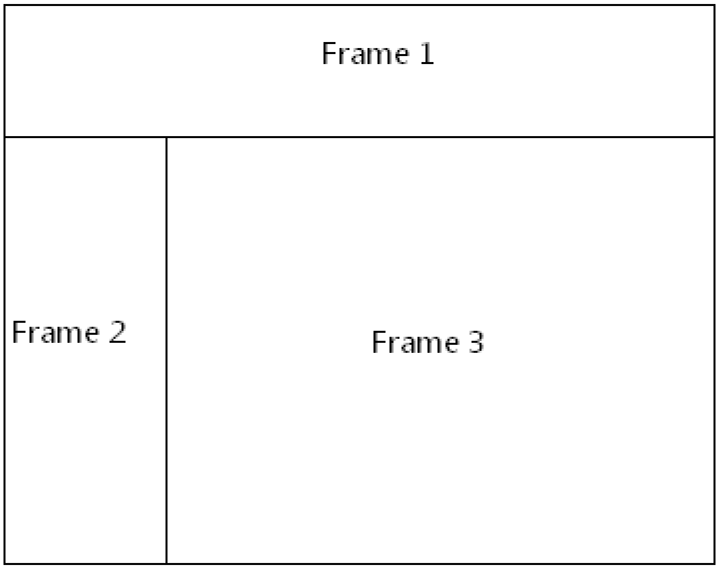
17.15.2.38 sourceFileName (Source File for Frame)

This element specifies the ID for the relationship which specifies the source file for a single frame within a frameset document.

The relationship referenced by this element's id attribute shall exist in the relationship part item for the Web Settings part,or this document shall be considered non-conformant. Also, the type of the relationship referenced by this element's id attribute shall be http://purl.oclc.org/ooxml/officeDocument/relationships/frame, or this document shall be considered non-conformant.

If this element is omitted, then no source file is present for the current frame, and one can be created dynamically as needed to display content within the frame.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frameset>
    …
    <w:frame>
      <w:name w:val="Frame 2" />
      <w:sourceFileName r:id="rId5" />
    </w:frame>
    …
  </w:frameset>
</w:frameset>
```

The sourceFileName element specifies that the contents of this frame must be the contents of the file targeted by the relationship with ID `rId5` in the web settings part's relationship part item. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: - http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element - http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element - http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element

Attributes	Description
	- http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements - http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element - http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rel) is located in §A.1. *end note*]

17.15.2.39 sz (Frame Size)

This element specifies the size for a single frame within a frameset.

This size shall be interpreted based on the contents of the frameLayout element (§17.15.2.17) for the parent frameset, as follows:

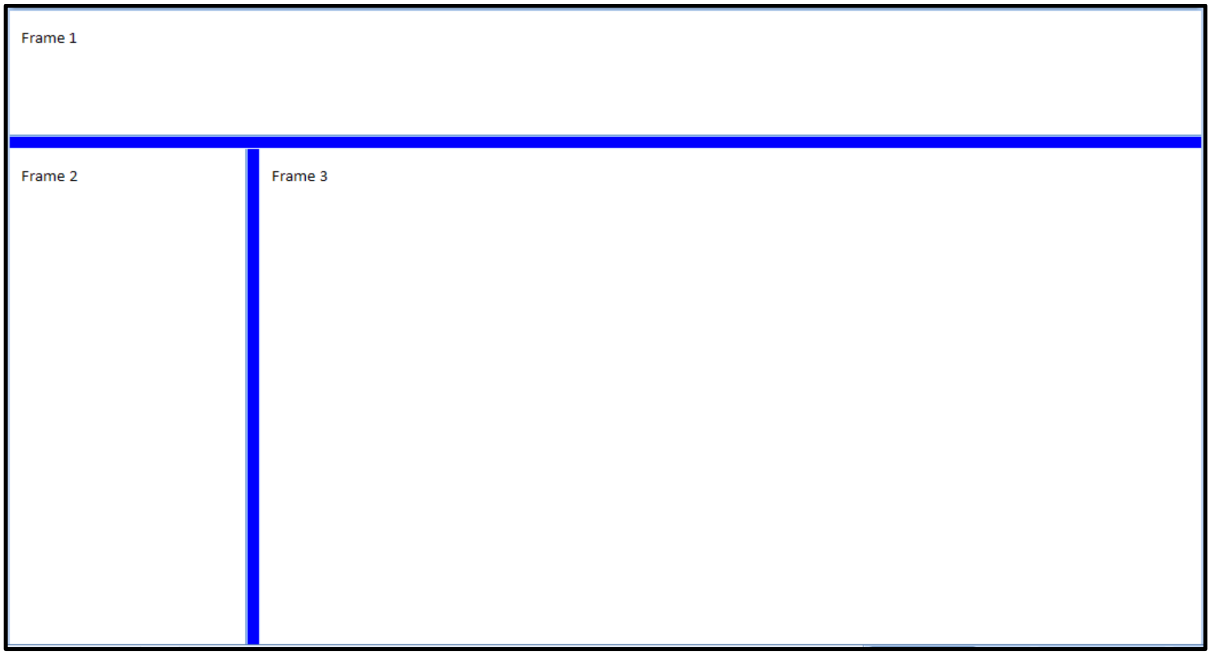
- If the val attribute on that element is `cols`, then this element specifies the width of the frame
- If the val attribute on that element is `rows`, then this element specifies the height of the frame

Once the axis of this measurement has been established using the criteria above, the actual value of the measurement shall be determined by the following:

- If the val attribute ends in a asterisk (*), then this measurement is a relative measurement (relative to all other frames in this frameset).
- If the val attribute ends in a percentage symbol (%), then this measurement is a percentage of the height and/or width of the parent window, respectively.
- Otherwise, the value of the val attribute specifies the size of the frame in pixels. This measurement shall be interpreted in the context of the pixelsPerInch element (§17.15.2.34) to determine the width of the resulting measurement in inches.

If this element is omitted, then no information shall be implied about the size of the current frame.

[*Example*: Consider a frameset consisting of the following three frames:



The following properties define the presentation of the top frame within this frameset:

```
<w:frameset>
  …
  <w:frame>
    <w:sz w:val="300" />
    <w:name w:val="Frame 1" />
  </w:frame>
  …
  <w:pixelsPerInch w:val="150" />
</w:frameset>
```

The sz element's val attribute specifies that the size of this frame is `300` - which translates to a height of exactly 300 pixels tall. In addition, this document specifies that the intended number of pixels per inch for this measurement is `150`, resulting in a 2" tall frame height. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`

Attributes	Description

	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.15.2.40 sz (Nested Frameset Size)

This element specifies the size for a frameset that has been nested within another frameset. If this size appears on a root frameset, then it can be ignored and the main frameset shall encompass the entire window.

This size shall be interpreted based on the contents of the frameLayout element (§17.15.2.17) for the parent frameset (not the current nested frameset), as follows:

- If the val attribute on that element is `cols`, then this element specifies the width of the frameset
- If the val attribute on that element is `rows`, then this element specifies the height of the frameset

Once the axis of this measurement has been established using the criteria above, the actual value of the measurement shall be determined by the following:

- If the val attribute ends in a asterisk (*), then this measurement is a relative measurement (relative to all other frames in this frameset).
- If the val attribute ends in a percentage symbol (%), then this measurement is a percentage of the height and/or width of the parent frameset, respectively.
- Otherwise, the value of the val attribute specifies the size of the frameset in pixels. This measurement shall be interpreted in the context of the pixelsPerInch element (§17.15.2.34) to determine the width of the resulting measurement in inches.

If this element is omitted, then no information shall be implied about the size of the current frameset.

[*Example*: Consider a nested frameset defined as follows:

```
<w:frameset>
  …
  <w:frameset>
    <w:sz w:val="50%" />
    …
  </w:frameset>
  …
  <w:pixelsPerInch w:val="150" />
</w:frameset>
```

The sz element's val attribute specifies that the size of this nested frameset is 50% - which translates to a width of fifty percent of the width of the parent frameset's height. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.15.2.41 targetScreenSz (Target Screen Size for Web Page)

This element specifies the ideal minimum target screen size (width by height, specified in pixels) on which web pages generated when saving this document is displayed. This setting can be used to optimize the output of web pages produced from this document.

If this element is omitted, then the target screen size for web pages produced from this document shall be assumed to be 800x600.

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:targetScreenSz w:val="1600x1200" />
</w:webSettings>
```

The targetScreenSz element's val attribute has a value of 1600x1200, which specifies that a target screen size of 1600 by 1200 pixels must be assumed when saving this document as a web page. *end example*]

Attributes	Description

val (Target Screen Size Value)	Specifies the target screen size for web pages produced by this document, as defined by the simple type referenced below.
	[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:
	`<w:webSettings>`
	`  <w:targetScreenSz w:val="1024x768" />`
	`</w:webSettings>`
	The val attribute has a value of 1024x768, which specifies that a target screen size of 1024 by 768 pixels must be assumed when saving this document as a web page. *end example*]
	The possible values for this attribute are defined by the ST_TargetScreenSz simple type (§17.18.86).

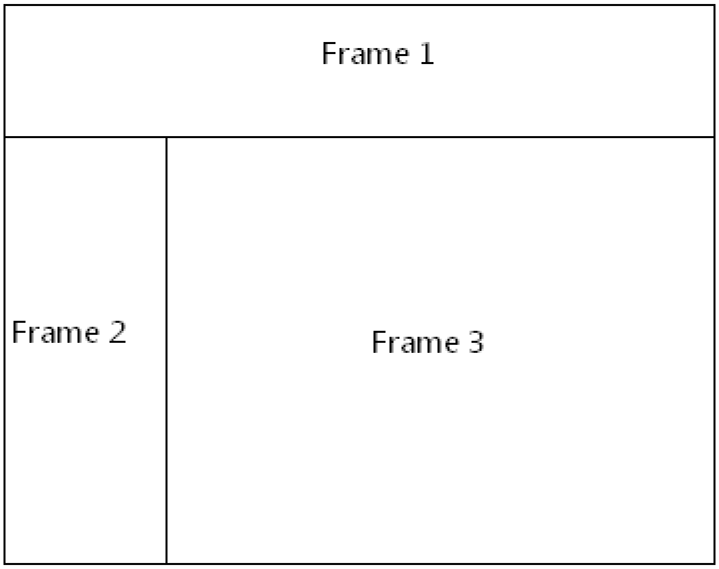
[*Note*: The W3C XML Schema definition of this element's content model (CT_TargetScreenSz) is located in §A.1. *end note*]

17.15.2.42 title (Frame or Frameset Title)

This element specifies advisory information about a single frame or frameset. The title information shall be stored in this element's val attribute. This property is analogous to the title attribute on the frame or frameset element in HTML.

If this element is omitted, then no title shall be associated with the given frame or frameset.

[*Example*: Consider a WordprocessingML document that serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  <w:title w:val="Our library of documents" />
  <w:frame>
    <w:name w:val="Frame 1" />
    <w:title w:val="Menu bar" />
  </w:frame>
  <w:frameset>
    <w:title w:val="Navigation and document collection" />
    …
    <w:frame>
      <w:name w:val="Frame 2" />
      <w:title w:val="Navigation bar" />
    </w:frame>
    <w:frame>
      <w:name w:val="Frame 3" />
      <w:title w:val="Documents" />
    </w:frame>
  </w:frameset>
</w:frameset>
```

The title element specifies supplementary information for each frame and frameset. In this case, the frames have titles of “`Menu bar`”, “`Navigation bar`”, and “`Documents`”, respectively, while the framesets have titles of “`Our library of documents`”, and “`Navigation and document collection`”, respectively. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.15.2.43 top (Top Border for HTML div)

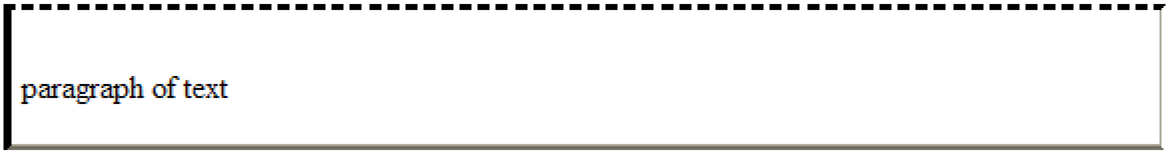
This element specifies the border which shall be displayed at the top of the boundaries of the current HTML `div` object.

If this element is omitted, then this HTML `div` object shall not have a top border.

[*Example*: Consider a simple HTML document defined as follows:

```
<html>
  <body>
    <div style=" border-left-style:solid; border-right-style:groove; border-
right-width:1px; border-top-style:dashed; border-top-width:3px; border-bottom-
style:outset; border-bottom-width:3px">
      <p>paragraph of text</p>
      </div>
    </body>
  </html>
```

This HTML would therefore normally appear as follows (image scaled appropriately):



paragraph of text

Now, when this document is saved in the WordprocessingML format, the information stored on the div elements is stored in the web setting part as follows:

```
  <w:divs>
    <w:div w:id="1785730240">
      …
      <w:divBdr>
        <w:top w:val="dashed" w:sz="18" w:space="7" w:color="auto" />
        <w:left w:val="single" w:sz="24" w:space="4" w:color="auto" />
        <w:bottom w:val="outset" w:sz="18" w:color="auto" />
        <w:right w:val="threeDEngrave" w:sz="6" w:color="auto" />
      </w:divBdr>
    </w:div>
  </w:divs>
```

The top element specifies border information about the top border for the single HTML div structure in the document; in this case, a 2.25 point bottom border of type dashed. The initial 3 pixel border was converted to 2.25 points using the following logic:

$$3\text{px} * \frac{1\text{ inch}}{96\text{ px}} * \frac{576\text{ eighth points}}{1\text{ inch}} = 18\text{ eighth points (2.25 points)}$$

end example]

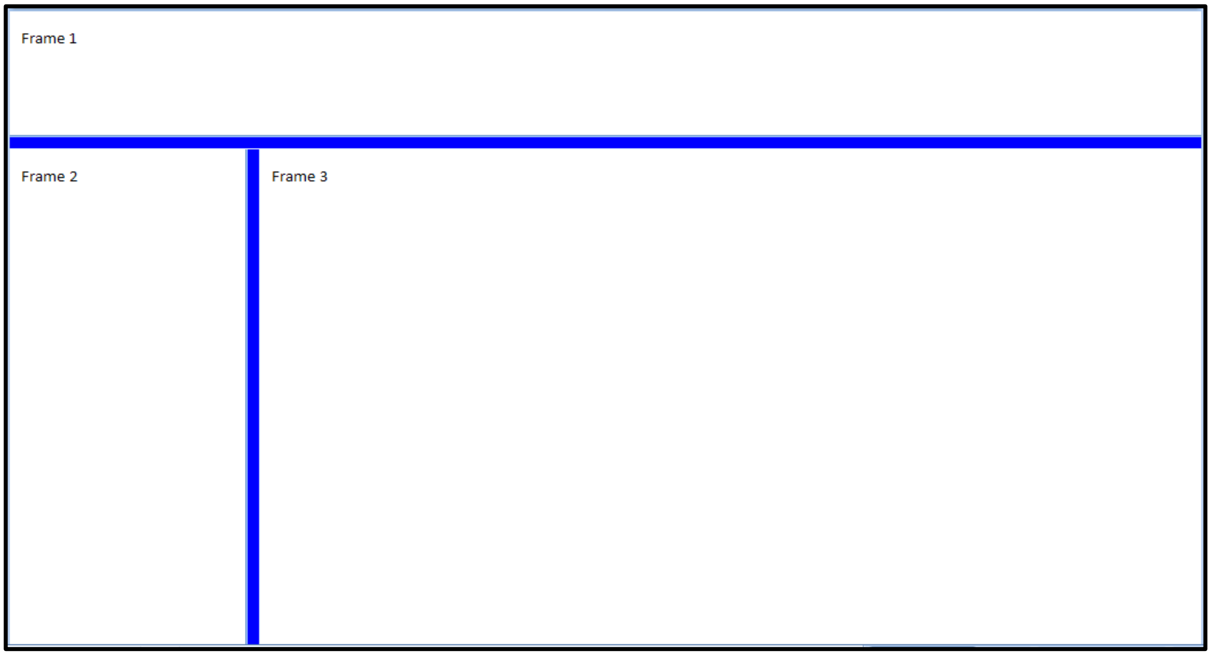
This element's content model is defined by the common border properties definition in §17.3.4.

17.15.2.44 w (Frameset Splitter Width)

This element specifies the width of the splitters within the frameset in this WordprocessingML document. This element shall only be honored on the root frameset for this document, and can be ignored for all nested framesets in this document.

If this element is omitted, then the default width of the splitters in this document shall be 4.5 points (90 twentieths of a point) wide. If the noBorder element (§17.15.2.31) is also specified, then this element shall be ignored and no splitters shall be displayed.

[*Example*: Consider a frameset consisting of the following three frames:



The following properties define the presentation of the splitter bars within this frameset:

```
<w:frameset>
  <w:framesetSplitbar>
    <w:w w:val="200" />
    <w:color w:val="0000FF" />
  </w:framesetSplitbar>
  …
</w:frameset>
```

The w element's val attribute specifies that the splitter must be 10 points (200 twentieths of a point) wide when the contents of this document are displayed. *end example*]

Attributes	Description
val (Measurement in Twentieths of a Point)	Specifies a positive measurement value, specified in twentieths of a point. This value is interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML element with a val attribute containing a positive measurement in twentieths of a point: `<… w:val="720" />`

Attributes	Description
	The val attribute has a value of 720, specifying that this measurement value is 720 twentieths of a point (0.5"). This value is interpreted by the parent element as needed. *end example*] The possible values for this attribute are defined by the ST_TwipsMeasure simple type (§22.9.2.14).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TwipsMeasure) is located in §A.1. *end note*]

17.15.2.45 webSettings (Web Page Settings)

This element specifies the set of web page settings that have been specified for a single WordprocessingML document. This element is the root element for the Web Settings part within a WordprocessingML document.

[*Example*: Consider the following WordprocessingML fragment for the web page settings in a WordprocessingML document:

```
<w:webSettings>
  …
  <w:frameset>
    …
  </w:frameset>
  <w:doNotUseLongFileNames w:val="true" />
  …
</w:webSettings>
```

The webSettings element contains all of the web page settings for this document. In this case, the web page settings specified for this document are: a frameset defined using the frameset element (§17.15.2.19); and a setting specifying that when this file is saved as a web page, all resulting files must not exceed 8 octets with 3 octet extension using the doNotUseLongFileNames element (§17.15.2.13). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_WebSettings) is located in §A.1. *end note*]

17.15.3 Language Compatibility Settings

The last group of settings in WordprocessingML are language compatibility settings. *Language compatibility settings* are optional settings used to specify changes appropriate to a subset of languages, but not usually appropriate in others. [*Example*: The doNotLeaveBackslashAlone setting changes the visual appearance of a specific character to match user expectation based on a historical use of that character in some code pages – users who have used those code pages would expect one value; those who have not would expect another. *end example*]. The behavior of each setting is fully defined in this subclause.

If language compatibility settings are needed, they are stored in the Document Settings part.

[*Note*: Although these settings can be applied in any WordprocessingML document, they are often applied when the document is created in one of the following contexts.

In the case of `ja-JP`, `ko-KR`, `zh-CN`, `zh-SG`, `zh-TW`, `zh-HK`, `zh-MO`, `ii-CN`:

- doNotLeaveBackslashAlone
- doNotExpandShiftReturn
- balanceSingleByteDoubleByteWidth
- adjustLineHeightInTable
- ulTrailSpace
- spaceForUL

In the case of `th-TH`, `lo-LA`, `km-KH`, `bo-CN`, `hy-AM`:

- applyBreakingRules

end note]

17.15.3.1 adjustLineHeightInTable (Add Document Grid Line Pitch To Lines in Table Cells)

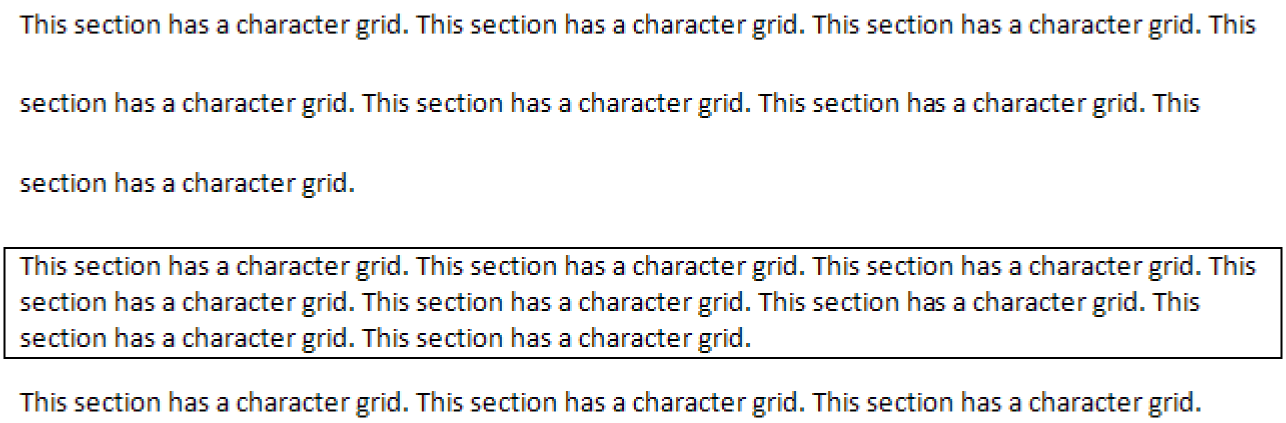
This element specifies whether a document grid defined using the docGrid element (§17.6.5) that specifies a line grid (manually adding additional pitch to each line in the section) shall also be applied to lines within table cells in this section.

Typically, when additional line pitch is added to all lines in a section via the document grid, it is not applied to text in tables. This element, when present with a val attribute value of `true` (or equivalent), specifies that additional line pitch shall be added to lines in table cells.

[*Example*: Consider a WordprocessingML document with a single section, whose document grid is defined such that 25.9 points of additional line pitch are added to each line in the section, as follows:

```
<w:docGrid w:type="lines" w:linePitch="518"/>
```

If text was entered into this section, the default behavior would have line pitch only added to lines which are not in a table cell:



This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid.

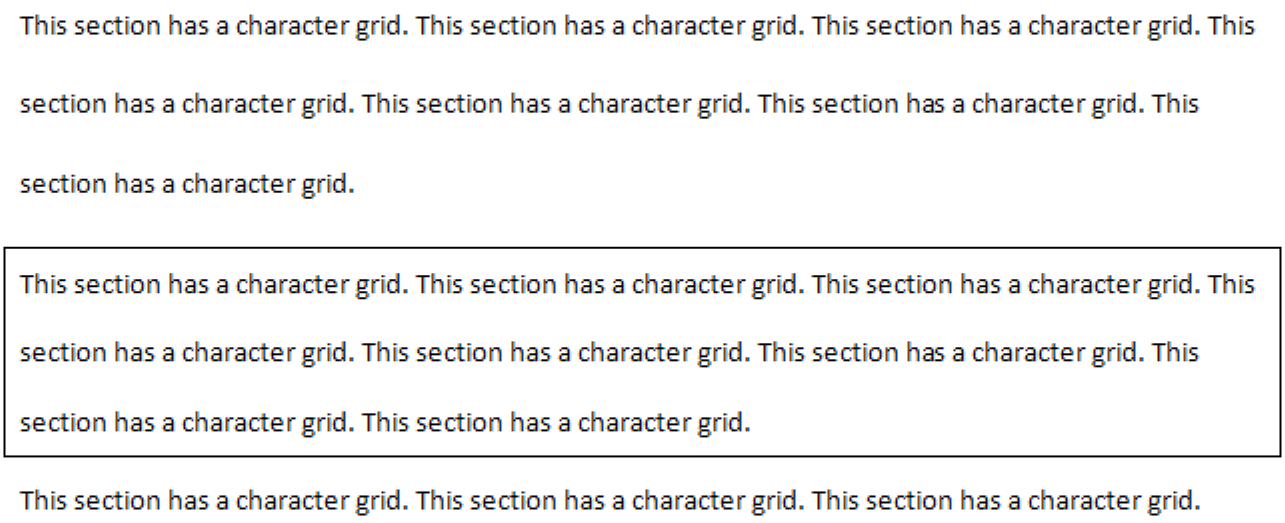
This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid.

This section has a character grid. This section has a character grid. This section has a character grid.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:adjustLineHeightInTable />
</w:compat>
```

Then all lines in this document would have the line pitch from the document grid added to them, resulting in the following output:



This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid.

This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid. This section has a character grid.

This section has a character grid. This section has a character grid. This section has a character grid.

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.3.2 applyBreakingRules (Use Legacy Ethiopic and Amharic Line Breaking Rules)

This element specifies whether applications shall use a legacy set of line breaking rules when determining line breaks for text consisting of Ethiopic and/or Amharic characters.

Typically, when line breaking this text, applications should allow line breaks to occur after a character between the UTF-16 (hexadecimal) values 0x1361 and 0x1368 when those characters appear in the document's content. This element, when present with a val attribute value of `true` (or equivalent), specifies that when a line break would occur after a character between the UTF-16 hexadecimal) values 0x1361 and 0x1368, the line break shall occur before all instances of these characters (i.e. no break opportunity shall be afforded after a character in this range).

[*Example*: Consider a WordprocessingML document with a series of Ethopic characters in this range. The default presentation would have any line breaks pushed before or after these characters, ensuring that the characters remain together on a single line.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:applyBreakingRules />
</w:compat>
```

Then a line break opportunity must be afforded at any point in a range of these characters, as needed. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

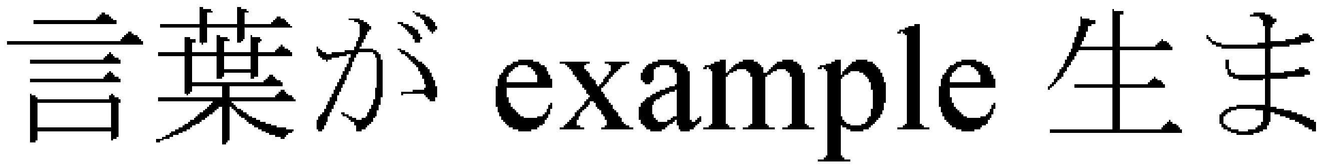
17.15.3.3 balanceSingleByteDoubleByteWidth (Balance Single Byte and Double Byte Characters)

This element specifies whether applications shall balance the width of Single Byte Character Set characters and Double Byte Character Set characters when rendering WordprocessingML documents. Specifically, this element specifies to adjust the fixed pitch fonts' half-width space character and full-width space character to attain a 1 to 2 ratio.

[*Note*: This element is used with East Asian content. Layout and line breaking for East Asian text is dependent on the character width. Half width characters (or Hankaku characters) are one half of an em wide, and full width characters (or Zenkaku characters) are one em wide. Legacy encoding often used a single byte to encode half-width characters and two bytes to encode full width characters. *end note*]

Typically, no adjustment is done on any character when it is displayed as part of a WordprocessingML document. This element, when present with a val attribute value of `true` (or equivalent), specifies that character sizes shall be adjusted as needed to meet the 1:2 ratio described above.

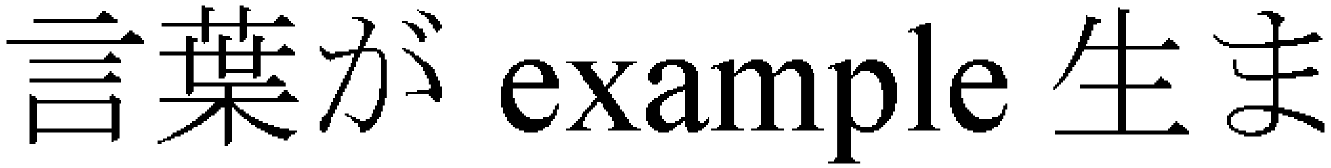
[*Example*: Consider a WordprocessingML document with both SBCS and DBCS characters. The default presentation would have the text displayed as follows:


言葉が example 生ま

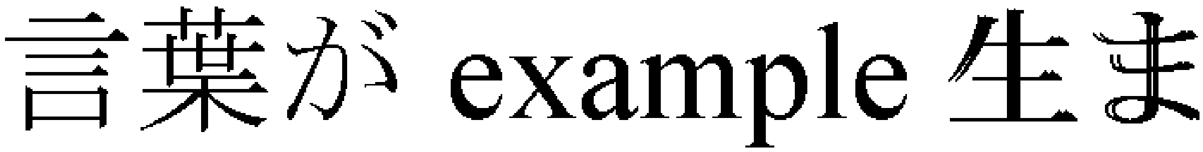
However, if this compatibility setting is turned on:

```
<w:compat>
  <w:balanceSingleByteDoubleByteWidth />
</w:compat>
```

Then this character-level adjustment must be performed, resulting in the following output:


言葉が example 生ま

This adjustment is usually very minute in nature, therefore the result is better illustrated by showing how the characters after the English text were pushed out due to the width balancing of that text:


言葉が example 生ま

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.3.4 compatSetting (Custom Compatibility Setting)

This element specifies a custom compatibility setting. The semantics for this element are implementation-defined. [*Note*: This element can be used to store the transitional compatibility settings specified in Part 4 of ISO/IEC 29500. *end note*]

[*Example*: Consider the following markup:

```
<w:compatSetting w:name="cooper" w:uri="http://www.example.com/exampleSetting"
w:val="1"/>
```

This custom compatibility setting specifies that:

- It is defined under the `http://www.example.com/exampleSetting` namespace.
- Its name is `cooper`.
- Its value is `1`.

Beyond this, the properties of this setting are implementation-defined by the creator of the http://www.example.com/exampleSetting namespace. *end example*]

Attributes	Description
name (Name of Setting)	Specifies the name of a custom compatibility setting. [*Example*: Consider the following markup: `<w:compatSetting w:name="cooper" w:uri="http://www.example.com/exampleSetting" w:val="1"/>` The name of this custom compatibility setting is `cooper`. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
uri (Namespace of Setting)	Specifies the namespace under which the compatibility setting is defined. [*Example*: Consider the following markup: `<w:compatSetting w:name="cooper" w:uri="http://www.example.com/exampleSetting" w:val="1"/>` The namespace of this custom compatibility setting is http://www.example.com/exampleSetting. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).
val (Value of Setting)	Specifies the value of a custom compatibility setting. [*Example*: Consider the following markup: `<w:compatSetting w:name="cooper" w:uri="http://www.example.com/exampleSetting" w:val="1"/>` The value of this custom compatibility setting is `1`. This value is interpreted using the implementation-defined behavior published by the creator of this property. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

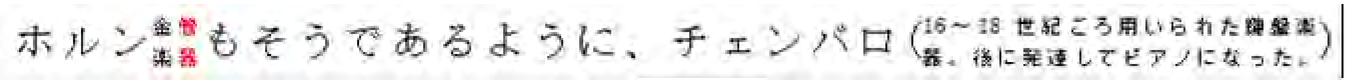
[*Note*: The W3C XML Schema definition of this element's content model (CT_CompatSetting) is located in §A.1. *end note*]

17.15.3.5 doNotExpandShiftReturn (Don't Justify Lines Ending in Soft Line Break)

This element specifies whether applications should fully justify the contents of incomplete lines which end in a soft line break when the parent paragraph is fully justified using the jc element (§17.3.1.13).

Typically, applications shall fully justify all lines in a paragraph when that setting is specified using the jc element except for the last line in the paragraph (the line ending with the paragraph mark). This element, when present with a val attribute value of `true` (or equivalent), specifies that any line which ends in a soft line break shall also not be fully justified when the paragraph specifies that setting.

[*Example*: Consider a WordprocessingML document with a paragraph whose first single line consists of East Asian characters followed by a soft paragraph mark. The default presentation would have the contents of that line fully justified:

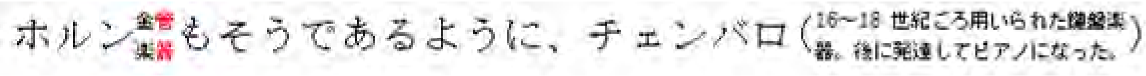


ホルン金管楽器もそうであるように、チェンバロ（16～18 世紀ころ用いられた鍵盤楽器。後に発達してピアノになった。）

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:doNotExpandShiftReturn />
</w:compat>
```

Then this line is not fully justified, as it ends with a soft line break, resulting in the following output:



ホルン金管楽器もそうであるように、チェンバロ（16～18 世紀ころ用いられた鍵盤楽器。後に発達してピアノになった。）

end example]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.15.3.6 doNotLeaveBackslashAlone (Display Backslash As Yen Sign)

This element specifies whether applications should autodisplay the backslash character using the yen character when displaying the contents of this document.

Typically, no automatic display-only conversion of one character to another is performed. This element, when present with a val attribute value of `true` (or equivalent), specifies that all occurances of the backslash character (\, U+005C) shall automatically be displayed using the yen symbol (¥, U+00A5) when the contents of the document are displayed. This setting does not change the Unicode value of the character stored in the underlying WordprocessingML document.

[*Rationale*: In Japanese code page 932, 0x5C is the yen sign (whereas, in most other code pages, it is the reverse solidus–also known as the backslash). In order to accommodate the user expectation that this code point appear as the yen sign, this setting dictates that the character be remapped, for display only, to the Unicode character ¥, such that the expected appearance is maintained. *end rationale*]

©ISO/IEC 2012 – All rights reserved

[*Example*: Consider a WordprocessingML document containing the following:

```
Hello \ world.
```

The default presentation would have exactly that:

```
Hello \ world.
```

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:doNotLeaveBackslashAlone />
</w:compat>
```

Then the backslash would be displayed as ¥, resulting in the following output:

```
Hello ¥ world.
```

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.3.7 spaceForUL (Add Additional Space Below Baseline For Underlined East Asian Text)

This element specifies whether East Asian content in a WordprocessingML document which has been underlined using the u element shall have additional descent added to the properties of the font in order to ensure that there is adequate spacing between the characters in the font and the underlining applied to the text.

Typically, no adjustments are made to the contents of text runs containing East Asian text which have been underlined. This element, when present with a val attribute value of `true` (or equivalent), specifies that whenever the following conditions are met:

- The text run contains East Asian characters
- The text run is not using baseline font alignment as defined by the textAlignment property

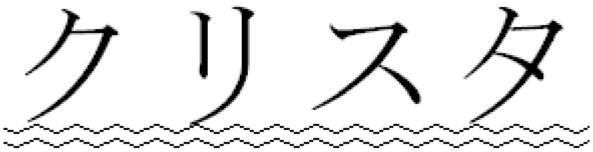
That the larger of the following two values is added to the descent property of that font in order to provide additional padding between the text characters and the underline:

- 3 percent of the font size
- 40 twentieths of a point (31 twentieths of a point for Japanese text)

[*Example*: Consider a WordprocessingML document consisting of a single run of underlined Japanese text, as follows:

```
<w:p>
  <w:r>
    <w:rPr>
      <w:u w:val="double" />
    </w:rPr>
    <w:t>クリスタ</w:t>
  </w:r>
</w:p>
```

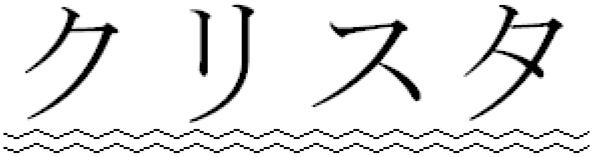
If this document is displayed, then the text is laid out along with the underline, as follows:



However, if this compatibility setting is turned on:

```
<w:compat>
  <w:spaceForUL />
</w:compat>
```

Then the additional descent specified using the logic above is added to the text, resulting in the following output:



end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.15.3.8 ulTrailSpace (Underline All Trailing Spaces)

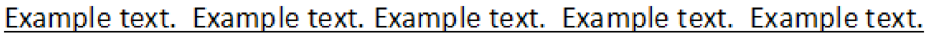
This element specifies whether applications shall display underlining beneath all trailing spaces in the contents of a line when those contents are underlined. *Trailing spaces* are all space characters which are not followed by non-space characters on the same line.

Typically, applications do not display underlining on all trailing spaces which have the underline property applied to them. This element, when present with a val attribute value of `true` (or equivalent), specifies that all characters with underline applied, including trailing spaces, shall display underlining if it is applied to that content.

[*Example*: Consider a WordprocessingML document with the following line of Latin alphabetical character and punctuation, trailed by a series of spaces:

```
<w:r>
  <w:rPr>
    <w:u w:val="single"/>
  </w:rPr>
  <w:t>Example text. Example text. Example text. Example text. Example text.
</w:t>
</w:r>
```

The default presentation would have no underlining on those trailing spaces:

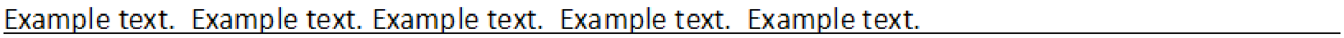


Example text. Example text. Example text. Example text. Example text.

However, if this compatibility setting is turned on:

```
<w:compat>
  <w:ulTrailSpace />
</w:compat>
```

Then all trailing spaces would be underlined, resulting in the following output:



Example text. Example text. Example text. Example text. Example text.

end example]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.16 Fields and Hyperlinks

Most text in a word processing document is static; that is, unless it is directly changed as the result of editing, its contents remain the same, no matter how the rest of the document might change. However, certain useful pieces of information can change value over the life of a document. Consider the case of a reference to a page number, as in "For more information on this topic, see page 56." Clearly, hard coding the page number as 56 means that that number needs to be manually replaced as the document's size or layout is changed. Even a simple change to any margin, line spacing, or font size can invalidate such references.

Fields provide a mechanism for placeholders, such as page reference numbers, that can be added to a document such that those placeholders are replaced by their corresponding values when the document is rendered for display or print. Other applications for fields include, but are not limited to, automatic numbering of tables and figures, document creation and current date and time, document author information, and the computation of totals for a table column.

A *field* is a set of codes that instructs a WordprocessingML consumer to insert text, graphics, page numbers, and other material into a document automatically. [*Example*: The DATE field causes the current date to be inserted.

end example] The text or graphics inserted into a document when a consumer carries out a field's codes is referred to as the *field result* (or *field value*) for that field. The default value for a field result is an empty string. The act of carrying out a field's codes is referred to as a *field update*. A field update shall be carried out such that the content contained within the field result correctly reflects the intended semantics of the field codes (as defined in the subclauses of §17.16.5). For a simple field defined via the fldSimple element (§17.16.19), this means replacing its child elements with the appropriate WordprocessingML content; for a complex field represented by use of properly matched fldChar elements (§17.16.18), this means replacing any content between the fldChar element with an fldCharType attribute value of separate and an fldCharType attribute value of end. As to when any field is updated is outside the scope of ISO/IEC 29500.

17.16.1 Syntax

The syntax rules in this subclause follow the system shown in ISO/IEC 14977: literal text is surrounded by double-quotes (or by apostrophes); the left-square-bracket and right-square-bracket designate the start and end of an option; the left-curly-bracket and right-curly-bracket designate the start and end of a sequence of one-or-more items; the vertical-line indicates an alternative; and each rule ends with a semicolon. Whenever hyphen is used as the exception-symbol (as per ISO/IEC 14977), it is surrounded by white space, and further clarified by a comment.

The syntax rules below were derived from the field-specific rules in the detailed definitions shown in §17.16.5.*. [*Note*: In order to produce an automated verifier, field-specific rules in subclauses 17.16.5.* must also be considered. *end note*]

When used in narrative, production names are set in an italic style, as in *comparison*, *field-argument*, and *switches*.

The syntax of a field is as follows:

```
field=
   field-type,  [switches]  ;
field-type=
   date-and-time  |
   document-automation  |
   document-information  |
   document-property  |
   equations-and-formulas  |
   index-and-tables  |
   links-and-references  |
   mail-merge  |
   numbering  |
   user-information  |
   form-field  |  user-defined-field  ;
user-defined-field=
   letter,  {letter}  ;
```

```
date-and-time=
   "CREATEDATE"  |  "DATE"  |  "EDITTIME"  |  "PRINTDATE"  |
   "SAVEDATE"  |  "TIME"  ;
document-automation=
   "COMPARE",  comparison  |
   "DOCVARIABLE",  field-argument  |
   "GOTOBUTTON",  2 * field-argument  |
   "IF",  comparison,  2 * field-argument  |
   "MACROBUTTON",  2 * field-argument  |
   "PRINT",  field-argument  |
   "PRIVATE"  ;
document-information=
   "FILENAME"  |  "FILESIZE"  |  "LASTSAVEDBY"  |  "NUMCHARS"  |
   "NUMPAGES"  |  "NUMWORDS"  |  "TEMPLATE"  ;
document-property=
   "AUTHOR",  [field-argument]  |
   "COMMENTS",  [field-argument]  |
   "DOCPROPERTY",  docprop-category  |
   "KEYWORDS",  [field-argument]  |
   "SUBJECT",  [field-argument]  |
   "TITLE",  [field-argument]  ;
equations-and-formulas=
   "=",  expression  |
   "ADVANCE"  |
   "SYMBOL",  field-argument  ;
index-and-tables=
   "INDEX"  |
   "RD",  field-argument  |
   "TA"  |
   "TC",  field-argument  |
   "TOA"  |
   "TOC"  |
   "XE",  field-argument  ;
```

```
links-and-references=
   "AUTOTEXT", field-argument |
   "AUTOTEXTLIST", field-argument |
   "BIBLIOGRAPHY" |
   "CITATION", field-argument |
   "HYPERLINK", field-argument |
   "INCLUDEPICTURE", field-argument |
   "INCLUDETEXT", field-argument, [field-argument] |
   "LINK", 2 * field-argument, [field-argument] |
   "NOTEREF", field-argument |
   "PAGEREF", field-argument |
   "QUOTE", field-argument |
   ["REF"], field-argument | (* see §17.16.5.51 *)
   "STYLEREF", field-argument ;
mail-merge=
   "ADDRESSBLOCK" |
   "ASK", 2 * field-argument |
   "DATABASE" |
   "FILLIN", [field-argument] |
   "GREETINGLINE" |
   "MERGEFIELD", field-argument |
   "MERGEREC" |
   "MERGESEQ" |
   "NEXT" |
   "NEXTIF", comparison |
   "SET", 2 * field-argument |
   "SKIPIF", comparison ;
numbering=
   "LISTNUM", [field-argument] |
   "PAGE" |
   "REVNUM" |
   "SECTION" |
   "SECTIONPAGES" |
   "SEQ", identifier, [field-argument] ;
user-information=
   "USERADDRESS", [field-argument] |
   "USERINITIALS", [field-argument] |
   "USERNAME", [field-argument] ;
form-field=
   "FORMCHECKBOX" | "FORMDROPDOWN" | "FORMTEXT" ;
bookmark-name=
   identifier ;
```

©ISO/IEC 2012 – All rights reserved

```
cell-name=
   column-name,  row-name  ;
cell-range=
   cell-name,  colon,  cell-name  |
   row-name,  colon,  row-name  |
   column-name,  colon,  column-name  ;
cell-reference=
   cell-name  |
   cell-name,  {  comma,  cell-name  }  |
   cell-range  ;
character=
  as defined by the production Char in the XML 1.0 specification, §2.2.
colon=
   ":"  ;
column-name=
   letter  +  [{letter}]  ;   (* allowing for A, … Z, AA, …., ZZ, AAA, … column
naming *)
comma=
   ","    ;
comparison=
   expression,  comparison-operator,  expression  ;
comparison-operator=
   "="  |  "<>"  |  "<"  |  "<="  |  ">"  |  ">="  ;
constant=
   number  ;
date-and-time-formatting-switch=
   "\@",  switch-argument  (* as defined in §17.16.4.1 *) ;
docprop-category=
   "AUTHOR"  |  "BYTES"  |  "CATEGORY"  |  "CHARACTERS"  |
   "CHARACTERSWITHSPACES"  |  "COMMENTS"  |
   "COMPANY  |  CREATETIME"  |  "HYPERLINKBASE"  |
   "KEYWORDS"  |  "LASTPRINTED"  |  "LASTSAVEDBY"  |
   "LASTSAVEDTIME"  |  "LINES"  |  "MANAGER"  |
   "NAMEOFAPPLICATION"  |  "ODMADOCID"  |  "PAGES"  |
   "PARAGRAPHS"  |  "REVISIONNUMBER"|  "SECURITY"  |
   "SUBJECT"  |  "TEMPLATE"  |  "TITLE"  |
   "TOTALEDITINGTIME"  |  "WORDS"  ;
double-quote=
   '"'   ;  (* one double-quote character *)
```

```
expression=
   "(",  expression,  ")"  |
   comparison  |
   constant  |
   prefix-operator,  expression  |
   expression, infix-operator,  expression  |
   field  |
   bookmark-name,  [cell-reference]  |
   function  |
   cell-reference  |
   text  |
   double-quote,  text,  double-quote  ;
field-argument=
   text  |
   double-quote,  text,  double-quote  ;
field-specific-switch=
   "\",  character,  [character],  [field-argument]  ;
      (* no whitespace is permitted after the backslash,
      also see definition of each field in §17.16.5 *)
formatting-switch=
   date-and-time-formatting-switch  |
   numeric-formatting-switch  |
   general-formatting-switch  ;
fractional-part=
   decimal-digit  {decimal-digit}  ;
full-stop=
   "."    ;  (* also known as "period" *)
```

```
function=
   "ABS(",  expression,  ")"  |
   "AND(",  expression,  ","  expression,  ")"  |
   "AVERAGE(",  list,  ")"  |
   "COUNT(",  list,  ")"  |
   "DEFINED(",  expression,  ")"  |
   "FALSE"  |
   "INT(",  expression,  ")"  |
   "MAX(",  list,  ")"  |
   "MIN(",  list,  ")"  |
   "MOD(",  expression,  ",",  expression,  ")"  |
   "NOT(",  expression,  ")"  |
   "OR(",  expression,  ",",  expression,  ")"  |
   "PRODUCT(",  list,  ")"  |
   "ROUND(",  expression,  ",",  expression,  ")"  |
   "SIGN(",  expression,  ")"  |
   "SUM(",  list,  ")"  |
   "TRUE"  ;
general-formatting-switch=
   "\*",  switch-argument  ;  (* as defined in §17.16.4.3 *)
identifier=
   character  + [{character}]  ;
infix-operator=
   "-"  |  "^"  |  "*"  |  "/"  |  "%"  |  "+"  |  "-"  |  "="  |
   "<>"  |  "<"  |  "<="  |  ">"  |  ">="  ;  (* see §17.16.3.3 *)
letter=
  "a"|"b"|"c"|"d"|"e"|"f"|"g"|"h"|"i"|"j"|"k"|"l"|"m"|
  "n"|"o"|"p"|"q"|"r"|"s"|"t"|"u"|"v"|"w"|"x"|"y"|"z"|
  "A"|"B"|"C"|"D"|"E"|"F"|"G"|"H"|"I"|"J"|"K"|"L"|"M"|
  "N"|"O"|"P"|"Q"|"R"|"S"|"T"|"U"|"V"|"W"|"X"|"Y"|"Z" ;
list=
   expression,  {list-separator,  expression}  ;
list-separator=
   comma | semicolon  ;
      (* depending on the document's listSeparator, §17.15.1.56*)
number=
   whole-number-part,  [full-stop]  |
   full-stop,  fractional-part  |
   whole-number-part,  full-stop,  fractional-part  ;
numeric-formatting-switch=
   "\#",  switch-argument  ;       (* as defined in §17.16.4.2 *)
prefix-operator=
   "-"  ;
```

```
row-name=
   whole-number-part  ;
semicolon=
   ";"    ;
switch-argument=
   text  |
   double-quote,  text,  double-quote  ;
switches=
   {field-specific-switch},  [formatting-switch]  ;
text=
   character,  {character}  ;
whole-number-part=
   decimal-digit,  {decimal-digit}  ;
```

Formulas (also called expressions) are discussed in §17.16.3, constants are discussed in §17.16.3.1, bookmarks are discussed in §17.16.3.2, operators are discussed in §17.16.3.3, functions are discussed in §17.16.3.4, table cell references are discussed in §17.16.3.5, and formatting switches are discussed in §17.16.4.

The semantics of a field having a *user-defined-field type* are unspecified.

If the *text* in a *field-argument* or *switch-argument* contains white space, the delimiting double-quote characters shall be present; otherwise, they are optional. To include a double-quote character in *text*, it shall be preceded with a backslash (\). [*Example*: The field argument `"\"name\""` results in the argument's actually being `"name"`. *end example*] To include a backslash character in *text*, it shall be preceded with another backslash (\). [*Example*: File system pathnames on some systems use a backslash as a directory separator, as in the field

```
INCLUDETEXT "E:\\ReadMe.txt"
```

in which case, each such separator needs to be preceded with a backslash, as shown above. *end example*]

Arbitrary amount of white space can occur before the first token, after the last token, and between successive tokens, including no white space at all, except that in a *field-specific-switch*, no white space is permitted between the initial “\” token and the *character* that follows it.

[*Example*: Here are examples of some fields:

```
DATE
DATE \@ "dddd, MMMM dd, yyyy"
DATE \@ "dddd, MMMM dd, yyyy" \h
```

The field result of all three is today's date: The first field uses some implementation-defined format and the Gregorian calendar; the second field uses the specified format and the Gregorian calendar; and the third field uses the specified format and the Hijri lunar calendar. When rendered in a US-English context on December 31, 2005, the results of these fields were as follows:

```
12/31/2005
Saturday, December 31, 2005
AsSabt, Thoul Ki'dah 30, 1426
```

end example]

The names of fields are alphabetic tokens [*Example*: Some field-type names are ASK, COMMENTS, NEXT, and SET. *end example*]. These tokens are called *field-type names*. Field-type names are case-insensitive. [*Example*: The field-type names DATE, Date, dAtE, and date are equivalent. *end example*]

The *character*s immediately following the "\" in *field-specific-switch*are case-insensitive. [*Example*: \b and \B are equivalent. *end example*]

There is no ordering of *field-specific-switch* entries in *switches*.

17.16.2 XML representation

Fields shall be implemented in XML using either of two approaches:

- As a *simple field implementation*, using the fldSimple element, or
- As a *complex field implementation*, using a set of runs involving the fldChar and instrText elements.

For a simple field implementation, only one element, fldSimple, shall be used, in which case, its instr attribute shall contain a *field*, and the body of the element shall contain the most recently updated field result. [*Example*: Here is the corresponding XML for a simple field implementation of DATE:

```
<w:fldSimple w:instr="DATE">
  <w:r>
    <w:t>12/31/2005</w:t>
  </w:r>
</w:fldSimple>
```

end example]

For a complex field implementation, a set of runs shall be used with each run containing, in sequence, the following elements:

- fldChar with attribute fldCharType value begin,
- One or more instrText elements, which, collectively, contain a complete *field*,
- Optionally,
  - fldChar with attribute fldCharType value separate, which separates the field from its field result,
  - Any number of runs and paragraphs that contains the most recently updated field result, and
- fldChar with attribute fldCharType value end.

[*Note*: Fields that are for display purposes only have no need to, and do not, store a field result. *end note*][*Example*: Here is the corresponding XML for a complex field implementation of DATE:

```
<w:r>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r>
  <w:instrText xml:space="preserve"> DATE </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r>
  <w:t>12/31/2005</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

end example]

[*Note*: Every simple field implementation for a given field has a corresponding complex field implementation. However, not every complex field implementation has a corresponding simple field implementation. If some characters in a *field* have different run properties than others, that field must be implemented using multiple runs, and that requires that complex field implementation be used. For an example, see §17.16.4.3, where the first letter of a DATE field is made bold, underlined, and red, while the other letters have none of these properties. *end note*]

As shown in §17.16.1, the *instruction* of one *field* can be another *field*, allowing fields to nest. In such cases, the XML run sequence for the inner field is defined at the point of reference for that inner field, inside the outer field's XML run sequence. [*Example*: Consider the following sentence:

It's `IF DATE \@ "M-d"<>"1-1" "not "` new year's day.

The `IF` field contains the nested field DATE `\@ "M-d"`. When updated, on January 1 of any year, the result sentence is "It's new year's day." On all other days of the year, the resulting sentence is "It's not new year's day."

Here is one way of writing the corresponding XML:

```
<w:r>
  <w:t xml:space="preserve">It’s </w:t>
</w:r>
<w:r …>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
```

```
  <w:r>
    <w:instrText xml:space="preserve">IF </w:instrText>
  </w:r>
  <w:r>
    <w:fldChar w:fldCharType="begin"/>
  </w:r>
  <w:r>
    <w:instrText xml:space="preserve"> DATE  \@ "M-d"  </w:instrText>
  </w:r>
  <w:r>
    <w:fldChar w:fldCharType="separate"/>
  </w:r>
  <w:r …>
    <w:instrText>1-4</w:instrText>
  </w:r>
  <w:r>
    <w:fldChar w:fldCharType="end"/>
  </w:r>
  <w:r>
    <w:instrText>&lt;&gt;"1-1" "not "</w:instrText>
  </w:r>
  <w:r …>
    <w:fldChar w:fldCharType="separate"/>
  </w:r>
  <w:r …>
    <w:t xml:space="preserve">not </w:t>
  </w:r>
  <w:r …>
    <w:fldChar w:fldCharType="end"/>
  </w:r>
  <w:r>
    <w:t>new year’s day!</w:t>
  </w:r>
```

end example]

17.16.3 Formulas and expressions

A field instruction can involve a calculation via a *formula*, which is simply an *expression* that is an arbitrary complex arithmetic expression involving constants (§17.16.3.1), bookmarks that refer to *expression*s (§17.16.3.2), arithmetic and logical operators (§17.16.3.3), functions (§17.16.3.4), values of cells in a table (§17.16.3.5), and *field*s that result in a single value. *expression* can contain grouping parentheses to document the default precedence or to override it.

All arithmetic terms in an *expression* are real numbers. Infinities and NaN (Not-a-Number) are not supported. [*Example*: In the expression 1/3, although the operands appear to be integers, they are, in fact real numbers, and the result is 0.33. *end example*]

17.16.3.1 Constants

A constant is a number. Exponents are not supported.

[*Example*: Here are some constants: 1234, 1234.560, 1234., and .1234. *end example*]

17.16.3.2 Bookmarks

Any arbitrary piece of text and/or graphics in a WordprocessingML document can be assigned a name, called a *bookmark*. If a bookmark references text that represents an *expression*, that bookmark's name can be used as an operand in another *expression*. If a whole field is bookmarked, its bookmark name can also be used as an operand in an *expression*. [*Example*: Given that X is a bookmark for the text 4, Y is a bookmark for the text 2, and Result is a bookmark for the following field:

```
=X + Y
```

the field

```
=Result * 10
```

has the result 60. *end example*]

17.16.3.3 Operators

The *operators* permitted in *expression* are:

Operators		
Operator	Description	Precedence
-	Unary minus	highest
^	Powers and roots	
*	Multiplication	
/	Division	
%	Percentage	
+	Addition	
-	Subtraction	
=	Equal to	
<>	Not equal to	
<	Less than	lowest
<=	Less than or equal to	
>	Greater than	

©ISO/IEC 2012 – All rights reserved

Operators		
>=	Greater than or equal to	

Operators in *expression* having the same precedence associate left-to-right.

[*Example*: Given that X is a bookmark for the text 4, and Y is a bookmark for the text 2, the field

```
=((-1 + X^2) * 3 - Y)/2
```

produces the result `21.5`. *end example*]

The equality, inequality, and relational operators yield 1 for true and 0 for false. An expression with value 0 tests logically false while one with any non-zero value tests true.

17.16.3.4 Functions

A *function* is a predefined procedure that computes and returns a result. Functions defined below with a parameter list of *list* accept two or more arguments separated by commas (,) or semicolons (;). As to which separator is permitted, is defined by the document's listSeparator (§17.15.1.56) element. Arguments to functions can be constants, formulas, or bookmark names that refer to constants or formulas. The functions AVERAGE, COUNT, MAX, MIN, PRODUCT, and SUM can also accept references to table cells as arguments. In the context of a table cell, functions taking a *list* also accept a single argument that designates a named-list of contiguous cells (§17.16.3.5). Function names are not case-sensitive, and white space can occur between a function's name and its argument list, if any.

The functions supported are as follows:

Functions	
Function	Description
ABS(*x*)	Returns the absolute value of *x*.
AND(*x*,*y*)	Returns 1 if the logical expressions *x* and *y* are both true; otherwise, it returns 0.
AVERAGE(*list*)	Returns the average value of the items in *list*.
COUNT(*list*)	Returns the number of items in *list*.
DEFINED(*x*)	Returns 1 if the expression *x* is well formed; otherwise, it returns 0.
FALSE	Returns 0.
INT(*x*)	Returns the value of the integer part of *x*.
MAX(*list*)	Returns the largest value in *list*.
MIN(*list*)	Returns the smallest value in *list*.
MOD(*x*,*y*)	Returns the value *x* - *ny*, for some integer *n* such that, if *y* is nonzero, the result has the same sign as *x* and magnitude less than the

Functions	
	magnitude of *y*. If *y* is zero, a diagnostic shall be issued. (*y* need not be a whole number.) [*Example*: `MOD(21,5)` results in 1 `MOD(21,-5)` results in 1 `MOD(-21,5)` results in -1 `MOD(-21,-5)` results in -1 *end example*]
`NOT(`*x*`)`	Returns 0 if the logical expression *x* is true, or 1 if the expression is false.
`OR(`*x*`,`*y*`)`	Returns 1 if either or both logical expressions *x* and *y* are true; otherwise, it returns 0.
`PRODUCT(`*list*`)`	Returns the result of multiplying together all members in *list*.
`ROUND(`*x*`,`*y*`)`	Returns the value of *x* rounded to the specified number of decimal places indicated by floor(*y*), where floor has the mathematical meaning. If *y* is negative, any fractional part is discarded and the integer part of the value is rounded to the corresponding power of 10.
`SIGN(`*x*`)`	Returns 1 if *x* is positive; returns 0 if *x* is zero; and returns –1 if *x* is negative.
`SUM(`*list*`)`	Returns the sum of the items in *list*.
`TRUE`	Returns 1.

17.16.3.5 Table cell references

Items in a WordprocessingML table are organized into rows and columns with the box formed by the intersection of a row and column being called a *cell*. Cells have names such as A1, A2, B1, B2, and so on, with the letter representing a column and the number representing a row. The cell at the top-left corner of each table is named A1. Column letters are not case-sensitive.

A *cell reference* shall be one of the following:

- The name of a cell.
- A comma-separated set of cell names.
- A cell range where a colon (:) is used to separate the first and last cells in a designated range of cells that has a contiguous rectangular shape. Specifying a row or column's name only as the first and last cell in a range, selects that whole row or column, regardless of the number of rows and columns the table has now or might have in the future.

An *expression* inside a table's cell can have operands that are references to other cells in that table.

[*Example*: Consider a table with three rows (1, 2, and 3) and two columns (A and B):

A1 + B1	Returns the sum of the contents of cells A1 and B1.
SUM(A1,B2,A3)	Returns the sum of the contents of the list of cells.
SUM(B1:B3)	Returns the sum of the contents of all cells between B1 and B3, inclusive.
SUM(B:B)	Returns the sum of the contents of all cells in column B (even if new rows are added later).
SUM(A1:B2)	Returns the sum of the contents of all (four) cells in the rectangular grid delimited by A1 and B2, inclusive.
SUM(1:1,2:2)	Returns the sum of the contents of all cells in rows 1 and 2.

end example]

When used in a table cell, the functions taking a *list* argument can have a single argument of ABOVE, BELOW, LEFT, or RIGHT, spelled in any case combination. Such lists designate, respectively, all the cells above, below, to the left of, or to the right of that cell. However, the designated range terminates if a cell with blank or non-numeric contents is reached, except that if the first cell is blank, it is treated as containing 0. [*Example*: Given the following table:

12	=COUNT(BELOW)	
	10	
2	20	=SUM(LEFT)
3	xxx	
=AVERAGE(ABOVE)	40	

AVERAGE(ABOVE) results in 2.5, the average of cells A4 and A3; COUNT(BELOW) results in 2, B2 and B3; and SUM(LEFT) results in 22, the sum of B3 and A3. *end example*]

An *expression* used outside a table or in a cell of one table can refer to cells in a second table by making a bookmark to that second table and qualifying cell names in that table by their table name using the form

```
(tableBookmarkName cellReference)
```

[*Example*: Given that Table1 is a bookmark for a 3x2 table, =SUM(Table1 A1:A3) book results in the sum of column A's cells. *end example*]

17.16.4 Field formatting

The result of a field has a *format*, either by default or because that field contains a *formatting-switch*. There are three kinds of field formatting: date and time (§17.16.4.1), numeric (§17.16.4.2), and general (§17.16.4.3).

17.16.4.1 Date and time formatting

date-and-time-formatting-switch=
  \@ *switch-argument ;*

A *date-and-time-formatting-switch* specifies the format of a date or time result. [*Note*: This switch is sometimes called a *picture* switch because it allows the use of symbols to represent the format of the field result. *end note*] If the result of a field is not a date or time, this switch has no effect.

If no *date-and-time-formatting-switch* is present, a date or time result is formatted in an implementation-defined manner.

A date and time *switch-argument* is made up of a series of *picture items*.

Date Formatting Picture Items	
Picture Item	Description
aaa	Formats the day of the week or month in an abbreviated form according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions: • If the lang element is ja-JP or ko-KR, display in the corresponding language. • For all other lang element values, display picture item as text. Multiple instances of the picture item create repeated content.
A	Formats the day of the month as a number without a leading zero for single-digit days in Japanese numerals. Multiple instances of the picture item create repeated content.
bb	Formats the year as a 2-digit number according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions: • If the lang element is zh-TW, zh-CN, zh-HK, zh-SG, or zh-MO, use the Gregorian year [ISO 8601] . • For all other lang element values, use Thai Buddhist Era year This picture item can modify the behaviour of other picture items. Multiple instances of the picture item create repeated content unless the pattern contains the “bbbb” picture item. Those portions are formatted following the description outlined for the “bbbb” picture item.
bbbb	Formats the year as a 4-digit number according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions: • If the lang element is ZH-TW, zh-CN, zh-HK, zh-SG, or zh-MO, use the Gregorian year [ISO 8601] • For all other lang element values, use Thai Buddhist Era year This picture item can modify the behaviour of other picture items. Multiple instances of the picture item create repeated content.
BB	Formats the year as a 2-digit number.

Date Formatting Picture Items	
	Defaults to the Gregorian [ISO 8601] calendar, but also responds to the \s and \h switches. Multiple instances of the picture item create repeated content as specified by the following: 3. Working from the text direction of the run, create as many groups as possible that contain “BBBB” in each group. Format each group using the description outlined for the “BBBB” picture item. 4. Repeat step 1 for groups of “BB” using the description for that picture item.
BBBB	Formats the year as a 4-digit number. Defaults to the Gregorian calendar [ISO 8601], but also responds to the \s and \h switches. Multiple instances of the picture item create repeated content as specified by the “BB” picture item description.
d	Formats the day of the week or day of the month as a number without a leading zero for single-digit days. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb or bbbb picture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “dddd” in each group. Format each group using the description outlined for the “dddd” picture item. 2. Repeat step 1 for groups of “ddd”, “dd”, and “d” using the respective description for each picture item group.
dd	Formats the day of the week or day of the month as a number with a leading zero0 for single-digit days. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb or bbbb picture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the “d” picture item description.
ddd	Formats the day of the week or month in its abbreviated form according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence

Date Formatting Picture Items	
	of the \s and \h switches, and the bb, bbbb, ปปปป, ปป, ดดดด, ดดด, ดด, ด, วววว, ววว, วว, and ว picture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the “d” picture item description.
dddd	Formats the day of the week as its full name according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb, bbbb, ปปปป, ปป, ดดดด, ดดด, ดด, ด, วววว, ววว, วว, and ว picture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the “d” picture item description.
D	Formats the day of the week or day of the month as a number without a leading zero for single-digit days. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb or bbbb picture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “DDDD” in each group. Format each group using the description outlined for the “DDDD” picture item. 2. Repeat step 1 for groups of “DDD”, “DD”, and “D” using the respective description for each picture item group.
DD	Formats the day of the month as a two-digit number (with a leading zero for single-digit days). Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb or bbbb picture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the “D” picture item description.
DDD	Formats the day of the week in an abbreviated form according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb or bbbb, ปปปป, ปป, ดดดด, ดดด, ดด, ด, วววว, ววว, วว, and ว picture item (to the Thai Buddhist Era calendar).

Date Formatting Picture Items	
	Multiple instances of the picture item create repeated content as specified by the “D” picture item description.
DDDD	Formats the day of the week as its full name according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb, bbbb, ปปปป, ปป, ดดดด, ดดด, ดด, ด, วววว, ววว, วว, and ว picture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the “D” picture item description.
e	Formats the Japanese Emperor Era era with no leading zero for single-digit years. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “ee” in each group. Format each group using the description outlined for the “ee” picture item. 2. Repeat step 1 for groups of “e” using the description for that picture item.
ee	Formats the Japanese Emperor Era era with a leading zero for single-digit years. Multiple instances of the picture item create repeated content as specified by the “e” picture item description.
E	Formats the era according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions: • If the lang element is ja-JP, display the Japanese Emperor Era era with no leading zero for single-digit years in the corresponding language. • If the lang element is zh-TW, display the Taiwanese year in the corresponding language. • For all other values, display the Gregorian year [ISO 8601] as a four-digit number using ja-JP. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “EE” in each group. Format each group using the description outlined for the “EE” picture item. 2. Repeat step 1 for groups of “E” using the description for that picture item.
EE	Formats the Gregorian year [ISO 8601] as a four-digit number, according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions.

Date Formatting Picture Items	
	• If the lang element is ja-JP, zh-TW, zh-CN, zh-HK, zh-SG, or zh-MO display in the corresponding language. • For all other lang element values, display in ja-JP. Multiple instances of the picture item create repeated content as specified by the “E” picture item description.
g	Formats the era according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions: If the lang element is ko-KR or zh-TW, display nothing. For all other lang element values, display Japanese Emperor Era era as its abbreviated form in en-US. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “ggg” in each group. Format each group using the description outlined for the “ggg” picture item. 2. From the remaining values, repeat step 1 for groups of “gg” using the description for that picture item. 3. From the remaining values, repeat step 1 for groups of “g” using the description for that picture item.
gg	Formats the era according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions: • If the lang element is ja-JP, display the Japanese Emperor Era era as its abbreviated form in the corresponding language. • If the lang element is ko-KR, display the Korean Tangun era as its full name in the corresponding language. • If the lang element is zh-TW, display the Taiwanese era as its full name in the corresponding language. • For all other lang element values, display the Japanese Emperor Era era as its abbreviated form in ja-JP. Multiple instances of the picture item create repeated content as specified by the “g” picture item description.
ggg	Formats the era according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions: • If the lang element is ja-JP, display the Japanese Emperor Era era as its full name in the corresponding language. • If the lang element is ko-KR, display the Korean Tangun era as its full name in the corresponding language. • If the lang element is zh-TW, display the Taiwanese era as its full name in the corresponding language. • For all other lang element values, display the Japanese Emperor Era era as its full name in ja-JP.

Date Formatting Picture Items	
	Multiple instances of the picture item create repeated content as specified by the "g" picture item description.
G	Formats the era according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions. • If the lang element is ko-KR or zh-TW, display nothing. • For all other lang element values, display the Japanese Emperor Era era as its abbreviated form in ja-JP. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain "GG" in each group. Format each group using the description outlined for the "GG" picture item. 2. Repeat step 1 for groups of "G" using the description for that picture item.
GG	Formats the era according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions. • If the lang element is ko-KR, display the Korean Tangun era as its full name in the corresponding language. • If the lang element is zh-TW, display the Taiwanese era as its full name in the corresponding language. • For all other lang element values, display the Japanese Emperor Era era as its full name in ja-JP. Multiple instances of the picture item create repeated content as specified by the "G" picture item description.
M	Formats the month as a number without a leading zero for single-digit months. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb or bbbb picture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain "MMMM" in each group. Format each group using the description outlined for the "MMMM" picture item. 2. Repeat step 1 for groups of "MMM", "MM", and "M" using the respective description for each picture item group.
MM	Formats the month as a number with a leading zero for single-digit months. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb or bbbb picture item (to the Thai Buddhist Era calendar).

Date Formatting Picture Items	
	Multiple instances of the picture item create repeated content as specified by the “M” picture item description.
MMM	Formats the month in its abbreviated form according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb, bbbb, ปปปป, ปป, ดดดด, ดดด, ดด, ด, วววว, ววว, วว, and ว picture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the “M” picture item description.
MMMM	Formats the month as its full name according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb, bbbb, ปปปป, ปป, ดดดด, ดดด, ดด, ด, วววว, ววว, วว, and ว picture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the “M” picture item description.
n	Formats the Japanese Emperor Era era with no leading zero for single-digit years. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “nn” in each group. Format each group using the description outlined for the “nn” picture item. 2. Repeat step 1 for groups of “n” using the description for that picture item.
nn	Formats the Japanese Emperor Era era with leading zero for single-digit years. Multiple instances of the picture item create repeated content as specified by the “n” picture item description.
O	Formats the month as a number without a leading zero for single-digit months in Japanese numerals. Multiple instances of the picture item create repeated content.
w	Formats the day of the week in an abbreviated form according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions: • If the lang element is ja-JP or ko-KR, display in the corresponding language. • For all other lang element values, display picture item as literal text.

Date Formatting Picture Items	
	Multiple instances of the picture item create repeated content.
W	Formats the day of the week in an abbreviated form according to the language specified by the lang element (§17.3.2.20) on the run containing the field instructions: - If the lang element is ja-JP, ko-KR, zh-TW, zh-CN, zh-HK, zh-SG, or zh-MO, display in the corresponding language. - For all other lang element values, display the picture item as literal text. Multiple instances of the picture item create repeated content.
y	Formats the year as a 2-digit number. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “yyyy” in each group. Format each group using the description outlined for the “yyyy” picture item. 2. Repeat step 1 for groups of “yy” and “y” using the respective description for each picture item group.
yy	Formats the year as a 2-digit number. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches. Multiple instances of the picture item create repeated content as specified by the “y” picture item description.
yyyy	Formats the year as a 4-digit number. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches. Multiple instances of the picture item create repeated content as specified by the “y” picture item description.
Y	Formats the year as a 2-digit number. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb, bbbb, ปปปป, and ปป picture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the following:

Date Formatting Picture Items	
	1. Working from the text direction of the run, create as many groups as possible that contain “YYYY” in each group. Format each group using the description outlined for the “YYYY” picture item. 2. Repeat step 1 for groups of “YY” and “Y” using the respective description for each picture item group.
YY	Formats the year as a 2-digit number. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb, bbbb, ปปปป, and ปปpicture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the “Y” picture item description.
YYYY	Formats the year as a 4-digit number. Defaults to the Gregorian calendar [ISO 8601], but also changes in the presence of the \s and \h switches, and the bb, bbbb, ปปปป, and ปปpicture item (to the Thai Buddhist Era calendar). Multiple instances of the picture item create repeated content as specified by the “Y” picture item description.
ว	Formats the day of the month as a number without a leading zero for single-digit days in Thai numerals. This picture item can modify the behaviour of other picture items. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “วววว” in each group. Format each group using the description outlined for the “วววว” picture item. 2. Repeat step 1 for groups of “ววว”, “วว” and “ว” using the respective description for each picture item group.
วว	Formats the day of the month as a two-digit number (with a leading zero for single-digit days) in Thai numerals. This picture item can modify the behaviour of other picture items. Multiple instances of the picture item create repeated content as specified by the “ว” picture item description.
ววว	Formats the Thai Buddhist Era day of the week in its abbreviated form in Thai. This picture item can modify the behaviour of other picture items.

©ISO/IEC 2012 – All rights reserved

Date Formatting Picture Items	
	Multiple instances of the picture item create repeated content as specified by the “ว” picture item description.
วววว	Formats the Thai Buddhist Era day of the week as its full name in Thai. This picture item can modify the behaviour of other picture items. Multiple instances of the picture item create repeated content as specified by the “ว” picture item description.
ด	Formats the Thai Buddhist Era month as a number without a leading zero for single-digit months in Thai numerals. This picture item can modify the behaviour of other picture items. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “ดดดด” in each group. Format each group using the description outlined for the “ดดดด” picture item. 2. Repeat step 1 for groups of “ดดด”, “ดด” and “ด” using the respective description for each picture item group.
ดด	Formats the Thai Buddhist Era month as a two-digit number (with a leading zero for single-digit months) in Thai numerals. This picture item can modify the behaviour of other picture items. Multiple instances of the picture item create repeated content as specified by the “ด” picture item description.
ดดด	Formats the Thai Buddhist Era month in its abbreviated form. This picture item can modify the behaviour of other picture items. Multiple instances of the picture item create repeated content as specified by the “ด” picture item description.
ดดดด	Formats the Thai Buddhist Era month as its full name. This picture item can modify the behaviour of other picture items. Multiple instances of the picture item create repeated content as specified by the “ด” picture item description.
ปป	Formats the Gregorian year as a 2-digit number using Thai numerals. This picture item can modify the behaviour of other picture items. Multiple instances of the picture item create repeated content as specified by

Date Formatting Picture Items	
	the following: 1. Working from the text direction of the run, create as many groups as possible that contain "ปปปป" in each group. Format each group using the description outlined for the "ปปปป" picture item. 2. Repeat step 1 for groups of "ปป" using the description for that picture item.
ปปปป	Formats the Gregorian year [ISO 8601] as a 4-digit number using Thai numerals. This picture item can modify the behaviour of other picture items. Multiple instances of the picture item create repeated content as specified by the "ปป" picture item description.

Time Formatting Picture Items	
Picture Item	Description
C	Formats the seconds as a number using Japanese numerals without a leading zero for single-digit seconds. Multiple instances of the picture item create repeated content.
hor H	Formats the hour on a 12-hr clock without a leading zero for single-digit hours. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain "hh" in each group. Format each group using the description outlined for the "hh" picture item. 2. Repeat step 1 for groups of "h" using the description for that picture item. Formats the hour without a leading 0 for single-digit hours.
hh	Formats the hour on a 12-hr clock with a leading zero for single-digit hours. Multiple instances of the picture item create repeated content as specified by the "h" picture item description.
hh or HH	Formats the hour with a leading 0 for single-digit hours.
H	Formats the hour on a 24-hr clock without a leading zero for single-digit hours. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain "HH" in each group. Format each group using the description outlined for the "HH" picture item. 2. Repeat step 1 for groups of "H" using the description for that picture

Time Formatting Picture Items	
	item.
HH	Formats the hour on a 24-hr clock with a leading zero for single-digit hours. Multiple instances of the picture item create repeated content as specified by the “H” picture item description.
I	Formats the minutes without a leading zero for single-digit days in Japanese numerals. Multiple instances of the picture item create repeated content.
m	Formats the minutes without a leading zero for single-digit minutes. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “mm” in each group. Format each group using the description outlined for the “mm” picture item. 2. Repeat step 1 for groups of “m” using the description for that picture item.
mm	Formats the minutes with a leading zero for single-digit minutes. Multiple instances of the picture item create repeated content as specified by the “m” picture item description.
r	Formats the hour on a 12-hr clock, without a leading zero for single-digit hours, according to the language specified by the lang element (§2.3.2.18) on the run containing the field instructions, as follows: • If the lang element is ja-JP, ko-KR, zh-TW, zh-CN, zh-HK, zh-SG, or zh-MO, display in the corresponding language. • For all other lang element values, display in zh-TW. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “rr” in each group. Format each group using the description outlined for the “rr” picture item. 2. Repeat step 1 for groups of “r” using the description for that picture item.
rr	Formats the hour on a 12-hr clock, without a leading zero for single-digit hours along with 12-hour clock indicators, according to the language specified by the lang element (§2.3.2.18) on the run containing the field instructions, as follows: • If the lang element is ja-JP, ko-KR, zh-TW, zh-CN, zh-HK, zh-SG, or zh-MO, display in the corresponding language.

Time Formatting Picture Items	
	• For all other lang element values, display in zh-TW. Multiple instances of the picture item create repeated content as specified by the “r” picture item description.
R	Formats the hour on a 24-hr clock, without a leading zero for single-digit hours, according to the language specified by the lang element (§2.3.2.18) on the run containing the field instructions, as follows: • If the lang element is ja-JP, ko-KR, zh-TW, zh-CN, zh-HK, zh-SG, or zh-MO, display in the corresponding language. • For all other lang element values, display in zh-TW. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “RR” in each group. Format each group using the description outlined for the “RR” picture item. 2. Repeat step 1 for groups of “R” using the description for that picture item.
RR	Formats the hour on a 12-hr clock as Arabic numerals, without a leading zero for single-digit hours and 12-hour clock indicators, according to the language specified by the lang element (§2.3.2.18) on the run containing the field instructions, as follows: • If the lang element is ja-JP, ko-KR, zh-TW, zh-CN, zh-HK, zh-SG, or zh-MO, display in the corresponding language. • For all other lang element values, display in zh-TW. Multiple instances of the picture item create repeated content as specified by the “R” picture item description.
s	Formats the seconds without a leading zero for single-digit seconds. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “ss” in each group. Format each group using the description outlined for the “ss” picture item. 2. Repeat step 1 for groups of “s” using the description for that picture item.
ss	Formats the seconds as a two-digit number (with a leading zero for single-digit seconds). Multiple instances of the picture item create repeated content as specified by the “s” picture item description.

Time Formatting Picture Items	
S	Formats the seconds without a leading zero for single-digit seconds. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “SS” in each group. Format each group using the description outlined for the “SS” picture item. 2. Repeat step 1 for groups of “S” using the description for that picture item.
SS	Formats the seconds as a two-digit number (with a leading zero for single-digit seconds). Multiple instances of the picture item create repeated content as specified by the “S” picture item description.
am/pm or AM/PM	Formats the uppercase 12-hour clock indicators according to the language specified by the lang element (§2.3.2.18) on the run containing the field instructions. Multiple instances of the picture item creates repeated content. Formats using an am/AM or pm/PM suffix.
AMPM or PMAM or JAM/JPM or JPM/JAM	Formats the uppercase or lowercase 12-hour clock indicators according to the language specified by the lang element (§2.3.2.18) on the run containing the field instructions. Multiple instances of the picture item create repeated content.
am/pm or pm/am or A/P or P/A or a/p or p/a	Formats the uppercase or lowercase 12-hour clock indicators according to the language specified by the lang element (§2.3.2.18) on the run containing the field instructions. Multiple instances of the picture item create repeated content.
ม	Formats the hour on a 24-hr clock without a leading zero for single-digit hours. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “มม” in each group. Format each group using the description outlined for the “มม” picture item. 2. Repeat step 1 for groups of “ม” using the description for that picture item.
มม	Formats the hour on a 24-hr clock as a two-digit number (with a leading zero for single-digit hours). Multiple instances of the picture item create repeated content as specified by the “ม” picture item description.

Time Formatting Picture Items	
นน	Formats the minutes with a leading zero for single-digit minutes. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “นน” in each group. Format each group using the description outlined for the “นน” picture item. 2. Repeat step 1 for groups of “น” using the description for that picture item.
น	Formats the minutes without a leading zero for single-digit minutes. Multiple instances of the picture item create repeated content as specified by the “นน” picture item description..
ทท	Formats the seconds with a leading zero for single-digit seconds. Multiple instances of the picture item create repeated content as specified by the following: 1. Working from the text direction of the run, create as many groups as possible that contain “ทท” in each group. Format each group using the description outlined for the “ทท” picture item. 2. Repeat step 1 for groups of “ท” using the description for that picture item.
ท	Formats the seconds without a leading zero for single-digit seconds. Multiple instances of the picture item create repeated content as specified by the “ทท” picture item description.

Miscellaneous Formatting Picture Items	
Picture Item	Description
Other character	Includes the specified character in the result at that position. [*Note*: Commonly used characters are colon (:), hyphen (-), asterisk (*), slash (/), and space. *end note*]
'*text*'	Includes *text* in the result.
`*numbered-item*`	Includes, in Arabic numerals, the number of the preceding item numbered as a caption or resulting from a SEQ field (§17.16.5.56). *numbered-item* shall be the same name as *identifier* in that SEQ field.

[*Example*: When updated in a US-English context on the date and time shown below, the following fields produced these results:

DATE \@ "M/d/yyyy"	1/3/2006
DATE \@ "dddd, MMMM dd, yyyy"	Tuesday, January 03, 2006
DATE \@ "MMMM d, yyyy"	January 3, 2006
DATE \@ "M/d/yy"	1/3/06
DATE \@ "yyyy-MM-dd"	2006-01-03
DATE \@ "d-MMM-yy"	3-Jan-06
DATE \@ "M.d.yyyy"	1.3.2006
DATE \@ "MMM. d, yy"	Jan. 3, 06
DATE \@ "d MMMM yyyy"	3 January 2006
DATE \@ "MMMM yy"	January 06
DATE \@ "MMM-yy"	Jan-06
DATE \@ "M/d/yyyy h:mm am/pm"	1/3/2006 5:28 PM
DATE \@ "M/d/yyyy h:mm:ss am/pm"	1/3/2006 5:28:34 PM
DATE \@ "h:mm am/pm"	5:28 PM
DATE \@ "h:mm:ss am/pm"	5:28:34 PM
DATE \@ "HH:mm"	17:28
DATE \@ "'Today is 'HH:mm:ss"	Today is 17:28:34

end example]

17.16.4.2 Numeric formatting

numeric-formatting-switch=
\#   *switch-argument* ;

A *numeric-formatting-switch* specifies the format of a numeric result. If the result of a field is not a number, this switch has no effect.

If no *numeric-formatting-switch* is present, a numeric result is formatted without leading spaces or trailing fractional zeros. If the result is negative, a leading minus sign is present. If the result is a whole number, no radix point is present.

A numeric *switch-argument* is made up of a series of *picture items*.

Numeric Formatting Picture Items	
Picture Item	Description
0	Specifies the requisite numeric positions to display in the result. If the result does not include a digit in that position, 0 is displayed. [*Example*: In a US-English context, =4+5 \# 00.00 displays "09.00". *end example*]
#	Specifies the requisite numeric positions to display in the result. If the result does not include a digit in that position, a space is displayed. Extra fractional digits are rounded off. [*Example*: =9+6 \# $### displays "$ 15". *end example*]
x	Drops digits to the left of the x placeholder. If the placeholder is to the right of the decimal point, the result is rounded to that place. [*Example*: In a US-English context, =111053+111439 \# x## displays "492", =1/8 \# 0.00x displays "0.125", and =3/4 \# .x displays ".8". *end example*]
.	Indicates the radix-point position. [*Example*: In a US-English context, =95.4 \# $###.00 displays "$ 95.40. *end example*] The radix-point character displayed is locale-specific.
,	Separates groups of three digits. [*Example*: In a US-English context, =2456800 \# $#,###,### displays "2,456,800". *end example*] The separator character displayed is locale-specific.
-	Prepends a minus sign to a negative result, or prepends a space if the result is positive or 0. [*Example*: =80-90 \# -## displays "-10", while =90-80 \# -## displays " 80". *end example*]
+	Prepends a plus sign to a positive result, a minus sign to a negative result, or a space if the result is 0. [*Example*: =90-80 \# +## displays "+10", and =80-90 \# +## displays "-10". *end example*]
Other character	Includes the specified character in the result at that position. [*Example*: =33 \# ##% displays "33%". *end example*]
'*text*'	Includes text in the result. [*Example*: In a US-English context, if Price is a bookmark for 26.5, =Price*15% \# "##0.00 'is the sales tax'" displays "$ 3.98 is the sales tax". *end example*]
`*numbered-item*`	Includes, in Arabic numerals, the number of the preceding item numbered as a caption or resulting from a SEQ field (§17.16.5.56). *numbered-item* shall be the same name as *identifier* in that SEQ field. [*Example*: =SUM(A1:D4) \# "##0.00 'is the total of Table' `table`" displays "456.34 is the total of Table 2". *end example*]
positive-result ; *negative-result*	Specifies different sets of picture items for positive and negative results. A zero value uses the positive picture. [*Example*: =Sales95 \# $#,##0.00;-$#,##0.00 displays that bookmark's positive values using $#,##0.00, and it's negative values using -$#,##0.00. *end example*]
positive-result ; *negative-result* ; *zero-result*	Specifies different sets of picture items for positive, negative, and zero results. [*Example*: =Sales95 \# $#,##0.00;-$#,##0.00;$0 displays that bookmark's positive values using $#,##0.00, it's negative values using -$#,##0.00, and its zero values using $0. *end example*]

©ISO/IEC 2012 – All rights reserved

17.16.4.3 General formatting

general-formatting-switch=
  * *switch-argument* ;

A *general-formatting-switch* specifies a variety of formats for a numeric or text result. If the result type of a field does not correspond to the format specified, this switch has no effect.

A *switch-argument* is made up of a series of *picture items*.

17.16.4.3.1 General formatting - Numeric Values

The following *switch-argument*s apply to fields whose field result is a numeric value. If the result type of the field is not numeric, then these switches have no effect. If the field result varies based on the language of the field instructions, those variations are noted inline:

General Formatting Switch Arguments	
Switch Argument	Description
`AIUEO`	Formats a numeric result using hiragana characters in the traditional a-i-u-e-o order. [*Example*: `1 \* AIUEO` results in ア. *end example*] Corresponds to an ST_NumberFormat enumeration value of `aiueoFullWidth`.
`ALPHABETIC`	Formats a numeric result as one or more occurrences of an uppercase alphabetic Latin character. Value 1 results in the letter A, value 2 results in the letter B, and so on up to value 26, which results in the letter Z. For values greater than 26, 26 is repeatedly subtracted from the value until the result is 26 or less. The result value determines which letter to use, and the same letter is repeated for each time 26 was subtracted from the original value. [*Example*: `=54 \* ALPHABETIC` results in "BBB" as subtracting 26 from 54 two times, producesthe value 2, which is represented by the letter B. *end example*] Corresponds to an ST_NumberFormat value of `upperLetter`.
`alphabetic`	Formats a numeric result as one or more occurrences of an lowercase alphabetic Latin character. Value 1 results in the letter a, value 2 results in the letter b, and so on up to value 26, which results in the letter z. For values greater than 26, 26 is repeatedly subtracted from the value until the result is 26 or less. The result value determines which letter to use, and the same letter is repeated for each time 26 was subtracted from the original value. [*Example*: `=52 \* alphabetic` results in "zz" as subtracting 26 from 52 one time, produces the value 26, which is represented by the letter z.. *end example*] Corresponds to an ST_NumberFormat enumeration value of `lowerLetter`.
`Arabic`	Formats a numeric result using Arabic cardinal numerals. [*Example*: For page 123, PAGE `\* Arabic` results in "123". *end example*] Corresponds to an ST_NumberFormat enumeration value of `decimal`.
`ARABICABJAD`	Formats a numeric result using ascending Abjad numerals. [*Example*: `12 \* ARABICABJAD` results in ل. *end example*] Corresponds to an ST_NumberFormat enumeration value of `arabicAbjad`.
`ARABICALPHA`	Formats a numeric result using characters in the Arabic alphabet. [*Example*:

General Formatting Switch Arguments	
	12 * ARABICABJAD results in س. *end example*] Corresponds to an ST_NumberFormat enumeration value of arabicAlpha.
ArabicDash	Formats a numeric result using Arabic cardinal numerals, with a prefix of "- " and a suffix of " -". [*Example*: For page 123, PAGE * ArabicDash results in "- 123 -". *end example*] Corresponds to an ST_NumberFormat enumeration value of numberInDash.
BAHTTEXT	Formats a numeric result in the following form: • If the value is an integer, it is displayed using the . Thai counting system, with บาทถ้วน appended to the result. • If the value includes a fractional value, the fractional part is rounded to two decimal places, and the resulting value is displayed in the form integer-part-in-Thai-counting format บาท fractional-part-in-Thai-counting format สตางค์. [*Example*: 1 * BAHTTEXT results in หนึ่งบาทถ้วน. *end example*] Corresponds to an ST_NumberFormat enumeration value of bahtText.
CardText	Formats a numeric result as lowercase cardinal text. [*Example*: For page 123, PAGE * CardText results in "one hundred twenty-three". *end example*] Corresponds to an ST_NumberFormat enumeration value of cardinalText.
CHINESENUM1	Formats a numeric result using ascending numbers from the appropriate counting system. [*Example*: 10 * CHINESENUM1 results in 十. *end example*] Corresponds to an ST_NumberFormat enumeration value of chineseCounting (zh-CN) or taiwaneseCounting (zn-TW).
CHINESENUM2	Formats a numeric result using sequential numbers from the appropriate legal format. [*Example*: 123 * CHINESENUM2 results in 壹佰貳拾參. *end example*] Corresponds to an ST_NumberFormat enumeration value of chineseLegalSimplified (zh-CN) or ideographLegalTraditional (zh-TW).
CHINESENUM3	Formats a numeric result using sequential numbers from the appropriate counting thousand system. [*Example*: 10 * CHINESENUM3 results in 一百二十三. *end example*] Corresponds to an ST_NumberFormat enumeration value of chineseCountingThousand (zh-CN) or taiwaneseCountingThousand (zh-TW).
CHOSUNG	Formats a numeric result using sequential numbers from the Korean Chosung format. [*Example*: 1 * CHOSUNG results in ㄱ. *end example*] Corresponds to an ST_NumberFormat enumeration value of chosung.
CIRCLENUM	Formats a numeric result using decimal numbering enclosed in a circle, using the enclosed alphanumeric glyph character for numbers in the range 1–20. For non-negative numbers outside this range, formats them as with ARABIC. [*Example*: 12 * CIRCLENUM results in ⑫. *end example*] Corresponds to an ST_NumberFormat enumeration value of decimalEnclosedCircle.
DBCHAR	Formats a numeric result using double-byte Arabic numbering. [*Example*: 123 * DBCHAR results in １２３. *end example*]

General Formatting Switch Arguments	
	Corresponds to an ST_NumberFormat enumeration value of `decimalFullWidth`.
DBNUM1	Formats a numeric result using sequential digital ideographs, using the appropriate character. [*Example*: `12 \* DBNUM1` results in 一二. *end example*] Corresponds to an ST_NumberFormat enumeration value of `ideographDigital` (ja-JP) or `koreanDigital` (ko-KR).
DBNUM2	Formats a numeric result using sequential numbers from the appropriatecounting system. [*Example*: `12 \* DBNUM2` results in 十二. *end example*] Corresponds to an ST_NumberFormat enumeration value of `japaneseCounting` (ja-JP) or `koreanCounting` (ko-KR).
DBNUM3	Formats a numeric result using sequential numbers from the appropriate legal counting system. [*Example*: `12 \* DBNUM3` results in 壱拾弐. *end example*] Corresponds to an ST_NumberFormat enumeration value of `japaneseLegal` (ja-JP) or `koreanLegal` (ko-KR).
DBNUM4	Formats a numeric result using sequential numbers from the appropriate digital counting system. [*Example*: `12 \* DBNUM4` results in 一二. *end example*] Corresponds to an ST_NumberFormat enumeration value of `japaneseDigitalTenThousand` (ja-JP) or `koreanDigital2` (ko-KR) or `taiwaneseDigital` (zh-TW).
`DollarText`	Formats a numeric result in the following form: *integer-part-as-cardinal-text* `and` *nn*`/100` The fractional part is rounded to two decimal places, *nn*, and is formatted using Arabic cardinal numerals. [*Example*: `=1234.567 \* DollarText` results in `"one thousand two hundred thirty-four and 57/100"`. *end example*] Corresponds to an ST_NumberFormat enumeration value of `dollarText.`
GANADA	Formats a numeric result using sequential numbers from the Korean Ganada format. [*Example*: `12 \* GANADA` results in 타. *end example*] Corresponds to an ST_NumberFormat enumeration value of `ganada`.
GB1	Formats a numeric result using decimal numbering followed by a period, using the enclosed alphanumeric glyph character. [*Example*: `12 \* GB1` results in ⒓. *end example*] Corresponds to an ST_NumberFormat enumeration value of `decimalEnclosedFullstop`.
GB2	Formats a numeric result using decimal numbering enclosed in parenthesis, using the enclosed alphanumeric glyph character. [*Example*: `12 \* GB2` results in ⑿. *end example*] Corresponds to an ST_NumberFormat enumeration value of `decimalEnclosedParen`.
GB3	Formats a numeric result using decimal numbering enclosed in a circle, using the enclosed alphanumeric glyph character. Once the specified sequence reaches 11, the numbers can be replaced with non-enclosed equivalents. [*Example*:

General Formatting Switch Arguments	
	12 * GB3 results in 12. *end example*] Corresponds to an ST_NumberFormat enumeration value of decimalEnclosedCircleChinese.
GB4	Formats a numeric result using decimal numbering enclosed in a circle, using the enclosed alphanumeric glyph character. Once the specified sequence reaches 11, the numbers can be replaced with non-enclosed equivalents. [*Example*: 12 * GB4 results in 12. *end example*] Corresponds to an ST_NumberFormat enumeration value of ideographEnclosedCircle.
HEBREW1	Formats a numeric result using Hebrew numerals. [*Example*: 123 * HEBREW1 results in קכג. *end example*] Corresponds to an ST_NumberFormat enumeration value of hebrew1.
HEBREW2	Formats a numeric result using the Hebrew alphabet. [*Example*: 123 * HEBREW2 results in תתתתתמ. *end example*] Corresponds to an ST_NumberFormat enumeration value of hebrew2.
Hex	Formats the numeric result using uppercase hexadecimal digits. [*Example*: For page 355, PAGE * Hex results in "FF". *end example*] Corresponds to an ST_NumberFormat enumeration value of hex.
HINDIARABIC	Formats a numeric result using Hindi numbers. [*Example*: 123 * HINDIARABIC results in १२३. *end example*] Corresponds to an ST_NumberFormat enumeration value of hindiNumbers.
HINDICARDTEXT	Formats a numeric result using sequential numbers from the Hindi counting system. [*Example*: 123 * HINDICARDTEXT results in एक सौ तेईस. *end example*] Corresponds to an ST_NumberFormat enumeration value of hindiCounting.
HINDILETTER1	Formats a numeric result using Hindi vowels. [*Example*: 123 * HINDILETTER1 results in ठठठठ. *end example*] Corresponds to an ST_NumberFormat enumeration value of hindiVowels.
HINDILETTER2	Formats a numeric result using Hindi consonants. [*Example*: 123 * HINDILETTER2 results in ओओओओओओओ. *end example*] Corresponds to an ST_NumberFormat enumeration value of hindiConsonants.
IROHA	Formats a numeric result using the Japanese iroha. [*Example*: 12 * IROHA results in オ. *end example*] Corresponds to an ST_NumberFormat enumeration value of irohaFullWidth.
KANJINUM1	Formats a numeric result using a Japanese style using the appropriate counting system. [*Example*: 12 * KANJINUM1 results in 一二. *end example*] Corresponds to an ST_NumberFormat enumeration value of koreanDigital (ko-KR), ideographDigital (ja-JP), chineseCounting (zh-CN), or taiwaneseCounting (zh-TW).
KANJINUM2	Formats a numeric result using the appropriatecounting system. [*Example*:

©ISO/IEC 2012 – All rights reserved

General Formatting Switch Arguments	
	12 * KANJINUM2 results in 十二. *end example*] Corresponds to an ST_NumberFormat enumeration value of koreanCounting (ko-KR), chineseCountingThousand (ja-JP), chineseLegalSimplified (zh-CN), or ideographLegalTraditional (zh-TW).
KANJINUM3	Formats a numeric result using the appropriatecounting system. [*Example*: 12 * KANJINUM3 results in 壱拾弐. *end example*] Corresponds to an ST_NumberFormat enumeration value of koreanLegal (ko-KR) or japaneseLegal (ja-JP) or chineseCountingThousand (zh-CN) or taiwaneseCountingThousand (zh-TW).
Ordinal	Formats a numeric result using lowercase ordinal Arabic numerals. [*Example*: =32 * Ordinal results in "32nd". *end example*] Corresponds to an ST_NumberFormat enumeration value of ordinal.
OrdText	Formats a numeric result as lowercase ordinal text. Apart from being used to round off the whole number part, the fractional part is not used. [*Example*: =1234.567 * OrdText results in "one thousand two hundred thirty-fifth". *end example*] Corresponds to an ST_NumberFormat enumeration value of ordinalText.
Roman	Formats a numeric result using uppercase Roman numerals. [*Example*: For page 123, PAGE * Roman results in "CXXIII". *end example*] Corresponds to an ST_NumberFormat enumeration value of upperRoman.
roman	Formats a numeric result using lowercase Roman numerals. [*Example*: For page 123, PAGE * roman results in "cxxiii". *end example*] Corresponds to an ST_NumberFormat enumeration value of lowerRoman.
SBCHAR	Formats a numeric result using single-byte Arabic numbering. [*Example*: 123 * SBCHAR results in 123. *end example*] Corresponds to an ST_NumberFormat enumeration value of decimalHalfWidth.
THAIARABIC	Formats a numeric result using Thai numbers. [*Example*: 123 * THAIARABIC results in ๑๒๓. *end example*] Corresponds to an ST_NumberFormat enumeration value of thaiNumbers.
THAICARDTEXT	Formats a numeric result using sequential numbers from the Thai counting system. [*Example*: 123 * THAICARDTEXT results in หนึ่งร้อยยี่สิบสาม. *end example*] Corresponds to an ST_NumberFormat enumeration value of thaiCounting.
THAILETTER	Formats a numeric result using Thai letters. [*Example*: 30 * THAILETTER results in ฮฮฮม. *end example*] Corresponds to an ST_NumberFormat enumeration value of thaiLetters.
VIETCARDTEXT	Formats a numeric result using Vietnamese numerals. [*Example*: 12 * VIETCARDTEXT results in mười hai. *end example*] Corresponds to an ST_NumberFormat enumeration value of vietnameseCounting.
ZODIAC1	Formats a numeric result using sequential numerical traditional ideographs.

General Formatting Switch Arguments	
	[*Example*: 1 * ZODIAC1 results in 甲. *end example*] Corresponds to an ST_NumberFormat enumeration value of ideographTraditional.
ZODIAC2	Formats a numeric result using sequential zodiac ideographs. [*Example*: 1 * ZODIAC2 results in 子. *end example*] Corresponds to an ST_NumberFormat enumeration value of ideographZodiac.
ZODIAC3	Formats a numeric result using sequential traditional zodiac ideographs. [*Example*: 1 * ZODIAC3 results in 甲子. *end example*] Corresponds to an ST_NumberFormat enumeration value of ideographZodiacTraditional.

17.16.4.3.2 General formatting - String Values

The following *switch-argument*s apply to fields whose field result is a string value:

General Formatting Switch Arguments	
Switch Argument	Description
Caps	Capitalizes the first letter of each word. [*Example*: USERNAME "mary smith" * Caps results in "Mary Smith", whereas USERNAME "marysmith" * Caps results in "Marysmith". *end example*]
FirstCap	Capitalizes the first letter of the first word. [*Example*: USERNAME "mary smith" * FirstCap results in "Mary smith". *end example*]
Lower	All letters are lowercase. [*Example*: USERNAME "Mary Smith" * Lower results in "mary smith". *end example*]
Upper	All letters are uppercase. [*Example*: USERNAME "Mary Smith" * Upper results in "MARY SMITH". *end example*]

17.16.4.3.3 General formatting - Field Result Formatting

The following *switch-argument*s apply to any field result, and provide directions to applications regarding the formatting which should be applied to a field result after a field update has been performed. As discussed in §17.16, as to when a field update is performed is outside the scope of ISO/IEC 29500.

The general formatting switch argument CHARFORMAT is an instruction that specifies formatting intended for application to the field result after a field update. If this switch is present, the formatting of the run containing the first instrText element after the fldChar element with a fldCharType attribute value of begin is applied to all runs in the field result whenever a new field result is generated. [*Example*: In a US-English context, on January 4, 2006, the field DATE * CHARFORMAT results in "1/4/2006". However, if the first run within the field instructions (containing the D in DATE) is bold, the field **D**ATE * CHARFORMAT results in "**1/4/2006**". If that D is italic, the field *D*ATE * CHARFORMAT results in "*1/4/2006*". If that D is bold, underlined, and red, the field **D**ATE * CHARFORMAT results in "**1/4/2006**".

The XML for the bold, underlined, red case is as follows:

```
<w:r>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r>
  <w:instrText xml:space="preserve"> </w:instrText>
</w:r>
<w:r …>
  <w:rPr>
    <w:b/>
    <w:color w:val="ED1C24"/>
    <w:u w:val="single"/>
  </w:rPr>
  <w:instrText>D</w:instrText>
</w:r>
<w:r>
  <w:instrText xml:space="preserve">ATE </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r …>
  <w:t>1/4/2006</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

Then use of the CHARFORMAT switch would cause the new field result to reuse the formatting on the first run in the field instructions, like this:

```
<w:r>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r>
  <w:instrText xml:space="preserve"> </w:instrText>
</w:r>
```

```
<w:r …>
  <w:rPr>
    <w:b/>
    <w:color w:val="ED1C24"/>
    <w:u w:val="single"/>
  </w:rPr>
  <w:instrText>D</w:instrText>
</w:r>
<w:r>
  <w:instrText xml:space="preserve">ATE /* CHARFORMAT</w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r …>
  <w:rPr>
    <w:b/>
    <w:color w:val="ED1C24"/>
    <w:u w:val="single"/>
  </w:rPr>
  <w:t>1/4/2006</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

end example]

If a format specified directly in the first run of a field's *field-type* name conflicts with a general formatting switch, the general formatting switch is ignored. [*Example*: If the first run is set in small caps and the switch `\* Lower` is also used, that switch is ignored. *end example*]

The general formatting switch argument `MERGEFORMAT` is used to specify an instruction regarding the formatting which is applied to the field result after a field update. If this switch is present, it specifies that applications that perform field updates should preserve the formatting of the existing field result when a new field result is generated, by applying the following logic:

- Delete all text from the current field result (leaving the paragraph and run structure intact).
- Insert the new field result text into the existing run/paragraph structure.
  - If the new text does not fill the existing structure, delete the superfluous runs/paragraphs.
  - If the new text overflows the existing structure, add additional runs/paragraphs as needed.

[*Example*: Consider the following field:

```
TIME \@ "HH:mm:ss" \* MERGEFORMAT
```

When it is updated, the result might be `12:22:27`, for example. If the seconds part of the displayed field result was underlined by the use of direct formatting within the original WordprocessingML, as in `12:22:27`, when that field is next updated, the seconds underlining is preserved.

If the original XML generated for this field is:

```
<w:r>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r>
  <w:instrText xml:space="preserve"> TIME \@ "HH:mm:ss" \* MERGEFORMAT
</w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r …>
  <w:t>17:02:</w:t>
</w:r>
<w:r …>
  <w:rPr>
    <w:u w:val="single"/>
  </w:rPr>
  <w:t>32</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

then use of the MERGEFORMAT switch would cause the new field result to reuse that structure, like this:

```
<w:r>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r>
  <w:instrText>TIME \@ "HH:mm:ss" \* MERGEFORMAT</w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r …>
  <w:t>12:22:</w:t>
</w:r>
```

```
<w:r …>
  <w:rPr>
    <w:u w:val="single"/>
  </w:rPr>
  <w:t>27</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

and omitting it would direct an application that it could replace it with a single new run, like this:

```
<w:r>
  <w:fldChar w:fldCharType="begin"/>
</w:r>
<w:r>
  <w:instrText>TIME \@ "HH:mm:ss"</w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r>
  <w:t>12:22:27</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

end example]

17.16.5 Field definitions

Each of the subclauses below this subclause describes a separate field, and each description contains a section marked **Syntax**. That section contains pieces of the field grammar as they pertain to that specific field. These pieces are presented in a slightly simpler form to aid in the understanding of the description. In those sections, the left-square-bracket and right-square-bracket designate the start and end of an option, as used in ISO/IEC 14977. However, the field-name, the open-parenthesis and the close-parenthesis designate actual literal text, as does each comma. [*Note*: Therefore, in a strict presentation according to ISO/IEC 14977, each field-name, open-parenthesis and close-parenthesis would appear with double-quotes surrounding each instance. *end note*]

The set of fields is divided into the following functional categories:

Category	Description	Fields
Date and Time	Inserts the current date and/or time, or date and/or	CREATEDATE (§17.16.5.11), DATE (§17.16.5.13), EDITTIME (§17.16.5.16),

Category	Description	Fields
	time of some kind of event.	PRINTDATE (§17.16.5.47), SAVEDATE (§17.16.5.53), TIME (§17.16.5.65)
Document Automation	Provides functionality for automated document processing.	COMPARE (§17.16.5.10), DOCVARIABLE (§17.16.5.15), GOTOBUTTON (§17.16.5.23), IF (§17.16.5.26), MACROBUTTON (§17.16.5.34), PRINT (§17.16.5.46)
Document Information	Inserts or stores information about the document.	AUTHOR (§17.16.5.4), COMMENTS (§17.16.5.9), DOCPROPERTY (§17.16.5.14), FILENAME (§17.16.5.17), FILESIZE (§17.16.5.18), KEYWORDS (§17.16.5.30), LASTSAVEDBY (§17.16.5.31), NUMCHARS (§17.16.5.41), NUMPAGES (§17.16.5.42), NUMWORDS (§17.16.5.43), SUBJECT (§17.16.5.60), TEMPLATE (§17.16.5.64), TITLE (§17.16.5.66)
Equations and Formulas	Defines formulas and calculates results; inserts symbols.	= *formula* (§17.16.3), ADVANCE (§17.16.5.2), SYMBOL (§17.16.5.61)
Form Fields	Defines fields that support insertion of data through form controls.	FORMCHECKBOX (§17.16.5.20), FORMDROPDOWN (§17.16.5.21), FORMTEXT (§17.16.5.22)
Index and Tables	Defines entries for, and builds, a table of contents, table of figures, or table of authorities.	INDEX (§17.16.5.29), RD (§17.16.5.50), TA (§17.16.5.62), TC (§17.16.5.63), TOA (§17.16.5.67), TOC (§17.16.5.68), XE (§17.16.5.72)
Links and References	Inserts information from another place in the same document or from a different document or file.	AUTOTEXT (§17.16.5.5), AUTOTEXTLIST (§17.16.5.6), BIBLIOGRAPHY (§17.16.5.7), CITATION (§17.16.5.8) HYPERLINK (§17.16.5.25), INCLUDEPICTURE (§17.16.5.27), INCLUDETEXT (§17.16.5.28), LINK (§17.16.5.32), NOTEREF (§17.16.5.40), PAGEREF (§17.16.5.45), QUOTE (§17.16.5.49), REF (§17.16.5.51), STYLEREF (§17.16.5.59)
Mail Merge	Defines information that is to be used in a mail merge. A *mail merge* is a process by which a data set (e.g., of names and addresses) is combined with a WordprocessingML document to produce a customized document for	ADDRESSBLOCK (§17.16.5.1), ASK (§17.16.5.3), COMPARE (§17.16.5.10), DATABASE (§17.16.5.12), FILLIN (§17.16.5.19), GREETINGLINE (§17.16.5.24), IF (§17.16.5.26), MERGEFIELD (§17.16.5.35), MERGEREC (§17.16.5.36), MERGESEQ (§17.16.5.37), NEXT (§17.16.5.38), NEXTIF (§17.16.5.39), SET (§17.16.5.57), SKIPIF (§17.16.5.58)

Category	Description	Fields
	each record in said data set. In other words, a *mail merge* is an operation by which an application replaces certain fields with the data in each record from a corresponding data source (see §17.14 for additional information).	
Numbering	Specifies numbering for document items such as sections and pages; also bar codes.	LISTNUM (§17.16.5.33), PAGE (§17.16.5.44), REVNUM (§17.16.5.52), SECTION (§17.16.5.54), SECTIONPAGES (§17.16.5.55), SEQ (§17.16.5.56)
User Information	Referencesthe name, initials, or address of a user account under which the document is manipulated. [*Note*: These fields can be used in documents to allow applications to perform implementation-defined updates under a particular user's context (if such a context exists); for example, to add the address of the current user to a generic form letter. *end note*]	USERADDRESS (§17.16.5.69), USERINITIALS (§17.16.5.70), USERNAME (§17.16.5.71)

17.16.5.1 ADDRESSBLOCK

Syntax:

ADDRESSBLOCK [*switches*]

Description: Represents an address block. An *address block* is a block of text specifying information appropriate for a postal mailing address, in the order required by the destination country. [*Example*: An address block for the Canadian postal system might consist of:

```
<<RECIPIENT NAME>>
<<POSTAL ADDRESS>>
<<CITY>>, <<PROVINCE>>, Canada
<<POSTAL CODE>>
```

end example]

Field Value: The address block.

Switches: Zero or more of the following *field-specific-switches*.

<table>
<tr><td>\c <em>field-argument</em></td><td><em>text</em> in this switch's <em>field-argument</em> specifies whether to include the name of the country/region: a value of 0 causes the country/region to be omitted; a value of 1 causes it to be included (regardless of any value given for \e), and a value of 2 causes country/region to be included, but only if it is different from any and all specifiedvalues for \e.</td></tr>
<tr><td>\d</td><td>Specifies that the address is to be formatted according to the country/region of the recipient as defined by POST*CODE (Universal Postal Union 2006). If this switch is not used, then addresses are formatted according to the ordering defined by the \f switch. If that switch is also omitted, addresses are formatted according to some implementation-specific preference.</td></tr>
<tr><td>\e <em>field-argument</em></td><td><em>text</em> in this switch's <em>field-argument</em> specifies acountry/region to exclude from the address block. [<em>Note</em>: This is useful when your mailing contains a mix of domestic and international recipients. <em>end note</em>] To exclude more than one country or region, use a \e switch for each one, as multiple \e switches are permitted.</td></tr>
<tr><td>\f <em>field-argument</em></td><td><em>text</em> in this switch's <em>field-argument</em> specifies the name and address format by providing a template of merge-field placeholders in the following format: "<<_" + MAPPED NAME + "_>>" (< = U+003C; > = U+003E; _ = U+005F), where MAPPED NAME corresponds to a value specified by the mappedName element (§17.14.23).

The following possible values of MAPPED NAME and corresponding mapped merge field names are reserved:

<table>
<tr><th>MAPPED NAME</th><th>Merge Field Name</th></tr>
<tr><td>TITLE0</td><td>Title</td></tr>
<tr><td>FIRST0</td><td>First Name</td></tr>
<tr><td>LAST0</td><td>Last Name</td></tr>
<tr><td>SUFFIX0</td><td>Suffix</td></tr>
<tr><td>COMPANY</td><td>Company</td></tr>
<tr><td>STREET1</td><td>Address 1</td></tr>
<tr><td>STREET2</td><td>Address 2</td></tr>
<tr><td>CITY</td><td>City</td></tr>
<tr><td>STATE</td><td>State</td></tr>
<tr><td>POSTAL</td><td>Postal Code</td></tr>
<tr><td>COUNTRY</td><td>Country or Region</td></tr>
</table>

[<em>Example</em>: Using these values, a U.S. postal address might be specified as follows:

\f "<<_TITLE0_ >><<_FIRST0_>><< _LAST0_>><< _SUFFIX0_>>
<<_COMPANY_>></td></tr>
</table>

	<<_STREET1_>> <<_STREET2_>> <<_CITY_>><<, _STATE_>><< _POSTAL_>><<_COUNTRY_>>" *end example*]
\l *field-argument*	*text* in this switch's *field-argument* specifies the language ID used to format the address. The default is to use the language ID of the first character of the document. This language ID is specified in the format defined by ST_Lang (§22.9.2.6)

17.16.5.2 ADVANCE

Syntax:

ADVANCE [*switches*]

Description: When text within a document is rendered, typically, each character is displayed immediately following the previous piece of content (another character, an image, and so on), according to the lexical order of the underlying WordprocessingML. The presence of this field moves the starting point at which the text that lexically follows the field is displayed to the right or left, up or down, or to a specific horizontal or vertical position. It does not affect the display of other run content (e.g., images). The switches used by this field can cause text to overlap when it is displayed. If the text is displayed within the constraints of a fixed page, repositioned text is not display if it is moved beyond the print margins of the current page (even if the target location would logically appear on a previous/following page).

Field Value: None.

Switches: Zero or more of the following *field-specific-switches*.

\d *field-argument*	Moves the text that follows the field down by the integral number of points specified by *text* in this switch's *field-argument*.
\l *field-argument*	Moves the text that follows the field left by the integral number of points specified by *text* in this switch's *field-argument*.
\r *field-argument*	Moves the text that follows the field right by the integral number of points specified by *text* in this switch's *field-argument*.
\u *field-argument*	Moves the text that follows the field up by the integral number of points specified by *text* in this switch's *field-argument*.
\x *field-argument*	Moves the text that follows the field the integral number of points specified by *text* in this switch's *field-argument* from the left edge of the column, frame, or text box.
\y *field-argument*	Moves the text that follows the field the integral number of points specified by *text* in this switch's *field-argument*. This shift is the vertical position relative to the page. The entire line of text that contains the field is moved. This switch is ignored if it specifies a location outside the page margins or if the switch is used inside any of the following: table, text box, footnote, endnote, annotation, header, or footer.

[*Example*: When the following WordprocessingML is displayed:

```
<w:r>
  <w:t>XX</w:t>
</w:r>
<w:fldSimple w:instrText="ADVANCE \u 6"/>
<w:r>
  <w:t>XX</w:t>
</w:r>
<w:fldSimple w:instrText="ADVANCE \d 12"/>
<w:r>
  <w:t>XX</w:t>
</w:r>
<w:fldSimple w:instrText="ADVANCE \l 20"/>
<w:r>
  <w:t>+</w:t>
</w:r>
<w:fldSimple w:instrText="ADVANCE \x 150"/>
<w:r>
  <w:t>ZZ</w:t>
</w:r>
```

The results are:

```
  XX
XX
    X+X                ZZ
```

end example]

17.16.5.3 ASK

Syntax:

ASK *field-argument-1* *field-argument-2* [*switches*]

field-argument-1:
field-argument

field-argument-2:
field-argument

Description: Prompts the user to enter information and assigns the bookmark designated by *field-argument-1* to represent the user's response. *text* in *field-argument-2* specifies the prompt text, which is displayed in a dialog box. The prompt is displayed each time the ASK field is updated. A response remains assigned to the bookmark until a new response is entered.Represents information needed to elicit and store a response (where a response is any string of characters, including the empty string). When a field update is performed, the text specified by *field-argument-2* is intended for use when requesting the response, and the resulting response is stored within the bookmark (§17.18.29) whose name attribute matches the value specified by *field-argument-1*.

Field Value: None.

Switches: Zero or more of the following *field-specific-switches*.

`\d` *field-argument*	The *text* in this switch's *field-argument* specifies a default response if one is not entered. If no default response is specified, the *field* result remains unchanged. To specify a blank entry as the default, *field-argument* shall be `""`.
`\o`	When used in a mail merge main document, this causes the display of the prompt once instead of each time a new data record is merged. The same response is inserted in each resulting merged document.

[*Example*: When the following fields are updated and "John" is entered as the response,

```
ASK AskResponse "What is your first name?"
Hello REF AskResponse.
```

the result is

```
Hello John.
```

end example]

17.16.5.4 AUTHOR

Syntax:

AUTHOR [*field-argument*] [*switches*]

Description: Retrieves, and optionally sets, the document author's name, as recorded in the Creator element of the Core File Properties part or, if *field-argument* is present, the name specified by *text* in *field-argument*. Specifying a *field-argument* shall change Creator to *text*.

Field Value: The document author's name.

Switches: One of the following *general-formatting-switches*: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Consider the case in which the Creator element is as follows:

```
<Creator>William Jones</Creator>
```

and the following field is updated:

```
AUTHOR
```

The result is:

```
William Jones
```

Updating the following field:

```
AUTHOR "Tony Caruso"
```

causes the Creator element to take on the specified value. *end example*]

17.16.5.5 AUTOTEXT

Syntax:

AUTOTEXT *field-argument*

Description: References theAutoText entry (§**17.12.5**) that meets the following conditions:

- The val attribute of the gallery element (§**17.12.9**) has a value of `autoTxt`.
- The val attribute of the name element (§**17.12.13**) has the value specified by *text* in *field-argument*.

When a field update is performed, the field result is the WordprocessingML contents of the AutoText entry.

Field Value: The WordprocessingML content of the specified AutoText entry.

Switches: None.

[*Example*: Assuming the following entries are defined with values of current page number, salutation, and a notice:

```
AUTOTEXT "- PAGE -"
AUTOTEXT "Yours truly,"
AUTOTEXT Confidential
```

when evaluated, they might produce the following results.

```
 - 13 -
Yours truly,
CONFIDENTIAL
```

end example]

17.16.5.6 AUTOTEXTLIST

Syntax:

AUTOTEXTLIST *field-argument* [*switches*]

Description: Creates a shortcut menu based on AutoText entries in the active template. The list can vary based on the styles applied to the AutoText entries. *text* in *field-argument* is inserted into the document.

A complex field XML implementation shall be used, and the *field-argument* text shall be placed in one or more runs between the separate and end parts of the fldChar element.

Field Value: *text* in *field-argument*.

Switches: Zero or more of the following *field-specific-switches*.

`\s` *field-argument*	Specifies that the list is to contain entries based on the style specified by *text* in this switch's *field-argument*. Without this switch, entries of the current paragraph style appear. If there are no entries for the current style, all entries appear. The style can be a paragraph style or a character style.
`\t` *field-argument*	*text* in this switch's *field-argument* specifies the text to show in the ScreenTip.

[*Example*: The field:

```
{ AUTOTEXTLIST "List of salutations" \s Salutation \t "Choose a salutation" }
```

causes the following to be displayed: In the document, the Salutation list; in the ScreenTip, `Choose a salutation`, and on the shortcut menu, the list of entries whose style is `Salutation`.

end example]

17.16.5.7 BIBLIOGRAPHY

Syntax:

`BIBLIOGRAPHY` [*switches*]

Description: Retrieves and displays the contents of the document's Bibliography part in the bibliographic style specified within the SelectedStyle attribute of the Sources (§22.6.2.60) element of the Bibliography part.

Field Value: The formatted bibliographic data for all sources in the current document.

Switches: Zero of more of the following *field-specific-switches*.

`\l` *field-argument*	The *text* in this switch's *field-argument* specifies the language ID that shall be used in to format the bibliographic sources in the document that do not specify a locale using the LCID (§22.6.2.39) element. This language ID is specified in the format defined by ST_Lang (§22.9.2.6)
`\f` *field-argument*	The *text* in this switch's *field-argument* specifies the language ID that shall be used to filter the bibliographic data to only the sources in the document that use that language. This language ID is specified in the format defined by ST_Lang (§22.9.2.6)

©ISO/IEC 2012 – All rights reserved

`\m` *field-argument*	The *text* in this switch's *field-argument* specifies that only the source with a Tag (§22.6.2.65) element value matching *field-argument* shall be displayed in the bibliography.

[*Example:* Consider a document with bibliographic data for ISO/IEC 29500:

Author: Ecma International
Title: Office Open XML Document Interchange Specification
Year: 2006
Month: October

the following field is updated:

```
BIBLIOGRAPHY /l 1033
```

The result for MLA is:

```
Ecma International. Office Open XML Document Interchange Specification. October
2006.
```

And for APA:

```
Ecma International. (2006, October). Office Open XML Document Interchange
Specification.
```

end example]

17.16.5.8 CITATION

Syntax:

`CITATION` *field-argument* [*switches*]

Description: Displays the contents of the Source (§22.6.2.59) element with a Tag (§22.6.2.65) element value matching *field-argument* using the bibliographic style specified within the SelectedStyle attribute of the Sources (§22.6.2.60) element of the Bibliography part.

Field Value: The comments relating to the current document.

Switches: Zero or more of the following *field-specific-switches*.

`\l` *field-argument*	The *text* in this switch's *field-argument* specifies the language ID which shall be used in conjunction with the specified bibliographic style to format the citation in the document. This language ID is specified in the format defined by ST_Lang (§22.9.2.6)
`\f` *field-argument*	The *text* in this switch's *field-argument* specifies the prefix which shall be prepended to the citation.

`\s` *field-argument*	The *text* in this switch's *field-argument* specifies the suffix which shall be appended to the citation.
`\p` *field-argument*	The *text* in this switch's *field-argument* specifies the page number associated with the citation.
`\v` *field-argument*	The *text* in this switch's *field-argument* specifies the volume number associated with the citation.
`\n`	Specifies that the author information shall be suppressed from the citation.
`\t`	Specifies that the title information shall be suppressed from the citation.
`\y`	Specifies that the year information shall be suppressed from the citation.
`\m` *field-argument*	The *text* in this switch's *field-argument* specifies the Tag (§22.6.2.65) element value for another source to be included in this citation's field result.

[*Example:* Consider a case with bibliographic data for ISO/IEC 29500:

Tag: Ecma01
Author: Ecma International
Title: Office Open XML Document Interchange Specification
Year: 2006
Month: October

the following field is updated:

```
CITATION Ecma01 /l 1033
```

The result for MLA is:

```
(Ecma International)
```

And for APA:

```
(Ecma International, 2006)
```

end example]

17.16.5.9 COMMENTS

Syntax:

COMMENTS [*field-argument*] [*switches*]

Description: Retrieves, and optionally sets, the comments relating to the current document, as recorded in the Description element of the Core File Properties part or, if *field-argument* is present, the comments specified by *text* in *field-argument*. Specifying a field-argument shall change Description to *text*.

Field Value: The comments relating to the current document.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Consider the case in which the Description element is as follows:

```
<Description>Once upon a time, in a land far, far away …</Description>
```

and the following field is updated:

COMMENTS

The result is:

Once upon a time, in a land far, far away …

Updating the following field:

COMMENTS "I came, I saw, I was not impressed."

causes the Description element to take on the specified value. *end example*]

17.16.5.10 COMPARE

Syntax:

COMPARE *comparison* [*switches*]

Description: Compares the values designated by the two *expression*s in *comparison* using the operator designated by *comparison-operator*. [*Note*: This field can be used to create compound logical comparisons with AND and OR functions in a formula, and then by using the result of the formula in an `IF` field. *end note*]

[*Note*: *comparison-operator* can be any one of the six relational and equality operators specified for *operator* (§17.16.3.3). *end note*]

If *comparison-operator* is = or <>, the left-hand *expression* operand can contain a question mark (?) to represent any single character, or an asterisk (*) to represent any string of characters. The expression shall be enclosed in quotation marks so that it is compared as a character string.

Field Value: 1 if the comparison is true, or 0 if the comparison is false.

Switches: None.

[*Example*: Consider the case in which the `IF` field in the following example is inserted into a mail merge main document. The `COMPARE` fields examine the data fields `CustomerNumber` and `CustomerRating` as each data record is merged. The OR function of the formula returns the value 1 if at least one of the data fields indicates poor credit, in which case the first text in quotation marks is printed:

```
{ IF { = OR ( { COMPARE { MERGEFIELD CustomerNumber } >= 4 },
  { COMPARE { MERGEFIELD CustomerRating } <= 9 } ) } = 1
  "Credit not acceptable" "Credit acceptable"}
```

The following COMPARE field results in the value 1 if any value in the PostalCode data field is the range 98500–98599:

```
{ COMPARE "{ MERGEFIELD PostalCode }" = "985*" }
```

end example]

17.16.5.11 CREATEDATE

Syntax:

CREATEDATE [*switches*]

Description: Retrieves the date and time at which the document was created, as recorded in the DateCreated element of the Core File Properties part. By default, the Gregorian calendar is used and the *date-and-time-formatting-switch* used is implementation-defined.

Field Value: The date and time at which the document was created.

Switches: Zero or one *date-and-time-formatting-switch* and zero or one of the following *field-specific-switches*.

\h	Use the Hijri Lunar or Hebrew Lunar calendar, depending on the language specified by the lang element (§17.3.2.20).
\s	Use the Saka Era calendar.

[*Example*: Consider the case in which the DateCreated element is as follows:

```
<DateCreated>2006-01-05T03:31:00Z</DateCreated>
```

and the following fields are updated in a US-English context that is UTC -5:

```
CREATEDATE
CREATEDATE \@ "dddd, MMMM dd, yyyy  HH:mm:ss"
CREATEDATE \@ "dddd, MMMM dd, yyyy  HH:mm:ss" \h
CREATEDATE \@ "dddd, MMMM dd, yyyy  HH:mm:ss" \s
```

the results are:

```
1/4/2006 10:31:00 PM
Wednesday, January 04, 2006 22:31:00
AlArbia'a, Thoul Hijjah 04, 1426 22:31:00
Budhavara, Pausa 14, 1927 22:31:00
```

end example]

17.16.5.12 DATABASE

Syntax:

DATABASE [*switches*]

DATABASE [switches *]*

Description: Inserts the results of a database query into a WordprocessingML table. If the number of columns is 62 or more, the field inserts the results of a query in columns separated by tabs. The DATABASE field contains all the information needed to connect to a database and perform an SQL query. Each time the field is updated, the database is queried again.

Field Value: The results of a database query as a WordprocessingML table.

Switches: Zero or more of the following *field-specific-switches*.

\b *field-argument*	The *text* in this switch's *field-argument* specifies which attributes of the format set by the \l switch are to be applied to the table. If the \l switch is blank, the \b switch value shall be 16 (AutoFit). *text* can have a value that is the bitwise-or of any combination of the following: 0, None 1, Borders 2, Shading 4, Font 8, Color 16, AutoFit 32, Heading Rows 64, Last Row 128, First Column 256, Last Column
\c *field-argument*	The *text* in this switch's *field-argument* specifies a connection to the data.
\d *field-argument*	The *text* in this switch's *field-argument* specifies the complete path and file name of the database. Used for all database queries except a query to an SQL database table using ODBC.
\f *field-argument*	The *text* in this switch's *field-argument* specifies the integral record number of the first data record to insert
\h	Inserts the field names from the database as column headings in the resulting table.
\l *field-argument*	The *text* in this switch's *field-argument* specifies the format that is to be applied to the result of the database query. If this switch is used and the \b switch doesn't specify the table attributes, an unformatted table is inserted.
\o *field-argument*	Inserts data at the beginning of a merge. By adding the \o switch to

	the database field, it only gets the data for the database field at the beginning of a merge instead of once for each record merged. This is a performance optimization and should only be used when the database field doesn't rely on record specific information to gather.
`\s` *field-argument*	The *text* in this switch's *field-argument* specifies a set of SQL instructions. Each quotation mark in the instructions shall be preceded by a backslash (\).
`\t` *field-argument*	The *text* in this switch's *field-argument* specifies the integral record number of the last data record to insert.

[*Example*: The following field results from a query to a database through ODBC:

```
{ DATABASE \d "C:\\Data\\Sales93.mdb" \c "DSN=MS Access Database;
  DBQ=C:\\Data\\Sales93.mdb; FIL=RedISAM"
  \s "select * from \"Customer List\"" \f "2445" \t "2486" \l "2"
```

end example]

17.16.5.13 DATE

Syntax:

DATE [*switches*]

DATE [switches *]*

Description: The current date and time. If no calendar is specified via a *field-specific-switch* or *date-and-time-formatting-switch,*the Gregorian calendar [ISO 8601, §3.2.1]is used. As specified in §17.16.4.1, if no *date-and-time-formatting-switch* is present, a date or time result is formatted in an implementation-defined manner.

Field Value: The current date and time.

Switches: Zero or one *date-and-time-formatting-switch* and zero or one of the following *field-specific-switches*.

`\h`	Use the Hijri Lunar or Hebrew Lunar calendar, depending on the language specified by the lang element (§17.3.2.20).
`\l`	When a field update is performed, if no *date-and-time-formatting-switch* is used, this switch is an instruction specifying that the field shall use the *date-and-time-formatting-switch* last used by the hosting application when inserting a new DATE field. If there is no last-used date format available, then the *date-and-time-formatting-switch* used is implementation-defined.
`\s`	Use the Saka Era calendar.

[*Example*: Consider the case in which the following fields are updated in a US-English context by an application with a default *date-and-time-formatting-switch* of M/d/yyyy on 2006-01-05T19:09:01:

```
DATE
DATE \@ "dddd, MMMM dd, yyyy  HH:mm:ss"
DATE \@ "dddd, MMMM dd, yyyy  HH:mm:ss" \h
DATE \@ "dddd, MMMM dd, yyyy  HH:mm:ss" \s
```

the results are:

```
1/5/2006
Thursday, January 05, 2006  19:09:01
AlKhamis, Thoul Hijjah 05, 1426  19:09:01
Bruhaspathivara, Pausa 15, 1927  19:09:01
```

end example]

[*Note*: For some *date-and-time-formatting-switch*es, the DATE and TIME (§17.16.5.65) fields can produce the same result. *end note*]

17.16.5.14 DOCPROPERTY

Syntax:

DOCPROPERTY *docprop-category* [*switches*]

Description: Retrieves the indicated document information. For some combinations of DOCPROPERTY and *docprop-category*, there is an equivalent field, in which case, the description for the combination can be obtained from that field. For those combinations not having an equivalent field, the description is shown directly. When used directly, some of the equivalent fields allow the value of the designated property to be changed. However, when the corresponding DOCPROPERTY field is used, such values shall not be changed. This is indicated in the following table by "Read-only operation."

docprop-category	Corresponding Field	Description
AUTHOR	AUTHOR (§17.16.5.4)	**Read-only operation.**
BYTES	FILESIZE (§17.16.5.18)	
CATEGORY	No equivalent	The contents of the Category element of the Core File Properties part.
CHARACTERS	NUMCHARS (§17.16.5.41)	
CHARACTERSWITHSPACES	No equivalent	**Like** NUMCHARS**, but includes all white space characters as well.**
COMMENTS	COMMENTS (§17.16.5.7)	**Read-only operation.**

docprop-category	Corresponding Field	Description
COMPANY	No equivalent	The contents of the Company element of the Application-Defined File Properties part.
CREATETIME	CREATEDATE (§17.16.5.11)	
HYPERLINKBASE	No equivalent	The contents of the HyperlinkBase element of the Application-Defined File Properties part.
KEYWORDS	No equivalent	The contents of the Keywords element of the Core File Properties part.
LASTPRINTED	PRINTDATE (§17.16.5.47)	
LASTSAVEDBY	LASTSAVEDBY (§17.16.5.31)	
LASTSAVEDTIME	SAVEDATE (§17.16.5.53)	
LINES	No equivalent	The contents of the Lines element of the Application-Defined File Properties part.
MANAGER	No equivalent	The contents of the Manager element of the Application-Defined File Properties part.
NAMEOFAPPLICATION	No equivalent	The contents of the Application element of the Application-Defined File Properties part.
ODMADOCID		
PAGES	NUMPAGES (§17.16.5.42)	
PARAGRAPHS	No equivalent	The contents of the Paragraphs element of the Application-Defined File Properties part.
REVISIONNUMBER	REVNUM (§17.16.5.52)	
SECURITY	No equivalent	The contents of the DocSecurity element of

©ISO/IEC 2012 – All rights reserved

docprop-category	Corresponding Field	Description
		the Application-Defined File Properties part.
SUBJECT	SUBJECT (§17.16.5.60)	**Read-only operation.**
TEMPLATE	TEMPLATE (§17.16.5.64)	
TITLE	TITLE (§17.16.5.66)	**Read-only operation.**
TOTALEDITINGTIME	EDITTIME (§17.16.5.16)	
WORDS	No equivalent	The contents of the Words element of the Application-Defined File Properties part.

Field Value: The indicated document information.

17.16.5.15 DOCVARIABLE

Syntax:

DOCVARIABLE *field-argument*

DOCVARIABLE field-argument

Description: Inserts the string assigned to the document variable designated by *text* in *field-argument*. Each WordprocessingML document has a collection of variables. This field is used to access and display the contents of docVar (§17.15.1.31) elements in the Document Settings part.

Field Value: The value of the specified document variable.

Switches: None.

17.16.5.16 EDITTIME

Syntax:

EDITTIME [*switches*]

Description: Retrieves the total editing time, in minutes, since the document was created, as recorded in the TotalTime element of the Application-Defined File Properties part. By default, the *numeric-formatting-switch* or *general-formatting-switch* used is implementation-defined.

Field Value: The total editing time, in minutes.

Switches: Zero or one *numeric-formatting-switch* or *general-formatting-switch*.

[*Example*: Consider the case in which the TotalTime element is as follows:

```
<TotalTime>930</TotalTime>
```

and the following fields are updated in a US-English context:

EDITTIME
EDITTIME * OrdText
EDITTIME \# "#,##0"

the results are:

```
930
nine hundred thirtieth
 930
```

end example]

17.16.5.17 FILENAME

Syntax:

`FILENAME` [*switches*]

Description: Retrieves the name of the current document from its storage location.

Field Value: The name of the current document.

Switches: One of the following *general-formatting-switches*: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`, and zero or one of the following *field-specific-switches*.

\p	Include the full file path name.

[*Example*: Consider the case in which the following fields are updated:

```
FILENAME \* Upper
FILENAME \p
```

the results might be:

```
FIELD DEMO SUITE.DOCX
E:\Std\OOXML\Fields\Field Demo Suite.docx
http://example.com/files/myFile.docx
```

end example]

17.16.5.18 FILESIZE

Syntax:

`FILESIZE` [*switches*]

Description: The size of the WordprocessingML package in bytes, or the empty string if the size of the package cannot be determined when a field update is performed. [*Example*: If the package is being streamed, and its size

©ISO/IEC 2012 – All rights reserved

is not yet known by the application reading that stream. *end example*] [*Note*: This information is not stored inside the document's XML. *end note*]

Field Value: The size of the current document in bytes.

Switches: Zero or one *numeric-formatting-switch* or *general-formatting-switch* and zero or one of the following *field-specific-switches*.

`\k`	Round to the nearest thousand bytes.
`\m`	Round to the nearest million bytes.

[*Example*: Consider the case in the document's size is 4,660,736 bytes and the following fields are updated:

```
FILESIZE \# #,##0
FILESIZE \k
FILESIZE \m
```

the results are:

```
4,660,736
4661
5
```

end example]

17.16.5.19 FILLIN

Syntax:

`FILLIN` [*field-argument*] [*switch*]

Description: Prompts the user to enter text. *text* in *field-argument* contains the prompt. The prompt is displayed each time the field is updated. When a new document is created based on a template containing `FILLIN` fields, those fields are updated automatically.

Field Value: The user's response.

Switches: Zero or more of the following *field-specific-switches*.

`\d` *field-argument*	The *text* in this switch's *field-argument* specifies a default response if one is not entered. If no default response is specified, the most recent response is used. To specify a blank entry as the default, *field-argument* shall be `""`.
`\o`	When used in a mail merge main document, this causes the display of the prompt once instead of each time a new data record is merged. The same response is inserted in each resulting merged document.

[*Example*: The following `FILLIN` field helps the user fill in the correct information by displaying the patient name from the current data record:

```
FILLIN "Please enter the appointment time for MERGEFIELD PatientName :"
```

end example]

17.16.5.20 FORMCHECKBOX

Syntax:

```
FORMCHECKBOX
```

Description: Inserts a check box style form field which, when the editing of form fields is enabled using the documentProtection element (§17.15.1.29), can be checked and unchecked. An instance of this field shall be accompanied by a use of the ffData element (§17.16.17) which contains the form field's properties.

Field Value: A check box based on the properties of the ffData element (§17.16.17).

Switches: None.

[*Example*: Consider the following fields:

```
FORMCHECKBOX
```

Assuming the appropriate properties are used in the child XML elements of the field, a check box is displayed.

end example]

17.16.5.21 FORMDROPDOWN

Syntax:

```
FORMDROPDOWN
```

Description: Inserts a drop-down list style form field which, when the editing of form fields is enabled using the documentProtection element (§17.15.1.29), can be used to select an entry in the list. An instance of this field shall be accompanied by a use of the ffData element (§17.16.17) which contains the form field's properties.

Field Value: A drop-down list based on the properties of the ffData element (§17.16.17).

Switches: None.

[*Example*: Consider the following fields:

```
FORMDROPDOWN
```

Assuming the appropriate properties are used in the child XML elements of the field, a drop-down list is displayed.

end example]

17.16.5.22 FORMTEXT

Syntax:

```
FORMTEXT
```

Description: Inserts a text box style form field which, when the editing of form fields is enabled using the documentProtection element (§17.15.1.29), can be typed into. An instance of this field shall be accompanied by a use of the ffData element (§17.16.17) which contains the form field's properties.

Field Value: A text box based on the properties of the ffData element (§17.16.17).

Switches: None.

[*Example*: Consider the following fields:

```
FORMTEXT
```

Assuming the appropriate properties are used in the child XML elements of the field, a text box is displayed.

end example]

17.16.5.23 GOTOBUTTON

Syntax:

GOTOBUTTON *field-argument-1 field-argument-2*

field-argument-1:
expression

field-argument-2:
expression

Description: Inserts a jump command, such that when it is activated, the insertion point of the document is moved to the location specified by *text* in *field-argument-1*. *text* can be a bookmark, a page number, or some other item (as described below). The page number can be a reference resulting from a REF field. The other items than can be locations are:

`a` *n*	annotation
`f` *n*	footnote
`l` *n*	line
`p` *n*	page
`s` *n*	section

where *n* is an integer that designates the *n*th occurrence of the corresponding item (which is not necessarily the item numbered *n*).

text in *field-argument-2* is the text or graphic "button" that appears in the document, such that it can be selected to activate the jump. [*Note*: The BOOKMARK and INCLUDEPICTURE fields make for some interesting button possibilities. *end note*] The text or graphic shall appear on one line in the field result; otherwise, an error occurs.

Field Value: None.

Switches: None.

[*Example*: Consider the following fields:

```
GOTOBUTTON MyBookmark Dest
GOTOBUTTON p3 Page
GOTOBUTTON "f 2" Footnote
```

When the Dest "button" is activated, the insertion point becomes the location marked by MyBookmark. When Page is activated, the insertion point becomes the beginning of the third page. When Footnote is activated, the insertion point becomes the marker of the second footnote, at the place it is used in the document, not in any footnote list.

end example]

17.16.5.24 GREETINGLINE

Syntax:

GREETINGLINE [*switches*]

Description: Inserts a mail merge greeting line.

Field Value: The greeting line.

Switches: Zero or more of the following *field-specific-switches*.

\c *field-argument*	The *text* in this switch's *field-argument* specifies the text to include in the merge field if the name field in the data source is blank.
\c *field-argument*	The *text* in this switch's *field-argument* specifies the format of the name included in the field.
\l *field-argument*	The *text* in this switch's *field-argument* specifies the language ID used to format the name. it defaults to the language ID of the first character of the document. This language ID is specified in the format defined by ST_Lang (§22.9.2.6)

17.16.5.25 HYPERLINK

Syntax:

HYPERLINK *field-argument* [*switches*]

Description: When selected, causes control to jump to the location specified by *text* in *field-argument*. That location can be a bookmark or a URL.

Field Value: None.

Switches: Zero or more of the following *field-specific-switches*.

`\l` *field-argument*	*text* in this switch's *field-argument* specifies a location in the file, such as a bookmark, where this hyperlink jumps.
`\m`	Appends coordinates to a hyperlink for a server-side image map.
`\n`	Causes the destination site to be opened in a new web browser window.
`\o` *field-argument*	*text* in this switch's *field-argument* specifies the ScreenTip text for the hyperlink.
`\t` *field-argument*	*text* in this switch's *field-argument* specifies the target to which the link should be redirected. Use this switch to link from a frames page to a page that you want to appear outside of the frames page. The permitted values for *text* are: • `_top`, whole page (the default) • `_self`, same frame • `_blank`, new web browser window • `_parent`, parent frame

[*Example*:

```
HYPERLINK http://www.example.com/
HYPERLINK "E:\\ReadMe.txt"
```

end example]

17.16.5.26 IF

Syntax:

`IF` *comparison field-argument-1 field-argument-2*

field-argument-1:
expression

field-argument-2:
expression

Description: Compares the values designated by the *expression*s in *comparison* using the operator designated by *comparison-operator*.

[*Note*: *comparison-operator* can be any one of the six relational and equality operators specified for *operator* (§17.16.3.3). *end note*]

If *comparison-operator* is = or <>, the left-hand *expression* operand can contain a question mark (?) to represent any single character, or an asterisk (*) to represent any string of characters.

Field Value: *field-argument-1* if the comparison is true; otherwise, *field-argument-2*.

Switches: None.

[*Example*: The following example specifies that if the customer order is greater than or equal to 100 units, the result is "Thanks"; but if the customer order is fewer than 100 units, the result is "The minimum order is 100 units":

```
{IF order >= 100 "Thanks" "The minimum order is 100 units" }
```

For other examples, see §17.16.2, and the COMPARE field (§17.16.5.10) and the QUOTE field (§17.16.5.48). *end example*]

17.16.5.27 INCLUDEPICTURE

Syntax:

INCLUDEPICTURE *field-argument* [*switches*]

Description: Retrieves the picture named by *field-argument* and displays it as the field result. *Field-argument* specifies the location of the picture using an IRI.

[*Note*: A list of possible image formats is provided in §15.2.14. *end note*]

Field Value: The specified picture.

Switches: Zero or more of the following *field-specific-switches*.

\c *field-argument*	If *text* in this switch's *field-argument* identifies the graphics filter to be used.
\d	Reduce the file size by not storing graphics data with the document.

[*Example*:

```
INCLUDEPICTURE "file:///g:/photos/Ellen%20in%20Oslo.jpg"
```

end example]

17.16.5.28 INCLUDETEXT

Syntax:

INCLUDETEXT *field-argument-1* [*field-argument-2*] [*switches*]

©ISO/IEC 2012 – All rights reserved

field-argument-1:
field-argument

field-argument-2:
field-argument

Description: Inserts all or part of the text and graphics contained in the document named by *field-argument-1*. *field-argument-1* specifies the location of the document using an IRI (Duerst and Suignard 2005). If the document is a WordprocessingML document, the portion marked by the optional bookmark *field-argument-2* is inserted. If no such bookmark is specified here, the whole document is inserted. If the document is an XML file, the fragment referred to by an XPath expression in the `\x` switch is inserted. If no such switch is specified, the whole XML file is inserted.

If *field-argument-1* contains white space, it shall be enclosed in double quotes.

[*Note*: A list of possible text formats is provided in §11.3.1. *end note*]

Field Value: The specified text and graphics.

Switches: Zero or more of the following *field-specific-switches*.

`\!`	Prevents this field from being updated unless all fields in the inserted text are first updated in their original document.
`\c` *field-argument*	Specifies that the file specified by *field-argument-2* shall be processed by a document filter whose name matches the corresponding *field-argument* value. Possible *field-argument* values are implementation-defined.
`\e`	Specifies the encoding applied to the data within the file referenced by *field-argument-1*. If this argument is omitted, applications should attempt to determine the encoding by reading the file’s contents if possible.
`\m`	Specifies the MIME type of the file referenced by *field-argument-1*, as defined by http://www.iana.org/assignments/media-types/. If this argument is omitted, applications should attempt to determine the file type from its contents if possible.
`\n` *field-argument*	The *text* in this switch's *field-argument* specifies a namespace mapping for XPath queries. This switch is required if the `\x` switch refers to an element by name in an XML file that declares a namespace.
`\t` *field-argument*	The *text* in this switch's *field-argument* specifies an XSLT for formatting XML data.
`\x` *field-argument*	The *text* in this switch's *field-argument* specifies the XPath for returning a fragment of data in an XML file.

[*Example*: The following field inserts the portion of the WordprocessingML document referred to by the bookmark `Summary`:

```
INCLUDETEXT "file:///C:/Winword/Port Development RFP" Summary
```

The following field inserts the `Name` element of the XML document `Resume.xml` and applies the XSLT `Display.xsl` to it:

```
INCLUDETEXT "file:///C:/Resume.xml" \n xmlns:a=\"resume-schema\"
  \t "file:///C:/display.xsl" \x a:Resume/a:Name
```

end example]

17.16.5.29 INDEX

Syntax:

`INDEX` [*switches*]

Description: Builds an index using the index entries specified by XE fields (§17.16.5.72), and inserts that index at this place in the document. Each index entry and subentry is a separate paragraph unless the `\r` switch is used, in which case, an index entry and all its subentries together make up a paragraph.

Field Value: The index.

Switches: Zero or more of the following *field-specific-switches*.

`\b` *field-argument*	Builds an index for the portion of the document marked by the bookmark indicated by *text* in this switch's *field-argument*.
`\c` *field-argument*	Builds an index having the number of columns per page specified by *text* in this switch's *field-argument*. That number can be 1–4. Without this switch, the number of columns is 1.
`\d` *field-argument*	The *text* in this switch's *field-argument* specifies a sequence of characters that is used to separate sequence numbers and page numbers when the `\s` switch is used. By default, a hyphen (-) is used.
`\e` *field-argument*	The *text* in this switch's *field-argument* specifies a sequence of characters that is used to separate an index entry and its first page number. By default, a comma (,) and space sequence is used. If *text* contains a horizontal tab character, the page number list is right justified in the column.
`\f` *field-argument*	Builds an index using only those entries having the entry type (§17.16.5.72) specified by *text* in this switch's *field-argument*. Without this switch, all entries included.
`\g` *field-argument*	The *text* in this switch's *field-argument* specifies a sequence of characters that is used to separate the start and end of a page range. By default, an en dash is used.
`\h` *field-argument*	Builds an index such that the *text* in this switch's *field-argument* occurs as a heading—formatted with the Index Heading style—at the start of each set of entries for any given letter. If the first letter in *text* is A or a, that letter is replaced with the corresponding letter for each letter set. To replace the default heading with a blank line, use a space as *text*. [*Example*:

	`INDEX \h "a Entries Follow" \c "1" \z "1033"` **F Entries Follow** fox, 1 **Q Entries Follow** quick brown fox. *See* Fox *end example*]
`\k` *field-argument*	The *text* in this switch's *field-argument* specifies a sequence of characters that is used to separate an index entry and its cross reference (as produced by an XE entry (§17.16.5.72) having a `\t` switch). By default, a period (.) and space sequence is used. [*Example*: The quick brown fox{ XE "fox" } jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox jumps over the lazy dog. The quick brown fox{ XE "quick brown fox" \t "See Fox" } jumps over the lazy dog. The quick brown fox jumps over the lazy dog. INDEX \k <test> \c "1" \z "1033" fox, 1 quick brown fox <test>See Fox *end example*]
`\l` *field-argument*	The *text* in this switch's *field-argument* specifies a sequence of characters that is used to separate two page numbers in a page number list. By default, a comma (,) and space sequence is used.
`\p` *field-argument*	Builds an index using only those entries whose first letter is in the range of letters specified by *text* in this switch's *field-argument*. The letter range has the form *startLetter*-*endLetter*. If *startLetter* is `!`, entries whose first character is not a letter, are also included, as are the letters starting from A. The letters in the range can be either upper- or lowercase.
`\r`	Runs subentries into the same line as the main entry. Colons (:) separate main entries from subentries; semicolons (;) separate subentries.
`\s` *field-argument*	The *text* in this switch's *field-argument* is used as a sequence name, and the sequence number is included along with the page number, these numbers being separated by a hyphen (-), by default. Use the `\d` switch to specify a separator character other than the default.
`\y`	Enables the use of yomi text for index entries.
`\z` *field-argument*	The *text* in this switch's *field-argument* specifies the language ID used to generate the index. This language ID is specified in the format defined by ST_Lang (§22.9.2.6)

[*Example*: The index produced using the corresponding set of index entries and the field `INDEX \c "1" \e "`*tab*`" \g " to " \h "A" \z "1033"` is:

B

behavior

  implementation-defined **2**

    documenting **3**

I

item

  package-relationship .... See package-relationship item

O

Office Open XML **2**, *3*, 4

X

XML .... 1 to 4

end example]

17.16.5.30 KEYWORDS

Syntax:

    KEYWORDS [*field-argument*] [*switches*]

Description: Retrieves, and optionally sets, the document's keywords, as recorded in the Keywords element of the Core File Properties part or, if *field-argument* is present, the subject specified by *text* in *field-argument*. Specifying a *field-argument* shall change Keywords to *text*. The Keywords element contains a string of text whose format and semantics is unspecified by ISO/IEC 29500.

Field Value: The document's keywords

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Consider the case in which the Keywords element is as follows:

```
<Keywords>switch, field, syntax</Keywords>
```

and the following field is updated:

KEYWORDS

The result is:

```
switch, field, syntax
```

Updating the following field:

```
KEYWORDS "field, formatting, switch, syntax"
```

©ISO/IEC 2012 – All rights reserved

causes the Subject element to take on the given value. *end example*]

17.16.5.31 LASTSAVEDBY

Syntax:

LASTSAVEDBY [*switches*]

Description: Retrieves the name of the user who last modified and saved the current document, as recorded in the LastModifiedBy element of the Core File Properties part.

Field Value: The name of the user who last modified and saved the current document.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Consider the case in which the LastModifiedBy element is as follows:

```
<LastModifiedBy>Elizabeth Martin</LastModifiedBy>
```

and the following field is updated:

```
LASTSAVEDBY \* Upper
```

the result might be:

```
ELIZABETH MARTIN
```

end example]

17.16.5.32 LINK

Syntax:

LINK *field-argument-1* *field-argument-2* [*field-argument-3*] [*switches*]

field-argument-1:
field-argument

field-argument-2:
field-argument

field-argument-3:
field-argument

Description: For information copied from another application, this field links that information to its original source file. The application type of the link information is specified by *field-argument-1*. The name and location of the source file is specified by *field-argument-2*. *field-argument-3* specifies the portion of the source file that's being linked. [*Example*: If the source file is a SpreadsheetML document, the reference might be to a cell reference or a named range. For a WordprocessingML document, it might be a bookmark. *end example*]

Field Value: None.

Switches: Zero or more of the following *field-specific-switches*.

`\a`	Causes this field to be updated automatically.
`\b`	Inserts the linked object as a bitmap.
`\d`	Don’t store the graphic data with the document, thus reducing the file size.
`\f` *field-argument*	Causes the linked object to update its formatting in a particular way, according to the integral value of *text* in this switch's *field-argument*. The possible values are: • 0 Maintain the formatting of the source file • 1 Not supported • 2 Match the formatting of the destination document • 3 Not supported • 4 Maintain the formatting of the source file, if the source file is a SpreadsheetML workbook • 5 Match the formatting of the destination document, if the source file is a SpreadsheetML workbook
`\h`	Inserts the linked object as HTML format text.
`\p`	Inserts the linked object as a picture.
`\r`	Inserts the linked object in rich-text format (RTF).
`\t`	Inserts the linked object in text-only format.
`\u`	Inserts the linked object as Unicode text.

[*Example*: The following example inserts a range of cells from a SpreadsheetML worksheet. The `\a` switch ensures that the information is updated in the WordprocessingML document whenever the worksheet is changed:

```
{ LINK Excel.Sheet.8 "C:\\My Documents\\Profits.xls"
   "Sheet1!R1C1:R4C4" \a \p }
```

end example]

17.16.5.33 LISTNUM

Syntax:

`LISTNUM` [*field-argument*] [*switches*]

Description: Computes the next value from a specific level of a numbering definition, or a specific valuefrom a specific level of a numbering definition.

The terms numbering definition and level are used as defined in §17.9. Within the context of this field, determining the numbering definition is equivalent to reading the value of the numId element (§17.9.18), and determining the level is equivalent to reading to value of the ilvl element (§17.9.3). [*Rationale*: This field is distinct from the numbering defined by the syntax in §17.9. This syntax stores the numbering as run content, rather than a paragraph property. This allows multiple numbered items to occur in a single paragraph; at locations other than the start of the paragraph, etc. *end rationale*]

For LISTNUM fields, the associated numbering definition is calculated using the first of the following conditions met:

- If *field-argument* is `NumberValue`, `OutlineDefault`, or `LegalDefault`, all items with each value are part of a single numbering definition.
- If the current paragraph has numbering applied, all items within that paragraph are part of its numbering definition.
- If field-argument is present, the field belongs to the numbering definition used by level one of the abstract numbering definition (§17.9.1) whose name element (§17.9.13) matches *field-argument*. If no such list exists, then the field shall behave as though field-argument was omitted.
- If *field-argument* has been omitted, the field belongs to the same numbering definition as the last instance of a `LISTNUM` field or numbered paragraph present in the document.

Once the numbering definition has been determined, the level is calculated as follows:

- If the `\l` switch is present, the value of that switch overrides all subsequent rules.
- If *field-argument* is `NumberValue`, `OutlineDefault`, or `LegalDefault:`

  - The first instance of the field is level one.
  - Each instance of the field which occurs at the start of the paragraph preceded only by other `LISTNUM` fields (or nothing) increments the level by one. [*Example*: A `LISTNUM` field with no field-argument inserted as the first item in a paragraph using level one of a numbering definition is at level two. *end example*]
  - The first instance of the field which occurs after other paragraph content increments the level by one, and all subsequent `LISTNUM` fields belong to the same level. [*Example*: Consider the following paragraph content: `{LISTNUM} {LISTNUM} test {LISTNUM} test2 {LISTNUM} {LISTNUM}`. Within this paragraph, the first two instances of the field are levels one and two, and the third through fifth instances of the field are part of level three. *end example*]

- If *field-argument* has any other value:

  - If the current paragraph has numbering applied, that paragraph numbering resets the level to one.

    - Each `LISTNUM` field which occurs at the start of the paragraph preceded only by other `LISTNUM` fields (or nothing) increments the level by one. [*Example*: A `LISTNUM` field with no field-argument inserted as the first item in a paragraph using level one of a numbering definition is at level two. *end example*]
    - The first `LISTNUM` field which occurs after other paragraph content increments the level by one, and all subsequent `LISTNUM` fields belong to the same level. [*Example*: Consider the following paragraph content: `{LISTNUM} {LISTNUM} test {LISTNUM} test2 {LISTNUM}`

{LISTNUM}. Within this paragraph, the first two instances of the field are levels one and two, and the third through fifth instances of the field are part of level three. *end example*]

- Fields in paragraphs with no numbering applied start at the same level as the last instance of a LISTNUM field or numbered paragraph present in the document.

The XML generated for a complex field implementation shall not have the optional field value stored.

Although the value of a LISTNUM field is calculated as a consecutive-valued non-negative integer, once a value is computed, the corresponding integer is rendered in the appropriate format, as described below. In all possible display methods, the list entry value is considered to be a number, and the list to be a numbered list. There are nine levels of list, and the result style used for each is determined based on the *field-argument* as follows:

- If the *field-argument* is NumberValue, OutlineDefault, or LegalDefault, the formatting is based on the table below.
- For any other value, the formatting is based on the abstract numbering definition (§17.9.1) whose name element (§17.9.13) matches *field-argument*. If no such list exists, then the field shall behave as though *field-argument* was omitted.
- If *field-argument* has been omitted:
  - If the current paragraph has numbering applied, the field shall take on the format of the next level of the corresponding numbering definition. [*Example*: A LISTNUM field inserted into a paragraph using level one of a numbering definition is displayed in the format defined by level two of that numbering definition. *end example*]
  - If not, the field takes on the numbering format of the last instance of a LISTNUM field or numbered paragraph present in the document.

The following tables describe the predefined sequences NumberValue, OutlineDefault, and LegalDefault. The formats described are used as defined by ST_NumberFormat (§17.18.59).

Level	**NumberValue Value Content and Format**
1	The decimal format followed immediately by ")". [*Example*: "3)", "9)", and "15)". *end example*].
2	The lowerLetter format followed immediately by ")". [*Example*: "c)", "i)", and "o)". *end example*].
3	The lowerRoman format followed immediately by ")". [*Example*: "iii)", "ix)", and "xv)". *end example*].
4	The decimal format surrounded by "(" and ")", without any intervening white space. [*Example*: "(3)", "(9)", and "(15)". *end example*].
5	The lowerLetter format surrounded by "(" and ")", without any intervening white space. [*Example*: "(c)", "(i)", and "(o)". *end example*].
6	The lowerRoman format surrounded by "(" and ")", without any intervening white space. [*Example*: "(iii)", "(ix)", and "(xv)". *end example*].
7	The decimal format followed immediately by ".". [*Example*: "3.", "9.", and "15.". *end example*].

©ISO/IEC 2012 – All rights reserved

Level	NumberValue Value Content and Format
8	The lowerLetter format followed immediately by “.”. [*Example*: “c.”, “i.”, and “o.”. *end example*].
9	The lowerRoman format followed immediately by “.”. [*Example*: “iii.”, “ix.”, and “xv.”. *end example*].

Level	OutlineDefault Value Content and Format
1	The upperRoman format followed immediately by “.”. [*Example*: “III.”, “IX.”, and “XV.”. *end example*].
2	The upperLetter format followed immediately by “.”. [*Example*: “C.”, “I.”, and “O.”. *end example*].
3	The decimal format followed immediately by “.”. [*Example*: “3.”, “9.”, and “15.”. *end example*].
4	The lowerLetter format followed immediately by “)”. [*Example*: “c)”, “i)”, and “o)”. *end example*].
5	The decimal format surrounded by “(“ and “)”, without any intervening white space. [*Example*: “(3)”, “(9)”, and “(15)”. *end example*].
6	The lowerLetter format surrounded by “(“ and “)”, without any intervening white space. [*Example*: “(c)”, “(i)”, and “(o)”. *end example*].
7	The lowerLetter format defined by ST_NumberFormat surrounded by “(“ and “)”, without any intervening white space. [*Example*: “(iii)”, “(ix)”, and “(xv)”. *end example*].
8	The lowerLetter format surrounded by “(“ and “)”, without any intervening white space. [*Example*: “(c)”, “(i)”, and “(o)”. *end example*].
9	The lowerRoman format surrounded by “(“ and “)”, without any intervening white space. [*Example*: “(iii)”, “(ix)”, and “(xv)”. *end example*].

Level	LegalDefault Value Content and Format
1	The decimal format followed immediately by “.”. [*Example*: “3.”, “9.”, and “15.”. *end example*].
2–9	A sequence consisting of the last number from each previous level (delimited by “.” followed immediately by the next value at this level and “.” (all in the decimal format). [*Example*: “1.3.”, “2.4.9.”, and “1.3.7.2.15.”. *end example*].

Field Value: The next value, or a specific value, from the numbering definition which this instance of the LISTNUM field belongs to (as described above).

Switches: Zero or more of the following *field-specific-switches*.

\l *field-argument*	*text* in this switch's *field-argument* is an integer that specifies the level in the list, overriding the default behavior of the field. If \l is omitted, for a new series, the default value is 3; otherwise, the current level is continued.
\s *field-argument*	*text* in this switch's *field-argument* is an integer that specifies the starting value for this field. If \s is omitted, for a new series, the default value is 1; otherwise, the current series numbering is continued.

[*Example*: When the following fields are updated:

```
We need to perform the following functions: LISTNUM  NumberDefault \l 3 \s 1 Get
approval for the project. LISTNUM Arrange for funding. LISTNUM Hire staff.
```

The results are:

```
We need to perform the following functions: i) Get approval for the
project. ii) Arrange for funding. iii) Hire staff.
```

When the following fields are updated:

```
LISTNUM NumberDefault \l 3 \s 1
LISTNUM
LISTNUM NumberDefault
LISTNUM NumberDefault \s 3
LISTNUM
LISTNUM NumberDefault \l 1
LISTNUM
LISTNUM NumberDefault \l 1 \s 1
LISTNUM LegalDefault \1 1 \s 1
LISTNUM LegalDefault
LISTNUM LegalDefault \l 1
LISTNUM LegalDefault \s 4
LISTNUM LegalDefault
```

The results are:

```
i)
ii)
iii)
iii)
iv)
2)
3)
1)
1.1.1.
1.1.2.
2.
2.1.4.
2.1.5.
```

end example]

17.16.5.34 MACROBUTTON

Syntax:

©ISO/IEC 2012 – All rights reserved

```
MACROBUTTON field-argument-1  field-argument-2
field-argument-1:
    field-argument
field-argument-2:
    field-argument
```

Description: Allows the macro or command designated by *text* in *field-argument-1* to be run. *text* in *field-argument-2* designates the text or graphic to appear as the "button" that is selected to run the macro or command. The mechanism by which the command specified by *text* in *field-argument-1* is located and/or executed by an application is implementation-defined.

Field Value: *field-argument-2*.

Switches: None.

17.16.5.35 MERGEFIELD

Syntax:

```
MERGEFIELD field-argument  [ switches ]
```

Description: Retrieves the name of a data field designated by *text* in *field-argument* within the merge characters in a mail merge main document. When the main document is merged with the selected data source, information from the specified data field is inserted in place of the merge field.

The name designated by *text* shall match exactly the field name in the header record.

Field Value: The name of a data field designated by *text* in *field-argument*.

Switches: Zero or more of the following *field-specific-switches*.

\b *field-argument*	The *text* in this switch's *field-argument* specifies the text to be inserted before the MERGEFIELD field if the field is not blank.
\f *field-argument*	The *text* in this switch's *field-argument* specifies the text to be inserted after the MERGEFIELD field if the field is not blank.
\m	Specifies that the MERGEFIELD field is a mapped field.
\v	Enables character conversion for vertical formatting.

[*Example*: Placing the following three MERGEFIELD fields together and using the \f switch ensures that the three fields have spaces between them, but only if the designated field information is present in the data source.

```
MERGEFIELD CoutesyTitle \f " " MERGEFIELD FirstName \f " "
  MERGEFIELD LastName
```

end example]

17.16.5.36 MERGEREC

Syntax:

```
MERGEREC
```

Description: When an application uses the contents of this document to perform a mail merge, this field is an instruction that the *field result* display the number of the corresponding data record in the merged document. The number reflects the sequential order of the data records that were selected and possibly sorted for merging with the active main document. It does not indicate the actual order of the records as they occur in the physical data source. [*Note*: A personnel database might contain thousands of records. However, to send a form letter to employees who've reached their five-year anniversary with your company, you'd select as your data source only the records of those five-year employees, a much smaller set of records. To print a physical record number, you must include a record number field in the data source and insert the corresponding merge field in the main document. *end note*]

Field Value: The literal text «MERGEREC» unless a mail merge is being performed, in which case, the number of the data record being merged.

Switches: None.

[*Example*: The following example uses a MERGEREC field inside a formula to create unique invoice numbers. When the main document is merged with the data source, the number resulting from the MERGEREC field is added to the numbers representing the date and time the invoices are printed.

```
Invoice Number: = { PRINTDATE \@ "MMddyyyyHHmm" + MERGEREC }
```

The result might be:

```
Invoice Number for record 12, printed on Feb. 13, 2003 at 9:46:
  02132003094612
```

end example]

17.16.5.37 MERGESEQ

Syntax:

```
MERGESEQ
```

Description: Counts the number of data records that were successfully merged with the main document. Merged records are numbered starting from 1 each time documents are merged. [*Note*: The number might be different from the value inserted by the MERGEREC field. *end note*]

Field Value: The number of data records that were successfully merged with the main document.

Switches: None.

[*Example*: Consider the case in which only records 10–25 are merged. The MERGESEQ number corresponding to the first data record merged is 1, even though the MERGEREC number for that data record is 10. *end example*]

17.16.5.38 NEXT

Syntax:

```
NEXT
```

Description: Merges the next data record into the current resulting merged document, rather than starting a new merged document. [*Note*: This field is used when setting up a mailing label and envelope main document during a mail merge. *end note*]

Field Value: None.

Switches: None.

[*Example*: The following fields print three sets of names and phone numbers in each resulting merged document:

```
MERGEFIELD Name MERGEFIELD Phone
NEXT MERGEFIELD Name MERGEFIELD Phone
NEXT MERGEFIELD Name MERGEFIELD Phone
```

end example]

17.16.5.39 NEXTIF

Syntax:

NEXTIF *comparison*

Description: Compares the values designated by the *expression*s in *comparison* using the operator designated by *comparison-operator*. If the comparison is true, the next data record is merged into the current merge document. (Merge fields that follow the NEXTIF in the main document are replaced by values from the next data record rather than the current data record.) If the comparison is false, the next data record is merged into a new merge document.

[*Note*: *comparison-operator* can be any one of the six relational and equality operators specified for *operator* (§17.16.3.3). *end note*]

A NEXTIF fields shall not be used in a footnote, an endnote, annotation, a header, a footer, or a data source. A NEXTIF field shall not be nested within any field.

If *comparison-operator* is = or <>, the right-hand *expression* operand can contain a question mark (?) to represent any single character, or an asterisk (*) to represent any string of characters. The expression shall be enclosed in quotation marks so that it is compared as a character string.

Field Value: None.

Switches: None.

17.16.5.40 NOTEREF

Syntax:

NOTEREF *field-argument* [*switches*]

Description: Inserts the mark of the footnote or endnote that is marked by the bookmark specified by *text* in *field-argument*.

Field Value: The mark of the footnote or endnote.

Switches: Zero or more of the following *field-specific-switches*.

\f	For a footnote, inserts the reference mark with the same character formatting as the Footnote Reference style. For an endnote, inserts the reference mark with the same character formatting as the Endnote Reference style.
\h	Inserts a hyperlink to the bookmarked endnote or footnote.
\p	Inserts the relative position of the footnote or endnote. If the NOTEREF field occurs before the bookmark, the result is "below". If the NOTEREF field occurs after the bookmark, the result is "above".

[*Example*: Consider the case in which a bookmark called F10 marks the footnote of interest. When the field

```
… (see note { NOTEREF F10 }).
```

is updated, the result might be:

```
… (see note 5).
```

end example]

17.16.5.41 NUMCHARS

Syntax:

NUMCHARS [*switches*]

Description: Retrieves the number of characters in the current document, as recorded in the Characters element of the Application-Defined File Properties part.

Field Value: The number of characters in the current document.

Switches: Zero or one *numeric-formatting-switch* or *general-formatting-switch*.

[*Example*: Consider the case in the document has 6,183 words and the following fields are updated:

```
NUMCHARS
NUMCHARS \# #,##0
```

the results are:

```
6183
6,183
```

end example]

17.16.5.42 NUMPAGES

Syntax:

`NUMPAGES` [*switches*]

Description: Retrieves the number of pages in the current document, as recorded in the Pages element of the Application-Defined File Properties part.

Field Value: The number of pages in the current document.

Switches: Zero or one *numeric-formatting-switch* or *general-formatting-switch*.

[*Example*: Consider the case in the document has 19 pages and the following fields are updated:

```
NUMPAGES \# #,##0
NUMPAGES \* OrdText
```

the results are:

```
 19
nineteenth
```

end example]

17.16.5.43 NUMWORDS

Syntax:

`NUMWORDS` [*switches*]

Description: Retrieves the number of words in the current document, as recorded in the Words element of the Application-Defined File Properties part.

Field Value: The number of words in the current document.

Switches: Zero or one *numeric-formatting-switch* or *general-formatting-switch*.

[*Example*: Consider the case in the document has 1,243 words and the following fields are updated:

```
NUMWORDS
NUMWORDS \# #,##0
```

the results are:

```
1243
1,243
```

end example]

17.16.5.44 PAGE

Syntax:

PAGE [*switches*]

Description: Retrieves the number of the current page.

Field Value: The number of the current page.

Switches: Zero or more *general-formatting-switch*es.

[*Example*: When the current page number is 19 and the following fields are updated:

```
PAGE
PAGE \* ArabicDash
PAGE \* ALPHABETIC
PAGE \* roman
```

the results are:

```
19
- 19 -
S
xix
```

end example]

17.16.5.45 PAGEREF

Syntax:

PAGEREF *field-argument* [*switches*]

Description: Inserts the number of the page containing the bookmark specified by *text* in *field-argument* for a cross-reference.

Field Value: The number of the page containing the bookmark.

Switches: Zero or one of the *general-formatting-switch*es, zero or one of the *numeric-formatting-switch*es, and zero or more of the following *field-specific-switches*.

\h	Creates a hyperlink to the bookmarked paragraph.
\p	Causes the field to display its position relative to the source bookmark. If the PAGEREF field is on the same page as the bookmark, it omits "on page #" and returns "above" or "below" only. If the PAGEREF field is not on the same page as the bookmark, the string "on page #" is used.

[*Example*: Consider the case in which a bookmark called `Worldpop1990` marks the table containing figures for 1990. When the field

```
The world population in 1991 was 5 billion; for 1990 figures,
see the table { PAGEREF Worldpop1990 \p }.
```

is updated, the position of the table is inserted in place of the field. The result is one of the following::

```
… see the table above.
… see the table below.
… see the table on page 27.
```

end example]

17.16.5.46 PRINT

Syntax:

PRINT *field-argument* [*switches*]

Description: An instruction to send the printer-specific control code characters specified by *text* in *field-argument* to the selected printer when the document is printed.

Field Value: None.

Switches: Zero or more of the following *field-specific-switches*.

\p *field-argument-1 field-argument-2*	Allows PostScript strings to be sent to the printer as native PostScript codes. PostScript commands embedded in the document are carried out in the order in which they are inserted. The y-coordinate space used for PostScript commands in as follows: The graphics origin (0,0) is in the lower-left corner of the page, and the positive directions are up and to the right. PostScript drawing instructions take place within a drawing rectangle. The graphics origin is translated to the lower-left corner of the drawing rectangle. *text* in this switch's *field-argument-1* defines the drawing rectangle on which the subsequent PostScript instructions operate. *text* in this switch's *field-argument-2* contains the PostScript instructions.

17.16.5.47 PRINTDATE

Syntax:

```
PRINTDATE [ switches ]
```

Description: Retrieves the date and time on which the document was last printed, as recorded in the LastPrinted element of the Core File Properties part. By default, the Gregorian calendar is used and the *date-and-time-formatting-switch* used is implementation-defined. For a document that has never been printed, the date and time corresponds to `0000-00-00T00:00:00` local time and each text component is XXX.

Field Value: The date and time on which the document was last printed.

Switches: Zero or one *date-and-time-formatting-switch* and zero or one of the following *field-specific-switches*.

\h	Use the Hijri Lunar or Hebrew Lunar calendar, depending on the language specified by the lang element (§17.3.2.20).
\s	Use the Saka Era calendar.

[*Example*: Consider the case in which the LastPrinted element is as follows:

```
<LastPrinted>2006-01-06T19:58:00Z</LastPrinted>
```

and the following fields are updated in a US-English context that is UTC -5:

```
PRINTDATE
PRINTDATE \@ "dddd, MMMM dd, yyyy  HH:mm:ss"
```

the results are:

```
1/6/2006 2:58:00 PM
Friday, January 06, 2006 14:58:00
```

For a document that has never been printed, the result is:

```
0/0/0000 0:00:00 AM
XXX, XXX 00, 0000  00:00:00
```

end example]

17.16.5.48 PRIVATE

Syntax:

```
PRIVATE
```

Description: Provides a private storage area. This field is used to store data for documents converted from other file formats. The field contains data needed for converting a document back to its original file format.

A PRIVATE field is formatted as hidden text.

Field Value: None.

Switches: None.

17.16.5.49 QUOTE

Syntax:

QUOTE *field-argument*

Description: Retrieves the text specified by *text* in *field-argument*. This text can include any other fields except SYMBOL.

Field Value: The specified text.

Switches: One or more of the *date-and-time-formatting-switch*, *general-formatting-switch*, or *date-and-time-formatting-switch*es, depending on the type of *field-argument*.

[*Example*: When the current month is January and the following field is updated:

```
Last month was QUOTE IF DATE \@ "M" = 1 "12" "= DATE \@ "M" - 1"/1/2000 \@
"MMMM".
```

the result is:

```
Last month was December.
```

end example]

17.16.5.50 RD

Syntax:

RD *field-argument* [*switches*]

Description: *field-argument* identifies a file to include when creating a table of contents, a table of authorities, or an index using a TOC (§17.16.5.68), TOA (§17.16.5.67), or INDEX field (§17.16.5.29). RD fields that reference a series of files shall be in the same order as the files in the final document. If the location includes a long file name containing spaces, field-argument shall contain delimiting quotes. A single backslash in the file path shall be preceded directly by a backslash.

For a complex field implementation in XML the optional field-value storage is not needed.

Field Value: None.

Switches: One of the following *field-specific-switches*:

\p	Indicates that the path is relative to the current document.

[*Example*: The following fields inserted into one document create a table of contents that includes entries from the three referenced documents:

```
{ TOC }
{ RD C:\\Manual\\Chapters\\Chapter1.doc }
{ RD C:\\Manual\\Chapters\\Chapter2.doc }
{ RD C:\\Manual\\Chapters\\Chapter3.doc }
```

end example]

17.16.5.51 REF

Syntax:

[REF] *field-argument* [*switches*]

Description: Inserts the text or graphics represented by the bookmark specified by *text* in *field-argument*. The bookmark shall be defined in the current document. Provided the bookmark name is not exactly the same as a field name, the REF prefix can be omitted. If the text marked by the bookmark contains a paragraph mark, the text preceding the REF field assumes the formatting of the paragraph in the bookmark.

Field Value: The specified text or graphics.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`, and zero or one of the following *field-specific-switches*.

`\d` *field-argument*	*text* in this switch's *field-argument* specifies the character sequence that is used to separate sequence numbers and page numbers.
`\f`	Increments footnote, endnote, and annotation numbers that are marked by the bookmark, and inserts the corresponding footnote, endnote, and comment text.
`\h`	Creates a hyperlink to the bookmarked paragraph.
`\n`	For a referenced paragraph, causes the field result to have the entire paragraph number without trailing periods. No information about prior levels is displayed unless it is included as part of the current level.
`\p`	Causes the field result to contain the position relative to the source bookmark using the word "above" or "below." If the REF field appears in the document before the bookmark, it evaluates to "below". If the REF field appears after the bookmark, it evaluates to "above". If the REF field appears within the bookmark, an error is returned. This switch can also be used in conjunction with the `\n`, `\r`, and `\w` switches, in which case, "above" or "below" is appended to the end of the field result.
`\r`	Inserts the entire paragraph number of the bookmarked paragraph in relative context—or relative to its position in the numbering scheme —without trailing periods.
`\t`	Causes the REF field to suppress non-delimiter or non-numerical text when used in conjunction with the `\n`, `\r`, or `\w` switch.

`\w`	Inserts the paragraph number of the bookmarked paragraph in full context from anywhere in the document.

[*Example*: The following field

```
REF _Ref116788778 \r \h
```

makes a hyperlink reference. This kind of field is commonly used, to indicate forward references within a document. *end example*]

17.16.5.52 REVNUM

Syntax:

```
REVNUM
```

Description: Retrieves the document's revision number (which indicates the number of times the document has been saved), as recorded in the Revision element of the Core File Properties part.

Field Value: The document's revision number.

Switches: None.

[*Example*: Consider the case in which the Revision element is as follows:

```
<Revision>11</Revision>
```

and the following field is updated:

```
REVNUM
```

The result is:

```
11
```

end example]

17.16.5.53 SAVEDATE

Syntax:

SAVEDATE [*switches*]

Description: Retrieves the date and time on which the document was last saved, as recorded in the DateModified element of the Core File Properties part. By default, the Gregorian calendar is used and the *date-and-time-formatting-switch* used is implementation-defined. For a document that has never been saved, the date and time corresponds to `0000-00-00T00:00:00` local time and each text component is XXX.

Field Value: The date and time on which the document was last saved.

Switches: Zero or one *date-and-time-formatting-switch* and zero or one of the following *field-specific-switches*.

\h	Use the Hijri Lunar or Hebrew Lunar calendar, depending on the language specified by the lang element (§17.3.2.20).
\s	Use the Saka Era calendar.

[*Example*: Consider the case in which the DateModified element is as follows:

```
<DateModified>2006-01-06T20:15:00Z</DateModified>
```

and the following fields are updated in a US-English context that is UTC -5:

```
SAVEDATE
SAVEDATE \@ "dddd, MMMM dd, yyyy  HH:mm:ss"
```

the results are:

```
1/6/2006 3:15:00 PM
Friday, January 06, 2006 15:15:00
```

For a document that has never been saved, the result is:

```
0/0/0000 0:00:00 AM
XXX, XXX 00, 0000  00:00:00
```

end example]

17.16.5.54 SECTION

Syntax:

SECTION [*switches*]

Description: Retrieves the number of the current section.

Field Value: The number of the current section.

Switches: Zero or more *general-formatting-switch*es.

[*Example*: When the current section number is 19 and the following fields are updated:

```
SECTION
SECTION \* ArabicDash
SECTION \* ALPHABETIC
SECTION \* roman
```

the results are:

```
19
- 19 -
S
xix
```

end example]

17.16.5.55 SECTIONPAGES

Syntax:

`SECTIONPAGES` [*switches*]

Description: Retrieves the number of the current page within the current section.

Field Value: The number of the current page within the current section.

Switches: Zero or more *general-formatting-switch*es.

[*Example*: When the current page number within the current section is 19 and the following fields are updated:

```
SECTIONPAGES
SECTIONPAGES \* ArabicDash
SECTIONPAGES \* ALPHABETIC
SECTIONPAGES \* roman
```

the results are:

```
19
- 19 -
S
xix
```

end example]

17.16.5.56 SEQ

Syntax:

`SEQ` *identifier* [*field-argument*] [*switches*]

Description: Sequentially numbers chapters, tables, figures, and other user-defined lists of items in a document. If an item and its SEQ field are added, deleted, or moved, updating the remaining SEQ fields in the document reflects the new sequence. A SEQ field in a header, footer, annotation, or footnote shall not affect the sequence numbering that results from SEQ fields in the document text.

[*Note*: The `LISTNUM` field also produces automatic numbering and can be a better alternative when creating a complex numbered list. *end note*] *identifier* is the name assigned to the series of items that are to be numbered. [*Example*: *identifier* might be `Equation`, `Figure`, `Table`, or `Thing`, as the user deems appropriate for a caption. *end example*] *identifier* shall start with a Latin letter and shall consist of no more than 40 Latin letters, Arabic digits, and underscores. (See the TOC field (§17.16.5.68) switches `\c` and `\s` for uses of *identifier*.)

text in *field-argument* specifies a bookmark name that refers to an item elsewhere in the document rather than in the current location.

Field Value: The next number in the sequence.

Switches: Zero or one of the *numeric-formatting-switch*es, or zero or more of the following *field-specific-switches*. If no *numeric-formatting-switch* is present, `\* Arabic` is used.

`\c`	Repeats the closest preceding sequence number. [*Note*: This is useful for inserting chapter numbers in headers or footers. *end note*]
`\h`	Hides the field result unless a *general-formatting-switch* is also present.[*Note*: This switch can be used to refer to a SEQ field in a cross-reference without printing the number. *end note*]
`\n`	Inserts the next sequence number for the specified item. This is the default.
`\r` *field-argument*	Resets the sequence number to the integer number specified by *text* in this switch's *field-argument*.
`\s` *field-argument*	Resets the sequence number to the built-in (integer) heading level specified by *text* in this switch's *field-argument*.

[*Example*: When the following fields are updated:

```
SEQ Figure
SEQ Figure \* roman
SEQ Figure \n
SEQ Figure \c : …
SEQ Figure \h : …
SEQ Figure
SEQ Figure \r 1
SEQ Figure
```

the results are:

```
Figure 1
Figure ii
Figure 3
Figure 3: …
Figure : …
Figure 5
Figure 10
Figure 11
```

end example]

17.16.5.57 SET

Syntax:

`SET` *field-argument-1 field-argument-2*

field-argument-1:
field-argument

field-argument-2:
field-argument

Description: Defines the bookmark name specified by *field-argument-1* to represent the information specified by *field-argument-2*.

Field Value: None.

Switches: None.

[*Example*: Consider the following:

```
SET EnteredBy "Paul Smith"
SET UnitCost 25.00
SET Quantity FILLIN "Enter number of items ordered:"
SET SalesTax 10%
SET TotalCost = (UnitCost * Quantity) + ((UnitCost * Quantity) * SalesTax)
Total cost: TotalCost \# "$#0.00"
Thank you for your order,
EnteredBy
```

end example]

17.16.5.58 SKIPIF

Syntax:

`SKIPIF` *comparison*

Description: Compares the values designated by the *expressions* in *comparison* using the operator designated by *comparison-operator*. If the comparison is true, SKIPIF cancels the current merge document, moves to the next data record in the data source, and starts a new merge document. If the comparison is false, the current merge document is continued.

[*Note*: *comparison-operator* can be any one of the six relational and equality operators specified for *operator* (§17.16.3.3). *end note*]

If *comparison-operator* is = or <>, the right-hand *expression* operand can contain a question mark (?) to represent any single character, or an asterisk (*) to represent any string of characters. The expression shall be enclosed in quotation marks so that it is compared as a character string.

Field Value: None.

Switches: None.

[*Example*: Inserted into a mail merge main document, the following field examines the contents of the `Order` field for the current data record. If the field contains a number less than 100, no merged document is produced for that data record.

```
SKIPIF MERGEFIELD Order < 100
```

end example]

17.16.5.59 STYLEREF

Syntax:

STYLEREF *field-argument* [*switches*]

Description: Inserts the nearest piece of text prior to this field that is formatted by the style whose name is specified by *text* in *field-argument*. The style can be a paragraph style or a character style.

When this field is used in a header or footer, it results in the first or the last text formatted with the specified style on the current page, allowing for dictionary-style headers or footers.

The location at which a STYLEREF field is inserted determines the direction searched for the style, as follows:

- In document text, by default, the search goes backward from the STYLEREF field. If the style isn't found, the search goes forward from the STYLEREF field.
- In footnotes, annotations, and endnotes, the search goes backward from the footnote, annotation, or endnote reference mark. If the style isn't found, the search goes forward from the reference mark.
- In headers and footers in a printed document, the search is applied to the current page, by default, from top to bottom, for the specified style. If the style isn't found, the search goes from the top of the page to the beginning of the document, and then from the bottom of the page to the end of the document. If the `\l` switch is used, the search goes from the bottom of the page to the beginning and then to the end of the document.

- In headers and footers in an electronic document, the search goes on in the section that contains the STYLEREF field, from the beginning, for the specified style. If the style isn't found, the search goes from the end of the section to the end of the document.

Field Value: The nearest piece of text prior to this field that is formatted by the style whose name is specified by *text* in *field-argument*.

Switches: Zero or more of the following *field-specific-switches*.

`\l`	Inserts the nearest text following the field.
`\n`	Inserts the paragraph number of the referenced paragraph exactly as it appears in the document.
`\p`	Inserts the relative position of the referenced paragraph as being "above" or "below".
`\r`	Inserts the paragraph number of the referenced paragraph exactly in relative context.
`\t`	When used with the `\n`, `\r`, or `\w` switch, causes non-delimiter and non-numerical text to be suppressed.
`\w`	Inserts the paragraph number of the referenced paragraph in full context, from anywhere in the document.

[*Example*: When the following field is inserted in a header, it displays the contents of the first paragraph formatted with the style "Heading 3" on the current page:

On this page: { STYLEREF "Heading 3" }

To print the first and last names that appear on each page in a membership directory, for example, apply a character style called `Last Name` to each member's last name. Then insert the following fields in the header:

{ STYLEREF "Last Name" } — { STYLEREF "Last Name" \l }

end example]

17.16.5.60 SUBJECT

Syntax:

`SUBJECT` [*field-argument*] [*switches*]

Description: Retrieves, and optionally sets, the document's subject, as recorded in the Subject element of the Core File Properties part or, if *field-argument* is present, the subject specified by *text* in *field-argument*. Specifying a *field-argument* shall change Subject to *text*.

Field Value: The document's subject.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Consider the case in which the Title element is as follows:

```
<Subject>A specification for fields</Subject>
```

and the following field is updated:

SUBJECT

The result is:

```
A specification for fields
```

Updating the following field:

```
SUBJECT "A specification for WordprocessingML Fields"
```

causes the Subject element to take on the given value. *end example*]

17.16.5.61 SYMBOL

Syntax:

SYMBOL *field-argument* [*switches*]

Description: Retrieves the character whose code point value is specified in decimal or hexadecimal (by using a leading 0x or 0X) by *text* in *field-argument*. The formatting switches over ride any formatting applied directly to the result.

The XML generated for a complex field implementation shall not have the optional field value stored.

Field Value: The specified character.

Switches: Zero or more of the following *field-specific-switches*.

`\a`	Interprets *text* in *field-argument* as the value of an ANSI character.
`\f` *field-argument*	Interprets *text* in the switch's *field-argument* as the name of the font from which the character whose value is specified by *text* in the field's *field-argument*. By default, the font used is that for the current text run.
`\h`	Inserts the symbol without affecting the line spacing of the paragraph. If large symbols are inserted with this switch, text above the symbol might be overwritten.
`\j`	Interprets *text* in *field-argument* as the value of a Windows-31J character.
`\s` *field-argument*	Interprets *text* in the switch's *field-argument* as the integral font size in points.
`\u`	Interprets *text* in *field-argument* as the value of a Unicode character.

[*Example*: Consider the case in which the following fields are updated:

```
SYMBOL 65
SYMBOL 66 \a
SYMBOL 67 \u
SYMBOL 0x20ac \u
SYMBOL 68
SYMBOL 68 \f Symbol
SYMBOL 40 \f Wingdings \s 24
```

the results are:

A
B
C
€
D
Δ
☎

end example]

17.16.5.62 TA

Syntax:

TA [*switches*]

Description: Defines the text and page number for a table of authorities entry, which is used by a TOA field (§17.16.5.67).

Field Value: None.

Switches: Zero or one of the following *field-specific-switches*.

`\b`	Applies bold formatting to the page number for the entry. If the table of authorities style for the entry already has bold formatting, `\b` removes it.
`\c` *field-argument*	*text* in this switch's *field-argument* specifies the integral entry category, which is a number that corresponds to the order of categories. The number determines how citations are grouped in tables of authorities. If `\c` is omitted, category 1 is the default.
`\i`	Applies italic formatting to the page number for the entry. If the table of authorities' style for the entry already has italic formatting, `\i` removes it.

`\l` *field-argument*	*text* in this switch's *field-argument* defines the long citation for the entry.
`\r` *field-argument*	Inserts as the entry's page number the range of pages marked by the bookmark specified by *text* in this switch's *field-argument*.
`\s` *field-argument*	*text* in this switch's *field-argument* defines the short citation for the entry.

[*Example*: Given the following fields occurring on page 2:

```
TA \l "Hotels v. Leisure Time" \c 2
TA \l "Baldwin v. Alberti, 58 Wn. 2d 243 (1961)"
   \s "Baldwin v. Alberti" \c 1 \b
```

the table of authorities produced by `INDEX \e "`*`tab`*`" \c "1" \z "1033"` is:

Cases

Baldwin v. Alberti, 58 Wn. 2d 243 (1961)........ 2

Statutes

Hotels v. Leisure Time............................. 2

end example]

17.16.5.63 TC

Syntax:

TC *field-argument* [*switches*]

Description: Defines the text and page number for a table of contents (including a table of figures) entry, which is used by a TOC field (§17.16.5.68). The text of the entry is *text* in *field-argument*.

Field Value: None.

Switches: Zero or one of the following *field-specific-switches*.

`\f` *field-argument*	The type of items collected in a particular contents list. Use a unique list type identifier (typically a letter from A-Z) for each type of list. For example, to build a list of illustrations, mark each illustration with a field such as `TC "Illustration 1" \f i`, where `i` indicates only illustration entries. If no list type is specified, the entry is listed in a table of contents.
`\l` *field-argument*	The level of the TC entry. [*Example*: The field `TC "Entering Data" \l 4` marks a level-4 entry, and applies the built-in style

	TOC 4 to that entry in the table of contents. *end example*] If no level is specified, level 1 is assumed.
\n	Omits the page number for the entry.

17.16.5.64 TEMPLATE

Syntax:

TEMPLATE [*switches*]

Description: The file name of the template used by the current document.

Field Value: The file name of the template used by the current document, as specified by the target of the relationship specified by the id attribute of the attachedTemplate element (§17.15.1.6), or the empty string if no such element exists.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`, and zero or one of the following *field-specific-switches*.

\p	Include the full file path name.

[*Example*: Consider the case in which the following fields are updated:

```
TEMPLATE \* Upper
TEMPLATE \p
```

the results might be:

```
NORMAL.DOTM
C:\Templates\Normal.dotm
```

end example]

17.16.5.65 TIME

Syntax:

TIME [*switches*]

Description: The current date and time. As specified in Date and Time Formatting (§17.16.4.1), if no *date-and-time-formatting-switch* is present, a date or time result is formatted in an implementation-defined manner.

Field Value: The current date and time.

Switches: Zero or one *date-and-time-formatting-switch*.

[*Example*: Consider the case in which the following fields are updated on the given date and time:

```
TIME
TIME \@ "dddd, MMMM dd, yyyy  HH:mm:ss"
```

the results are:

```
1:59 PM
Friday, January 06, 2006  13:59:42
```

end example]

[*Note*: For some *date-and-time-formatting-switch*es, the DATE (§17.16.5.13) and TIME fields can produce the same result. *end note*]

17.16.5.66 TITLE

Syntax:

TITLE [*field-argument*] [*switches*]

Description: Retrieves, and optionally sets, the document's title, as recorded in the Title element of the Core File Properties part or, if *field-argument* is present, the name specified by *text* in *field-argument*. Specifying a *field-argument* shall change Title to *text*.

Field Value: The document's title.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Consider the case in which the Title element is as follows:

```
<Title>My Life's Story</Title>
```

and the following field is updated:

TITLE

The result is:

```
My Life's Story
```

Updating the following field:

```
TITLE "My Life, the Fantasy" \* Upper
```

causes the Title element to take on the value `My Life, the Fantasy` while the result is:

MY LIFE THE FANTASY

end example]

17.16.5.67 TOA

Syntax:

©ISO/IEC 2012 – All rights reserved

TOA [*switches*]

Description: Builds a table of authorities (that is, a list of the references in a legal document, such as references to cases, statutes, and rules, along with the numbers of the pages on which the references appear) using the entries specified by TA fields (§17.16.5.62).

Field Value: The table of authorities.

Switches: Zero or more of the following *field-specific-switches*.

`\b` *field-argument*	Includes entries only from the portion of the document marked by the bookmark specified by *text* in this switch's *field-argument*.
`\c` *field-argument*	*Includes the entries whose integral category is that specified by text* in this switch's *field-argument*.
`\d` *field-argument*	Used in conjunction with `\s` to specify the character sequence that separates the sequence numbers and page numbers. If `\d` is omitted, a hyphen (-) is used.
`\e` *field-argument*	*text* in this switch's *field-argument* specifies the character sequence that separates a table of authorities entry and its page number. If `\e` is not specified, a tab stop with leader dots is used.
`\f`	Removes the formatting of the entry text in the document from the entry in the table of authorities.
`\g` *field-argument*	*text* in this switch's *field-argument* specifies the character sequence that separates the pages in a page range. If `\g` is omitted, an en dash (–) is used.
`\h`	Includes the category heading for the entries in a table of authorities.
`\l` *field-argument*	*text* in this switch's *field-argument* specifies the character sequence that separates multiple page references. If `\l` is omitted, a comma (,) and space are used.
`\p`	Replaces five or more different page references to the same authority with "passim", which is used to indicate that a word or passage occurs frequently in the work cited.
`\s` *field-argument*	Includes a case or section number before the page number. The entry shall be numbered with a SEQ field (§17.16.5.56), and *text* in this switch's *field-argument* shall match the identifier in the SEQ field.

[*Example*: See TA (§17.16.5.62). *end example*]

17.16.5.68 TOC

Syntax:

TOC [*switches*]

Description: Builds a table of contents (which can also be a table of figures) using the entries specified by TC fields (§17.16.5.63), their heading levels, and specified styles, and inserts that table at this place in the document. Each table entry is a separate paragraph.

Field Value: The table of contents.

Switches: Zero or more of the following *field-specific-switches*.

`\a` *field-argument*	Includes captioned items, but omits caption labels and numbers. The identifier designated by *text* in this switch's *field-argument* corresponds to the caption label. Use `\c` to build a table of captions with labels and numbers.
`\b` *field-argument*	Includes entries only from the portion of the document marked by the bookmark named by *text* in this switch's *field-argument*.
`\c` *field-argument*	Includes figures, tables, charts, and other items that are numbered by a SEQ field (§17.16.5.56). The sequence identifier designated by *text* in this switch's *field-argument*, which corresponds to the caption label, shall match the identifier in the corresponding SEQ field.
`\d` *field-argument*	When used with `\s`, the *text* in this switch's *field-argument* defines the separator between sequence and page numbers. The default separator is a hyphen (-).
`\f` *field-argument*	Includes only those TC fields whose identifier exactly matches the *text* in this switch's *field-argument* (which is typically a letter).
`\h`	Makes the table of contents entries hyperlinks.
`\l` *field-argument*	Includes TC fields that assign entries to one of the levels specified by *text* in this switch's *field-argument* as a range having the form *startLevel*`-`*endLevel*, where *startLevel* and *endLevel* are integers, and *startLevel* has a value equal-to or less-than *endLevel*. TC fields that assign entries to lower levels are skipped.
`\n` *field-argument*	Without *field-argument*, omits page numbers from the table of contents. Page numbers are omitted from all levels unless a range of entry levels is specified by *text* in this switch's *field-argument*. A range is specified as for `\l`.
`\o` *field-argument*	Uses paragraphs formatted with all or the specified range of built-in heading styles. Headings in a style range are specified by *text* in this switch's *field-argument* using the notation specified as for `\l`, where each integer corresponds to the style with a style ID of HeadingX (e.g. `1` corresponds to `Heading1`). If no heading range is specified, all heading levels used in the document are listed.
`\p` *field-argument*	*text* in this switch's *field-argument* specifies a sequence of characters that separate an entry and its page number. The default is a tab with leader dots.
`\s` *field-argument*	For entries numbered with a SEQ field (§17.16.5.56), adds a prefix

	to the page number. The prefix depends on the type of entry. *text* in this switch's *field-argument* shall match the identifier in the SEQ field.
`\t` *field-argument*	Uses paragraphs formatted with styles other than the built-in heading styles. *text* in this switch's *field-argument* specifies those styles as a set of comma-separated doublets, with each doublet being a comma-separated set of style name and table of content level. `\t` can be combined with `\o`.
`\u`	Uses the applied paragraph outline level.
`\w`	Preserves tab entries within table entries.
`\x`	Preserves newline characters within table entries.
`\z`	Hides tab leader and page numbers in Web layout view.

[*Example*: The index produced using the corresponding set of index entries and the field TOC `\o "3-3" \h \z \t "Heading 1,1,Heading 2,2,Appendix 1,1,Appendix 2,2,Unnumbered Heading,1"` is:

1. Introduction ………………………………. 1

2. Syntax……………………………………….. 2

3. XML representation………………………. 4

4. Formulas and expressions…………………. 6

4.1 Constants…………………………………. 6

4.2 Bookmarks………………………………… 6

4.3 Operators…………………………………. 6

4.4 Functions…………………………………. 7

4.5 Table cell references………………………. 8

…

Annex A. Index………………………………. 12

end example]

17.16.5.69 USERADDRESS

Syntax:

USERADDRESS [*field-argument*] [*switches*]

Description: Retrieves the current user's postal address or, if *field-argument* is present, the address specified by *text* in *field-argument*. Specifying a field-argument shall not change the address of the current user.

Field Value: If a particular user's context exists, the current user's postal address; otherwise, an empty string.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Given the current user's address, the following fields:

```
USERADDRESS
USERADDRESS "10 Top Secret Lane, Chiswick" \* Upper
```

produce results of:

```
114 Rue du Rhône
CH-1204 Geneva
Switzerland

10 TOP SECRET LANE, CHISWICK
```

end example]

17.16.5.70 USERINITIALS

Syntax:

USERINITIALS [*field-argument*] [*switches*]

Description: Retrieves the current user's initials or, if *field-argument* is present, the initials specified by *text* in *field-argument*. Specifying a field-argument shall not change the initials of the current user.

Field Value: If a particular user's context exists, the current user's initials; otherwise, an empty string.

Switches: One of the following *general-formatting-switch*es: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Given a current user with initials "DW", the following fields:

```
USERINITIALS \* Lower
USERINITIALS "JaJ"
USERINITIALS "jaj" \* Upper
```

produce results of:

```
dw
JaJ
JAJ
```

end example]

17.16.5.71 USERNAME

Syntax:

USERNAME [*field-argument*] [*switches*]

Description: Retrieves the current user's name or, if *field-argument* is present, the name specified by *text* in *field-argument*. Specifying a field-argument shall not change the name of the current user.

Field Value: If a particular user's context exists, the current user's name; otherwise, an empty string.

Switches: One of the following *general-formatting-switches*: `\* Caps`, `\* FirstCap`, `\* Lower`, or `\* Upper`.

[*Example*: Given a current user of "David Williams", the following fields:

```
USERNAME \* Lower
USERNAME "John Jones"
USERNAME "Mary Smith" \* Upper
```

produce results of:

```
david williams
John Jones
MARY SMITH
```

end example]

17.16.5.72 XE

Syntax:

XE *field-argument* [*switches*]

Description: Defines the text and page number for an index entry, which is used by an INDEX field (§17.16.5.29). The text of the entry is *text* in *field-argument*. To indicate a subentry, the main entry text and the subentry text shall be separated by a colon (:). Subentries beyond one level are permitted.

Field Value: None.

Switches: Zero or one of the following *field-specific-switches*.

`\b`	Applies bold formatting to the entry's page number. However, if the index style for that entry is already bold, this switch removes that formatting for that entry.
`\f` *field-argument*	The *text* in this switch's *field-argument* defines an index entry type. If an INDEX field has the same `\f` switch and *field-argument*, this entry is included in the resulting index; otherwise, it is excluded.

`\i`	Applies italic formatting to the entry's page number. However, if the index style for that entry is already italic, this switch removes that formatting for that entry.
`\r` *field-argument*	Instead of the entry's page number, uses the range of pages marked by the bookmark specified by *text* in this switch's *field-argument*.
`\t` *field-argument*	Uses *text* from *field-argument* in place of a page number. [*Note*: Useful for "See …" or "See also …" entries. *end note*]
`\y` *field-argument*	Specifies that the *text* from *field-argument* defines the yomi (first phonetic character for sorting indexes) for the index entry.

[*Example*: Given the following fields spread over a series of pages, and a multi-page bookmark called `OOXMLPageRange`:

```
XE "Office Open XML" \b
XE "syntax" \f "Introduction"
XE "behavior:implementation-defined" \b
XE "Office Open XML" \i
XE "behavior:implementation-defined:documenting" \b
XE "grammar" \f "Introduction" \b
XE "Office Open XML"
XE "item: package-relationship" \t "See package-relationship item"
XE "XML" \r OOXMLPageRange
XE "grammar" \f "Introduction"
XE "production" \f "Introduction"
```

the index produced by `INDEX \e "`*`tab`*`" \c "1" \z "1033"` is:

behavior

implementation-defined …………………………………… **2**

documenting ……………………………………………. **3**

item

package-relationship ………… See package-relationship item

Office Open XML ……………………………………… **2**, *3*, 4

XML ……………………………………………………… 1–4

and that produced by `INDEX \f "Introduction" \e "`*`tab`*`" \c "1" \z "1033"` is:

grammar …………………………………………………. **3**, 5

production ……………………………………………………………… 5

syntax ………………………………………………………..…. 2

end example]

17.16.6 calcOnExit (Recalculate Fields When Current Field Is Modified)

This element specifies that the current contents of all fields within the current WordprocessingML document shall be recalculated from their field codes when the contents of the parent form field are modified. [*Note*: It is at the discretion of an application to determine the scope of a single modification, for example, when the user moves the insertion point in a user interface, or after each keystroke, etc. *end note*]

If this element is omitted, then modification of the contents of the current field shall not result in all fields in the current document being recalculated.

[*Example*: Consider the following WordprocessingML fragment for the contents of two fields in a document:

```
<w:bookmarkStart w:name="Text1" … />
<w:r>
  <w:fldChar w:fldCharType="begin">
    <w:ffData>
      <w:calcOnExit/>
      …
    </w:ffData>
  </w:fldChar>
  </w:r>
  <w:r>
    <w:instrText> FORMTEXT </w:instrText>
  </w:r>
  <w:r>
    <w:fldChar w:fldCharType="separate"/>
  </w:r>
  <w:r>
    <w:t>1</w:t>
  </w:r>
  <w:r>
    <w:fldChar w:fldCharType="end"/>
  </w:r>
  <w:bookmarkEnd … />
  <w:fldSimple w:instr="=Text1+10">
    <w:r>
      <w:t>11</w:t>
    </w:r>
  </w:fldSimple>
```

The first field above (the text form field) has a current value of 1, but also has the calcOnExit element present (therefore inheriting its default attribute value of true). This means that if the value of this form field is changed to 10, that all fields in the document must automatically be updated, resulting in the second field's value being automatically changed to 20. *end example*]

This element’s content model is defined by the common boolean property definition in §17.17.4.

17.16.7 checkBox (Checkbox Form Field Properties)

This element specifies a set of properties which shall be associated with the parent FORMCHECKBOX checkbox form field (§17.16.5.20) within the document.

If the parent form field is not a checkbox (i.e. its field code does not have a value of FORMCHECKBOX), then these properties can be ignored.

[*Example*: Consider the following WordprocessingML fragment for the properties of a checkbox form field:

```
<w:ffData>
  <w:checkBox>
    <w:size w:val="20" />
    <w:checked w:val="true" />
  </w:checkBox>
</w:ffData>
```

The checkBox element specifies that it contains a set of properties for the parent checkbox form field. In this case, these properties specify that the size of the checkbox must be exactly 10 points via the size element (§17.16.29), and that the current state of the checkbox must be checked via the checked element (§17.16.8). *end example*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_FFCheckBox) is located in §A.1. *end note*]

17.16.8 checked (Checkbox Form Field State)

This element specifies the current state for a checkbox form field. This value shall be used to specify the current value for a checkbox as explicitly chosen for that checkbox, as opposed its default value, which is specified using the default element (§17.16.12).

If this element is omitted, then the parent form field checkbox has no state, and its state shall be determined based on the value of the default element in the checkbox form field properties.

[*Example*: Consider the following WordprocessingML fragment for the properties of a checkbox form field:

```
<w:ffData>
  <w:checkBox>
    …
    <w:checked w:val="true" />
  </w:checkBox>
</w:ffData>
```

The checked element specifies that the current state of the checkbox is checked (via an attribute value of true). *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.16.9 ddList (Drop-Down List Form Field Properties)

This element specifies a set of properties which shall be associated with the parent FORMDROPDOWN drop-down list form field (§17.16.5.21) within the document.

If the parent form field is not a drop-down list (i.e. its field code does not have a value of FORMDROPDOWN), then these properties can be ignored.

[*Example*: Consider the following WordprocessingML fragment for the properties of a drop-down list form field:

```
<w:ffData>
  <w:ddList>
    <w:listEntry w:val="One" />
    <w:listEntry w:val="Two" />
    <w:listEntry w:val="Three" />
  </w:ddList>
</w:ffData>
```

The ddList element specifies that it contains a set of properties for the parent drop-down list form field. In this case, these properties specify that the drop-down list must contain three entries of One, Two, and Three via the listEntry elements (§17.16.25). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_FFDDList) is located in §A.1. *end note*]

17.16.10 default (Default Text Box Form Field String)

This element specifies the default string for the parent text box form field. This string is the content which shall be displayed in the document story within this form field if its current run contents are empty (i.e. there is not actual content within the text box). If the type (§17.16.34) of the current form field is calculation, then this string shall hold the calculation to be performed.

If this element is omitted, then the current text box form field shall not have a default value.

[*Example*: Consider the following WordprocessingML fragment for a text box form field:

```
<w:fldChar w:fldCharType="begin">
  <w:ffData>
    <w:textInput>
      <w:default w:val="No content."/>
    </w:textInput>
  </w:ffData>
</w:fldChar>
```

The default element specifies the default value of the text box form field to be `No content.`Since the form field does not contain any value, this is the content which must be displayed when the contents of the form field are displayed by an application. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.16.11 default (Default Drop-Down List Item Index)

This element specifies the zero-based index of the default entry for the parent drop-down list form field. This index value is the value within the drop-down list which shall be displayed in the document story within this form field if no element is selected (i.e. the result element (§17.16.28) is omitted).

If this element is omitted, then the current drop-down list form field shall have a default value of 0 (its first entry). If the attribute value references an index value which does not exist (i.e. a negative number or a number that exceeds the number of items in the drop-down list), then this value can be ignored and the current drop-down list form field shall have a default value of 0 (its first entry).

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
<w:fldChar w:fldCharType="begin">
  <w:ffData>
    <w:ddList>
      <w:default w:val="1" />
      <w:listEntry w:val="One" />
      <w:listEntry w:val="Two" />
      <w:listEntry w:val="Three" />
    </w:ddList>
  </w:ffData>
</w:fldChar>
```

The default element specifies the index of the default value of the drop-down list form field to be 1.Since the form field does not contain a result element, this is the index of the content which must be displayed when the contents of the form field are displayed by an application. In this case, the resulting default value text is Two. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.16.12 default (Default Checkbox Form Field State)

This element specifies the default checkbox state for the parent checkbox form field. This value determines the checkbox state when its current run contents are empty (i.e. there is not actual content within the drop-down list).

If this element is omitted, then the current checkbox form field shall have a default value of 0 (unchecked).

[*Example*: Consider the following WordprocessingML fragment for a checkbox list form field:

```
<w:fldChar w:fldCharType="begin">
  <w:ffData>
    <w:checkBox>
      <w:default w:val="true" />
    </w:checkBox>
  </w:ffData>
</w:fldChar>
```

The default element specifies the default state of the checkbox form field to be true.Since the form field does not contain any run content, this is the state which must be displayed when the contents of the form field are displayed by an application. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.16.13 delInstrText (Deleted Field Code)

This element specifies that this run contains deleted field codes (§17.16.5) within a complex field in the document. The delInstrText element shall be used for all runs containing field codes which are part of a region of text that is contained in a deleted region using the del element (§17.13.5.14).

If this element is not contained within a del element, then the document is non-conformant. If this element is contained within a run which is not part of a complex field's field codes, then it should be handled as regular deleted text.

[*Example*: Consider a complex field within a WordprocessingML document which was changed from a text box form field to a checkbox form field with revision tracking enabled. This field would therefore be represented as follows:

```
  <w:fldChar w:fldCharType="begin" />
  <w:ins>
    <w:r>
      <w:instrText>FORMCHECKBOX</w:instrText>
    </w:r>
  </w:ins>
  <w:del>
    <w:r>
      <w:delInstrText>FORMFIELDTEXT</w:delInstrText>
    </w:r>
  </w:del>
  <w:fldChar w:fldCharType="separate" />
  …
  <w:fldChar w:fldCharType="end" />
```

The deleted field code is contained in a delInstrText node, while the inserted (and current) field code is contained in an instrText node. *end example*]

Attributes	Description

xml:space (Content Contains Significant Whitespace)
Namespace: http://www.w3.org/XML/1998/namespace
	Specifies how white space should be handled for the contents of this element using the W3C space preservation rules.
	[*Example*: Consider the following run contained within a WordprocessingML document:
	`<w:r>`
	`  <w:t>   significant whitespace   </w:t>`
	`</w:r>`
	Although there are three spaces on each side of the text content in the run, that whitespace has not been specifically marked as significant, therefore it is subject to the space preservation rules currently specified in that run's scope. *end example*]
	The possible values for this attribute are defined by §2.10 of the XML 1.0 specification.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Text) is located in §A.1. *end note*]

17.16.14 enabled (Form Field Enabled)

This element specifies whether the parent form field shall behave as though it is enabled or disabled when it is displayed in the document. This setting shall have no effect on the behavior of this form field unless the document's Settings part specifies that the documentProtection element for the current document is in a state allowing the filling in of form fields.

If this element is omitted, then the parent form field shall be in its enabled state when the document settings specify that the document allows the filling-in of form fields.

[*Example*: Consider the following WordprocessingML fragment for a text box form field:

```
<w:r>
  <w:fldChar w:fldCharType="begin" >
  <w:ffData>
    <w:enabled w:val="false"/>
    <w:textInput>
      …
    </w:textInput>
  </w:ffData>
</w:fldChar
</w:r>
<w:r>
  <w:instrText> FORMTEXT </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r>
  <w:t>1</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

The enabled element specifies that the state of the current text box form field is disabled; therefore this text box must not be editable within the current document even when the state of the documentProtection element specifically allows the editing of form fields. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.16.15 entryMacro (Script Function to Execute on Form Field Entry)

This element specifies a subroutine in a scripting language which should be executed when the when the run contents of the parent form field are entered. The language and location of this subroutine can be determined using any method desired by an application. [*Note*: It is at the discretion of an application to determine the scope and timing of "entering" a form field, for example, when the user moves the insertion point in a user interface or upon each operation by an application without a user interface, etc. *end note*]

If this element is omitted, then no subroutine shall be associated with entering the run contents of the parent form field. If this element specifies a macro which cannot be located or is not supported by an application, then its value can be ignored, but shall not be lost upon resaving the file.

[*Example*: Consider the following WordprocessingML fragment for the properties of a checkbox form field:

```
<w:ffData>
  <w:entryMacro w:val="TestEntryFunction" />
  <w:checkBox>
    …
  </w:checkBox>
</w:ffData>
```

The entryMacro element specifies that any application which processes this file should attempt to locate and execute a scripting subroutine called `TestEntryFunction` when the contents of the checkbox are entered. If this subroutine cannot be located or executed, then this setting is silently ignored. *end example*]

Attributes	Description

val (Name of Script Function)	Specifies the name of a single scripting subroutine which shall be associated with the parent element. Its use is specifies based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment for the properties of a form field:
	`<w:ffData>`
	`  <w:exitMacro w:val="HelloWorld" />`
	`</w:ffData>`
	The val attribute specifies that a script function called `HelloWorld` must be used in the context of the parent element; in this case, to execute when the field is exited. *end example*]
	The possible values for this attribute are defined by the ST_MacroName simple type (§17.18.51).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MacroName) is located in §A.1. *end note*]

17.16.16 exitMacro (Script Function to Execute on Form Field Exit)

This element specifies a subroutine in a scripting language which should be executed when the when the run contents of the parent form field are exited. The language and location of this subroutine can be determined using any method desired by an application. [*Note*: It is at the discretion of an application to determine the scope and timing of "exiting" a form field, for example, when the user moves the insertion point in a user interface or upon each operation by an application without a user interface, etc. *end note*]

If this element is omitted, then no subroutine shall be associated with exiting the run contents of the parent form field. If this element specifies a macro which cannot be located or is not supported by an application, then its value can be ignored, but shall not be lost upon resaving the file.

[*Example*: Consider the following WordprocessingML fragment for the properties of a form field:

```
<w:ffData>
  <w:exitMacro w:val="TestExitFunction" />
</w:ffData>
```

The exitMacro element specifies that any application which processes this file should attempt to locate and execute a scripting subroutine called `TestExitFunction` when the contents of the form field are exited. If this subroutine cannot be located or executed, then this setting is silently ignored. *end example*]

Attributes	Description

val (Name of Script Function)	Specifies the name of a single scripting subroutine which shall be associated with the parent element. Its use is specifies based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment for the properties of a form field:
	`<w:ffData>`
	`  <w:exitMacro w:val="HelloWorld" />`
	`</w:ffData>`
	The val attribute specifies that a script function called `HelloWorld` must be used in the context of the parent element; in this case, to execute when the field is exited. *end example*]
	The possible values for this attribute are defined by the ST_MacroName simple type (§17.18.51).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MacroName) is located in §A.1. *end note*]

17.16.17 ffData (Form Field Properties)

This element specifies a set of properties which shall be associated with the parent form field within the document. This form field can be of any of the following types (with the associated field codes in parentheses):

- Checkbox (FORMCHECKBOX)
- Drop-down List (FORMDROPDOWN)
- Text box (FORMTEXT)

If this element is present and the field codes for the document do not specify a form field of one of these types, then the document shall be considered non-conformant.

If this element is omitted, then the properties associated with the parent form field shall be determined based on their default values.

[*Example*: Consider the following WordprocessingML fragment for a text box form field:

```
<w:r>
  <w:fldChar w:fldCharType="begin">
    <w:ffData>
      <w:name w:val="TextTextBox" />
      <w:enabled w:val="false"/>
      <w:textInput>
        <w:maxLength w:val="10" />
      </w:textInput>
    </w:ffData>
  </w:fldChar>
  </w:r>
  <w:r>
    <w:instrText> FORMTEXT </w:instrText>
  </w:r>
  <w:r>
    <w:fldChar w:fldCharType="separate"/>
  </w:r>
  <w:r>
    <w:t>1</w:t>
  </w:r>
  <w:r>
    <w:fldChar w:fldCharType="end"/>
  </w:r>
```

The ffData element specifies the set of properties for this text box form field; in this example, a form field name of `TestTextBox` via the name element (§17.16.27), a disabled state via the enabled element (§17.16.14), and a maximum character length of `10` Unicode scalar values via the maxLength element (§17.16.26). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_FFData) is located in §A.1. *end note*]

17.16.18 fldChar (Complex Field Character)

This element specifies the presence of a complex field character at the current location in the parent run. A *complex field character* is a special character which delimits the start and end of a complex field or separates its field codes from its current field result.

A complex field is defined via the use of the two required complex field characters: a *start character*, which specifies the beginning of a complex field within the document content; and an *end character*, which specifies the end of a complex field. This syntax allows multiple fields to be embedded (or "nested") within each other in a document.

As well, because a complex field can specify both its field codes and its current result within the document, these two items are separated by the optional *separator character*, which defines the end of the field codes and the beginning of the field contents. The omission of this character shall be used to specify that the contents of the field are entirely field codes (i.e. the field has no result).

[*Example*: Consider the following complex field definition within a WordprocessingML document:

```
<w:r>
  <w:fldChar w:fldCharType="begin" />
</w:r>
<w:r>
  <w:instrText>AUTHOR</w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate" />
</w:r>
<w:r>
  <w:t>Rex Jaeschke</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end" />
</w:r>
```

The three fldChar elements specify:

- The beginning of the field, using the type attribute value of `start`
- The separator between the field codes and the current field results, using the type attribute value of `separate`
- The end of the field, using the type attribute value of end

end example]

If a complex field is not closed before the end of a document story, then no field shall be generated and each individual run shall be processed as if the field characters did not exist (i.e. the contents of all field code run content shall not be displayed, and the field results shall be displayed as literal text).

[*Example*: Consider the following WordprocessingML document:

```
<w:body>
  <w:p>
    <w:r>
      <w:fldChar w:fldCharType="begin" />
    </w:r>
    <w:r>
      <w:instrText>AUTHOR</w:instrText>
    </w:r>
    <w:r>
      <w:fldChar w:fldCharType="separate" />
    </w:r>
    <w:r>
      <w:t>Rex Jaeschke</w:t>
    </w:r>
  </w:p>
</w:body>
```

The complex field is technically incorrect since no end character exists in the main document story. The resulting content must be interpreted as though no field characters exist, resulting in only the literal text `Rex Jaeschke` being displayed in the document. *end example*]

Attributes	Description

dirty (Field Result Invalidated)	Specifies that this field has been flagged by an application to indicate that its current results are no longer correct (stale) due to other modifications made to the document, and these contents should be updated before they are displayed if this functionality is supported by the next processing application.
	[*Rationale*: This functionality allows applications with limited subsets of the full functionality of ISO/IEC 29500 to process Word Open XML documents without needing to understand and update all fields based on the semantics for their field codes.
	For example, an application can add a new paragraph and flag the table of contents as dirty, without needing to understand anything about how to recalculate that field's content. *end rationale*]
	If this attribute is omitted, then its value shall be assumed to be `false`. If the type of the current field character is not `start`, then his setting can be ignored.
	[*Example*: Consider the following WordprocessingML for a complex field:
	`…`
	`<w:r>`
	`  <w:fldChar w:fldCharType="begin" w:dirty="true"/>`
	`</w:r>`
	`<w:r>`
	`  <w:instrText>TOC /l 1-3</w:instrText>`

Attributes	Description

	`  </w:r>`
	`  <w:r>`
	`    <w:fldChar w:fldCharType="separate"/>`
	`  </w:r>`
	`  …`
	
	The dirty attribute value of `true` specifies that the contents of this field are no longer current based on the contents of the document, and should be recalculated whenever an application with this functionality reads the document. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

fldCharType (Field Character Type)	Specifies the type of the current complex field character in the document.
	
	[*Example*: Consider the following WordprocessingML for a complex field character:
	
	`  …`
	`  <w:fldChar w:fldCharType="separate" />`
	`  …`
	
	The type attribute value of `separate` specifies that this is a complex field separator character; therefore it is being used to separate the field codes from the field contents in a complex field. *end example*]
	
	The possible values for this attribute are defined by the ST_FldCharType simple type (§17.18.29).

fldLock (Field Should Not Be Recalculated)	Specifies that the parent complex field shall not have its field result recalculated, even if an application attempts to recalculate the results of all fields in the document or a recalculation is explicitly requested.
	
	If this attribute is omitted, then its value shall be assumed to be `false`. If the type of the current field character is not `start`, then his setting can be ignored.
	
	[*Example*: Consider the following WordprocessingML for a complex field:
	
	`  <w:r>`
	`    <w:fldChar w:fldCharType="begin" w:fldLock="true"/>`
	`  </w:r>`
	`  …`
	`  <w:r>`
	`    <w:fldChar w:fldCharType="separate"/>`
	`  </w:r>`
	`  <w:r>`
	`    <w:t>field result</w:t>`
	`  </w:r>`
	`  <w:r>`
	`    <w:fldChar w:fldCharType="end" />`

Attributes	Description
	`</w:r>` The fldLock attribute value of `true` specifies that the contents of this field must remain `field result` regardless of the actual result of the current field codes. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FldChar) is located in §A.1. *end note*]

17.16.19 fldSimple (Simple Field)

This element specifies the presence of a simple field at the current location in the document. The semantics of this field are defined via its field codes (§17.16.5).

[*Example*: Consider the following WordprocessingML fragment for a simple field:

```
<w:fldSimple w:instr="FILENAME">
  <w:r>
    <w:t>Example Document.docx</w:t>
  </w:r>
</w:fldSimple>
```

The fldSimple element defines a `FILENAME` field (§17.16.5.17) using the simple field syntax. The current field result for the field is `Example Document.docx`. *end example*]

Attributes	Description
dirty (Field Result Invalidated)	Specifies that this field has been flagged by an application to indicate that its current results are no longer correct (stale) due to other modifications made to the document, and these contents should be updated before they are displayed if this functionality is supported by the next processing application. [*Rationale*: This functionality allows applications with limited subsets of the full functionality of ISO/IEC 29500 to process Word Open XML documents without needing to understand and update all fields based on the semantics for their field codes. For example, an application can add a new paragraph and flag the table of contents as dirty, without needing to understand anything about how to recalculate that field's content. *end rationale*] If this attribute is omitted, then its value shall be assumed to be `false`. [*Example*: Consider the following WordprocessingML for a simple field: `<w:fldSimple w:instr="AUTHOR" w:dirty="true"/>`

Attributes	Description
	The dirty attribute value of `true` specifies that the contents of this field are no longer current based on the contents of the document, and should be recalculated whenever an application with this functionality reads the document. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

fldLock (Field Should Not Be Recalculated)	Specifies that the parent field shall not have its field result recalculated, even if an application attempts to recalculate the results of all fields in the document or a recalculation is explicitly requested.
	If this attribute is omitted, then its value shall be assumed to be `false`.
	[*Example*: Consider the following WordprocessingML for a simple field:
	`<w:fldSimple w:instr="AUTHOR" w:fldLock="true">`
	`  <w:r>`
	`    <w:t>Rex Jaeschke</w:t>`
	`  </w:r>`
	`</w:fldSimple>`
	The fldLock attribute value of `true` specifies that the contents of this field must remain `Rex Jaeschke` regardless of the actual result of the current field codes. *end example*]
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

instr (Field Codes)	Specifies the field codes for the simple field. The possible field codes are defined in §17.16.5.
	[*Example*: Consider the following WordprocessingML for a simple field:
	`<w:fldSimple w:instr="AUTHOR" w:fldLock="true">`
	`  <w:r>`
	`    <w:t>Rex Jaeschke</w:t>`
	`  </w:r>`
	`</w:fldSimple>`
	The instr attribute specifies the field codes for this simple field to be `AUTHOR`. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SimpleField) is located in §A.1. *end note*]

17.16.20 format (Text Box Form Field Formatting)

This element specifies the field formatting which shall be applied to the contents of the parent form field whenever those contents are modified. The type of formatting which is applied to the field depends on the value of its type element (§17.16.34), as follows:

- When the type is equal to `currentDate`, `currentTime`, or `date`, a date formatting string using the syntax defined in §17.16.4.1
- When the type is equal to `calculated` or `number`, a number formatting string using the syntax defined in §17.16.4.2
- When the type is equal to regular, a text formatting string defined as follows:

Argument	Description
`Uppercase`	All letters are uppercase. [*Example*: `Mary Smith` results in `MARY SMITH`. *end example*]
`Lowercase`	All letters are lowercase. [*Example*: `Mary Smith` results in `mary smith`. *end example*]
`First capital`	Capitalizes the first letter of the first word. [*Example*: `Mary Smith` results in `Mary smith`. *end example*]
`Title case`	Capitalizes the first letter of each word. [*Example*: `Mary Smith` results in `Mary Smith`. *end example*]

[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:

```
<w:ffData>
  <w:textInput>
    <w:type w:val="number" />
    <w:maxLength w:val="4" />
    <w:format w:val="0.00" />
  </w:textInput>
</w:ffData>
```

The format element specifies the field formatting which is applied to the input to the field (in this case, a grouping of number formatting picture items as the type element specifies a value of `number`). If a value of 8 was entered into this field, the formatted result would be `8.00`. *end example*]

Attributes	Description
val (String Value)	Specifies that its contents contain a string. The contents of this string are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment:

Attributes	Description

	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`
	`  …`
	`</w:sdtPr>`
	In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.16.21 helpText (Associated Help Text)

This element specifies optional help text which shall be associated with the parent form field. The method or user interface by which this help text can be surfaced is not defined by ISO/IEC 29500.

If this element is omitted, then no help text shall be associated with the current form field.

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
<w:r>
  <w:fldChar w:fldCharType="begin">
    <w:ffData>
      <w:helpText w:type="text" w:val="Example help text."/>
      <w:ddList>
      …
      </w:ddList>
    </w:ffData>
  </w:fldChar>
</w:r>
<w:r>
  <w:instrText> FORMDROPDOWN </w:instrText>
</w:r>
```

```
  <w:r>
    <w:fldChar w:fldCharType="end"/>
  </w:r>
```

The helpText element specifies the help text for the parent form field - in this case, literal help text consisting of the string `Example help text.` *end example*]

Attributes	Description

type (Help Text Type)	Specifies the type of help text which is specified by this element, defined by the simple type below.
	If this attribute is omitted, then its value shall be assumed to be `text`.
	[*Example*: Consider the following WordprocessingML fragment for a form field:
	`<w:ffData>`
	`  <w:helpText w:type="text" w:val="Example help text." />`
	`</w:ffData>`
	The type attribute has a value of `text`, which specifies that the text in the val attribute is the literal help text for this form field. *end example*]
	The possible values for this attribute are defined by the ST_InfoTextType simple type (§17.18.43).

val (Help Text Value)	Specifies the help text for the current form field. Based on the value of the type attribute, the contents of this field shall be interpreted as follows:
	• When the type attribute value is `text`, contains the literal help text for the form field.
	• When the type attribute value is `autoText`, contains the name of a glossary document entry which contains the help text for the form field.
	[*Example*: Consider the following WordprocessingML fragment for a form field:
	`<w:ffData>`
	`  <w:helpText w:type="autoText" w:val="HelpText" />`
	`</w:ffData>`
	The text in the val attribute is the name of a glossary document entry containing the help text for this form field, since the type attribute has a value of `autoText`. *end example*]
	The possible values for this attribute are defined by the ST_FFHelpTextVal simple type (§17.18.25).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FFHelpText) is located in §A.1. *end note*]

17.16.22 hyperlink (Hyperlink)

This element specifies the presence of a hyperlink at the current location in the document.

[*Example*: Consider the following WordprocessingML fragment for a hyperlink:

```
<w:hyperlink r:id="rId10">
  <w:r>
    <w:t>Click here</w:t>
  </w:r>
</w:hyperlink>
```

The hyperlink element defines a hyperlink whose display text is `Click here`, and whose target is specified by the relationship with an Id attribute value of `rId10`. *end example*]

Attributes	Description

anchor (Hyperlink Anchor)	Specifies the name of a bookmark in the current document which shall be the target of this hyperlink.
	If this attribute is omitted, then the default behavior shall be to navigate to the start of the document. If a hyperlink target is also specified using the r:id attribute, then this attribute shall be ignored. If no bookmark exists in the current document with the given bookmark name, then the default behavior shall be to navigate to the start of the document.
	[*Example*: Consider the following WordprocessingML fragment for a hyperlink:
	`<w:hyperlink w:anchor="chapter3">`
	`  <w:r>`
	`    <w:t>Go to Chapter Three</w:t>`
	`  </w:r>`
	`</w:hyperlink>`
	The anchor attribute specifies that the target of the current hyperlink must be the text contained within the bookmark `chapter3` within the document. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

docLocation (Location in Target Document)	Specifies a location in the target of the hyperlink that has no bookmarks. The method by which the contents of this attribute are linked to document text is outside the scope of ISO/IEC 29500. If this attribute is omitted, then no location shall be associated with the parent hyperlink. If the anchor attribute is also specified, then this attribute can be ignored when the hyperlink is invoked. [*Example*: Consider the following WordprocessingML fragment for a hyperlink:

Attributes	Description

	`<w:hyperlink r:id="rId9" w:docLocation="table">`
	`  <w:r>`
	`    <w:t>Click Here</w:t>`
	`  </w:r>`
	`</w:hyperlink>`
	
	The docLocation attribute specifies that the target of the current hyperlink must be a region targeted by the string `table` within the target document. *end example*]
	
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

history (Add To Viewed Hyperlinks)	Specifies whether the target of the parent hyperlink (as specified via the r:id attribute) shall be added to a list of viewed hyperlinks when it is invoked.
	
	If this attribute is omitted, then its value shall be assumed to be `false`.
	
	[*Example*: Consider the following WordprocessingML fragment for a hyperlink:
	
	`<w:hyperlink r:id="rId9" w:history="true">`
	`  <w:r>`
	`    <w:t>http://www.example.com</w:t>`
	`  </w:r>`
	`</w:hyperlink>`
	
	The history attribute value of `true` specifies that the target of the current hyperlink must be added to a list of visited hyperlinks when invoked within the document. *end example*]
	
	The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

id (Hyperlink Target)
 	
Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships
	Specifies the ID of the relationship whose target shall be used as the target for this hyperlink.
	
	If this attribute is omitted, then there shall be no external hyperlink target for the current hyperlink - a location in the current document can still be target via the anchor attribute. If this attribute exists, it shall supersede the value in the anchor attribute.
	
	[*Example*: Consider the following WordprocessingML fragment for a hyperlink:
	
	`<w:hyperlink r:id="rId9">`
	`  <w:r>`
	`    <w:t>http://www.example.com</w:t>`
	`  </w:r>`
	`</w:hyperlink>`
	
	The id attribute value of `rId9` specifies that relationship in the associated relationship part item with a corresponding Id attribute value must be navigated to when this

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>hyperlink is invoked. For example, if the following XML is present in the associated relationship part item:

<code>&lt;Relationships xmlns="…"&gt;</code>
<code> &lt;Relationship Id="rId9" Mode="External" Target=http://www.example.com /&gt;</code>
<code>&lt;/Relationships&gt;</code>

The target of this hyperlink would therefore be the target of relationship <code>rId9</code> - in this case, <code>http://www.example.com</code>. <em>end example</em>]

The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).</td></tr>
<tr><td>tgtFrame (Hyperlink Target Frame)</td><td>Specifies a frame within the parent HTML frameset for the target of the parent hyperlink when one exists. All values specified by this element shall be handled as follows:

<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td><code>_top</code></td><td>Open hyperlink target in the full region of the current window.</td></tr>
<tr><td><code>_self</code></td><td>Open hyperlink target in the same frame as the hyperlink appears.</td></tr>
<tr><td><code>_parent</code></td><td>Open hyperlink target in the parent of the current frame, or the current frame if this frame has no parent.</td></tr>
<tr><td><code>_blank</code></td><td>Open hyperlink target in a new web browser window.</td></tr>
<tr><td>all other values</td><td>Open hyperlink target in the frame with the specified name. If no frame exists with this name, open in the current frame.

If this string does not begin with an alphabetic character, it shall be ignored.</td></tr>
</table>

If this attribute is omitted, then no target frame information shall be associated with the parent hyperlink. If the current document is not part of a frameset, then this information can be ignored.

[<em>Example</em>: Consider the following WordprocessingML fragment for a hyperlink:</td></tr>
</table>

Attributes	Description

	`<w:hyperlink r:id="rId9" w:tgtFrame="_top">`
	`  <w:r>`
	`    <w:t>http://example.com</w:t>`
	`  </w:r>`
	`</w:hyperlink>`
	The tgtFrame attribute value of `_top` specifies that the target of this hyperlink must be displayed in the full extents of the current window. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

tooltip (Associated String)	Specifies a string which can be surfaced in a user interface as associated with the parent hyperlink. The method by which this string is surfaced by an application is outside the scope of ISO/IEC 29500.
	If this attribute is omitted, then no associated string shall be linked to the parent hyperlink in the document.
	[*Example*: Consider the following WordprocessingML fragment for a hyperlink:
	`<w:hyperlink r:id="rId9" w:tooltip="Click here!">`
	`  <w:r>`
	`    <w:t>http://example.com</w:t>`
	`  </w:r>`
	`</w:hyperlink>`
	The tooltip attribute value specifies that the parent hyperlink has the associated string of `Click here!`, which can be used as desired. *end example*]
	The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Hyperlink) is located in §A.1. *end note*]

17.16.23 instrText (Field Code)

This element specifies that this run contains field codes (§17.16.5) within a complex field in the document.

If this element is contained within a run which is not part of a complex field's field codes, then it and its contents should be treated as regular text. If this element is contained within a del element, then the document is non-conformant.

[*Example*: Consider a complex checkbox field within a WordprocessingML. This field would be represented as follows:

```
<w:r>
  <w:fldChar w:fldCharType="begin" />
</w:r>
<w:r>
  <w:instrText>FORMCHECKBOX</w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate" />
</w:r>
…
<w:r>
  <w:fldChar w:fldCharType="end" />
</w:r>
```

The field code is contained in an instrText node which occurs within the field codes portion of the complex field (i.e. before the separator character). *end example*]

Attributes	Description

xml:space (Content Contains Significant Whitespace)
Namespace: http://www.w3.org/XML/1998/namespace
	Specifies how white space should be handled for the contents of this element using the W3C space preservation rules.
	[*Example*: Consider the following run contained within a WordprocessingML document:
	`<w:r>`
	`  <w:t>   significant whitespace   </w:t>`
	`</w:r>`
	Although there are three spaces on each side of the text content in the run, that whitespace has not been specifically marked as significant, therefore it is subject to the space preservation rules currently specified in that run's scope. *end example*]
	The possible values for this attribute are defined by §2.10 of the XML 1.0 specification.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Text) is located in §A.1. *end note*]

17.16.24 label (Form Field Label)

This element specifies the label identifier associated with the current form field. The identifier representing the label shall be stored on this element's val attribute and is used to reference the unique identifier value of a structured document tag. The contents of the structured document tag resolved by a specific unique identifier shall be used as the label content for the form field that references that specific unique identifier of the structured document tag. If multiple instances of the label element are present, the labels referenced are ordered from most general to most specific. [*Example*: A form element for specifying country name might reference the label for these three items (in order): “Sender”, “Home Address”, and “Country”. *end example*]

If this element is omitted or the value of the label identifier cannot be resolved, then no label shall be associated with the given form field.

[*Example*: Consider the following text box form field, which references a structured document tag as a label:

```
<w:sdt>
  <w:sdtPr>
    <w:id w:val="5" />
  </w:sdtPr>
  <w:sdtContent>
    <w:p>
      <w:r>
        <w:t>Name</w:t>
      </w:r>
    </w:p>
  </w:sdtContent>
</w:sdt>
…
<w:ffData>
  <w:name w:val="TextTextBox" />
  <w:enabled />
  <w:textInput />
  <w:label w:val="5" />
</w:ffData>
```

The label element specifies that the text box form field uses the contents of the structured document tag with an identifier value of 5 as a label source. In this example, the label contents is “Name”. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.16.25 listEntry (Drop-Down List Entry)

This element specifies the presence of a single drop-down list entry within the parent drop-down list form field in the document. The order of appearance of the series of listEntry elements in the WordprocessingML markup shall dictate the order of the entries in the drop-down list when it is displayed.

[*Example*: Consider the following WordprocessingML fragment for the properties of a drop-down list form field:

```
<w:ffData>
  <w:ddList>
    <w:listEntry w:val="One" />
    <w:listEntry w:val="Two" />
    <w:listEntry w:val="Three" />
  </w:ddList>
</w:ffData>
```

The three listEntry elements each specify one drop-down list entry for the parent drop-down list form field. In this case, these properties specify that the drop-down list must contain three entries of One, Two, and Three in that order when displayed. *end example*]

Attributes	Description

val (String Value)	Specifies that its contents contain a string.
	The contents of this string are interpreted based on the context of the parent XML element.
	[*Example*: Consider the following WordprocessingML fragment:
	`<w:pPr>`
	`  <w:pStyle w:val="Heading1" />`
	`</w:pPr>`
	The value of the val attribute is the ID of the associated paragraph style's `styleId`.
	However, consider the following fragment:
	`<w:sdtPr>`
	`  <w:alias w:val="SDT Title Example" />`

Attributes	Description
	… `</w:sdtPr>` In this case, the decimal number in the val attribute is the caption of the nearest ancestor structured document tag. In each case, the value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_String simple type (§22.9.2.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.1. *end note*]

17.16.26 maxLength (Text Box Form Field Maximum Length)

This element specifies the maximum length of text which should be allowed within the parent text box form field before any formatting specified by the format element (§17.16.20). If the current contents of this field exceed the specified value when the document is loaded, that violation shall not result in an error, but the application shall prevent the addition of any additional characters until the contents are brought below that limit.

If this element is omitted, then there shall be no limit on the number of characters in the parent text box form field.

[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:

```
<w:ffData>
  <w:textInput>
    <w:type w:val="number" />
    <w:maxLength w:val="4" />
    <w:format w:val="0.00" />
  </w:textInput>
</w:ffData>
```

The maxLength element specifies that the contents of this form field should not be allowed to exceed four characters when edited by an application. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber:

Attributes	Description
	`<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

17.16.27 name (Form Field Name)

This element specifies the name of the current form field.

[*Example*: Consider the following WordprocessingML fragment for a text box form field:

```
<w:r>
  <w:fldChar w:fldCharType="begin">
    <w:ffData>
      <w:name w:val=”FirstName” />
      <w:textInput>
        …
      </w:textInput>
    </w:ffData>
  </w:fldChar>
</w:r>
<w:r>
  <w:instrText> FORMTEXT </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r>
  <w:t>1</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

The name element specifies that the name of the current form field is `FirstName`. *end example*]

Attributes	Description

val (Form Field Name Value)	Specifies the name of the form field.
	If this attribute is omitted, then the parent form field shall have no name.
	[*Example*: Consider the following WordprocessingML fragment for a form field:
	`<w:ffData>`
	`  <w:name w:val="ExampleFieldName"/>`
	`</w:ffData>`
	The val attribute specifies that the name of the current form field is `ExampleFieldName`. *end example*]
	The possible values for this attribute are defined by the ST_FFName simple type (§17.18.26).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FFName) is located in §A.1. *end note*]

17.16.28 result (Drop-Down List Selection)

This element specifies the zero-based index of the currently selected entry for the parent drop-down list form field.

If this element is omitted, then the current drop-down list form field shall have the selection specified by the value of the default element (§17.16.11). If the attribute value references an index value which does not exist (i.e. a negative number or a number that exceeds the number of items in the drop-down list), then this value can be ignored and the current drop-down list form field shall have the selection specified by the value of the default element.

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
  <w:r>
    <w:fldChar w:fldCharType="begin">
      <w:ffData>
        <w:ddList>
          <w:default w:val="1" />
          <w:result w:val="2" />
          <w:listEntry w:val="One" />
          <w:listEntry w:val="Two" />
          <w:listEntry w:val="Three" />
        </w:ddList>
      </w:ffData>
    </w:fldChar>
  </w:r>
  <w:r>
    <w:instrText> FORMDROPDOWN </w:instrText>
  </w:r>
  <w:r>
    <w:fldChar w:fldCharType="end"/>
  </w:r>
```

The result element specifies the index of the currently selected value of the drop-down list form field to be 2. In this case, the resulting default value text is Three. *end example*]

Attributes	Description
val (Decimal Number Value)	Specifies that the contents of this attribute contains a decimal number. The contents of this decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of simple type ST_DecimalNumber: `<… w:val="1512645511" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_DecimalNumber simple type (§17.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DecimalNumber) is located in §A.1. *end note*]

©ISO/IEC 2012 – All rights reserved

17.16.29 size (Checkbox Form Field Size)

This element specifies the exact size for the parent checkbox form field. The resulting field shall be displayed in this point size regardless of the size specified by the formatting of its corresponding content in the document via the style hierarchy.

[*Example*: Consider the following WordprocessingML fragment for the properties of a checkbox form field:

```
<w:ffData>
  <w:checkBox>
    <w:size w:val="20" />
    <w:checked w:val="true" />
  </w:checkBox>
</w:ffData>
```

The size element specifies that the checkbox must be displayed in a ten point font size, regardless of the formatting which would normally be applied to this text via the style hierarchy. *end example*]

Attributes	Description
val (Half Point Measurement)	Specifies a positive measurement specified in half-points (1/144 of an inch). The contents of this attribute value are interpreted based on the context of the parent XML element. [*Example*: Consider the following WordprocessingML fragment: `<… w:val="30" />` The value in the val attribute is `30`, which is equivalent to 15 points (30 half-points). This value is interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_HpsMeasure simple type (§17.18.42).

[*Note*: The W3C XML Schema definition of this element's content model (CT_HpsMeasure) is located in §A.1. *end note*]

17.16.30 sizeAuto (Automatically Size Form Field)

This element specifies that the parent checkbox form field shall be formatted using the point size which is applied to its field characters via the style hierarchy.

[*Example*: Consider the following WordprocessingML fragment for the properties of a checkbox form field:

```
<w:r>
  <w:rPr>
    <w:sz w:val="40"/>
  </w:rPr>
  <w:fldChar w:fldCharType="begin">
    <w:ffData>
      <w:checkBox>
        <w:sizeAuto />
        <w:checked w:val="true" />
      </w:checkBox>
    </w:ffData>
  </w:fldChar>
</w:r>
```

The sizeAuto element specifies that the checkbox must be displayed in the point size of the formatting which would normally be applied to this text via the style hierarchy. In this case, this size is the twenty points specified via the direct formatting on the parent run. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.16.31 statusText (Associated Status Text)

This element specifies optional status text which shall be associated with the parent form field. The method or user interface by which this status text can be surfaced is not defined by ISO/IEC 29500.

If this element is omitted, then no status text shall be associated with the current form field.

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
<w:r>
  <w:fldChar w:fldCharType="begin">
    <w:ffData>
      <w:statusText w:type="text" w:val="Example status text."/>
      <w:ddList>
        …
      </w:ddList>
    </w:ffData>
  </w:fldChar>
</w:r>
<w:r>
  <w:instrText> FORMDROPDOWN </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

©ISO/IEC 2012 – All rights reserved

The statusText element specifies the status text for the parent form field - in this case, literal text consisting of the string `Example status text.` *end example*]

Attributes	Description

type (Status Text Type)	Specifies the type of status text which is specified by this element, defined by the simple type below.
	If this attribute is omitted, then its value shall be assumed to be `text`.
	[*Example*: Consider the following WordprocessingML fragment for a form field:
	`<w:ffData>`
	`  <w:statusText w:type="text" w:val="Example status text." />`
	`</w:ffData>`
	The type attribute has a value of `text`, which specifies that the text in the val attribute is the literal status text for this form field. *end example*]
	The possible values for this attribute are defined by the ST_InfoTextType simple type (§17.18.43).

val (Status Text Value)	Specifies the status text for the current form field. Based on the value of the type attribute, the contents of this field shall be interpreted as follows:
	• When the type attribute value is `text`, contains the literal status text for the form field.
	• When the type attribute value is `autoText`, contains the name of a glossary document entry which contains the status text for the form field.
	[*Example*: Consider the following WordprocessingML fragment for a form field:
	`<w:ffData>`
	`  <w:statusText w:type="autoText" w:val="MyStatusText" />`
	`</w:ffData>`
	The text in the val attribute is the name of a glossary document entry containing the status text for this form field, since the type attribute has a value of `autoText`. *end example*]
	The possible values for this attribute are defined by the ST_FFStatusTextVal simple type (§17.18.27).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FFStatusText) is located in §A.1. *end note*]

17.16.32 tabIndex (Form Field Navigation Order Index)

This element specifies the position of the current form field in the navigation (tab) order used in the document. The tabbing index shall be stored on this element's val attribute and is analogous to the tabIndex attribute in HTML.

Objects that support tab index shall be navigated by consumers in the following order:

- Objects for which the XML specifies a non-zero tabIndex value are navigated first. Navigation proceeds with the element with the lowest resolved value of tabIndex to the element with the highest resolved value of tabIndex.
  - Objects that specify identical resolved values of tabIndex is navigated in the lexical order in which the elements appear in the underlying WordprocessingML.
- Objects for which the XML does not specify an index or objects for which the XML specifies a resolved tabIndex value of 0 are navigated last. These objects are navigated in the lexical order in which they appear in the underlying WordprocessingML.

[*Example*: Consider the following two text box form fields where form field specifies a tab index:

```
<w:ffData>
  <w:name w:val="FirstName" />
  <w:enabled />
  <w:textInput />
  <w:tabIndex w:val="1" />
</w:ffData>
…
<w:ffData>
  <w:name w:val="LastName" />
  <w:enabled />
  <w:textInput />
  <w:tabIndex w:val="2" />
</w:ffData>
```

The tabIndex element specifies that the `FirstName` form field must be the first content to be reached via tabbing, whereas the `LastName` form field must be the second content to be reached via tabbing. *end example*]

Attributes	Description
val (Positive Decimal Number Value)	Specifies that the contents of this attribute contains a positive decimal number. The contents of this positive decimal number are interpreted based on the context of the parent XML element. [*Example*: Consider the following numeric WordprocessingML property of type ST_UnsignedDecimalNumber:

©ISO/IEC 2012 – All rights reserved

Attributes	Description
	`<… w:val="15" />` The value of the val attribute is a decimal number whose value must be interpreted in the context of the parent element. *end example*] The possible values for this attribute are defined by the ST_UnsignedDecimalNumber simple type (§22.9.2.16).

[*Note*: The W3C XML Schema definition of this element's content model (CT_UnsignedDecimalNumber) is located in §A.1. *end note*]

17.16.33 textInput (Text Box Form Field Properties)

This element specifies a set of properties which shall be associated with the parent FORMTEXT text box form field (§17.16.5.22) within the document.

If the parent form field is not a text box (i.e. its field code does not have a value of FORMTEXT), then these properties can be ignored.

[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:

```
<w:ffData>
  <w:textInput>
    <w:maxLength w:val="4" />
    …
    <w:type w:val="number" />
  </w:textInput>
</w:ffData>
```

The textInput element specifies that it contains a set of properties for the parent text box form field. In this case, these properties specify that the drop-down list must contain no more than four characters via the maxLength element (§17.16.26), and that its contents must contain a number via the type element (§17.16.34). *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_FFTextInput) is located in §A.1. *end note*]

17.16.34 type (Text Box Form Field Type)

This element specifies the type of the contents of the current text box form field. This element shall not be used to prevent the successful loading of any contents in the field, but shall be used to parse the formatting specified in the format element (§17.16.20) and should be used to prevent the addition of illegal content when its contents are edited by an application.

If this element is omitted, then its default value shall be assumed to be regular.

[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:

```
<w:ffData>
  <w:textInput>
    <w:type w:val="number" />
    <w:maxLength w:val="4" />
    <w:format w:val="0.00" />
  </w:textInput>
</w:ffData>
```

The type element specifies that the contents of this form field should be handled as a number by an application. *end example*]

Attributes	Description

val (Text Box Form Field Type Values)	Specifies the type of the text box form field, as defined by the simple type referenced below.
	[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:
	`<w:ffData>`
	`  <w:textInput>`
	`    <w:type w:val="currentDate" />`
	`  </w:textInput>`
	`</w:ffData>`
	The val attribute value of currentDate specifies that the contents of this form field should be the current date when the field is updated. *end example*]
	The possible values for this attribute are defined by the ST_FFTextType simple type (§17.18.28).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FFTextType) is located in §A.1. *end note*]

17.17 Miscellaneous Topics

This section covers topics not covered elsewhere within the WordprocessingML documentation.

17.17.1 Subdocuments

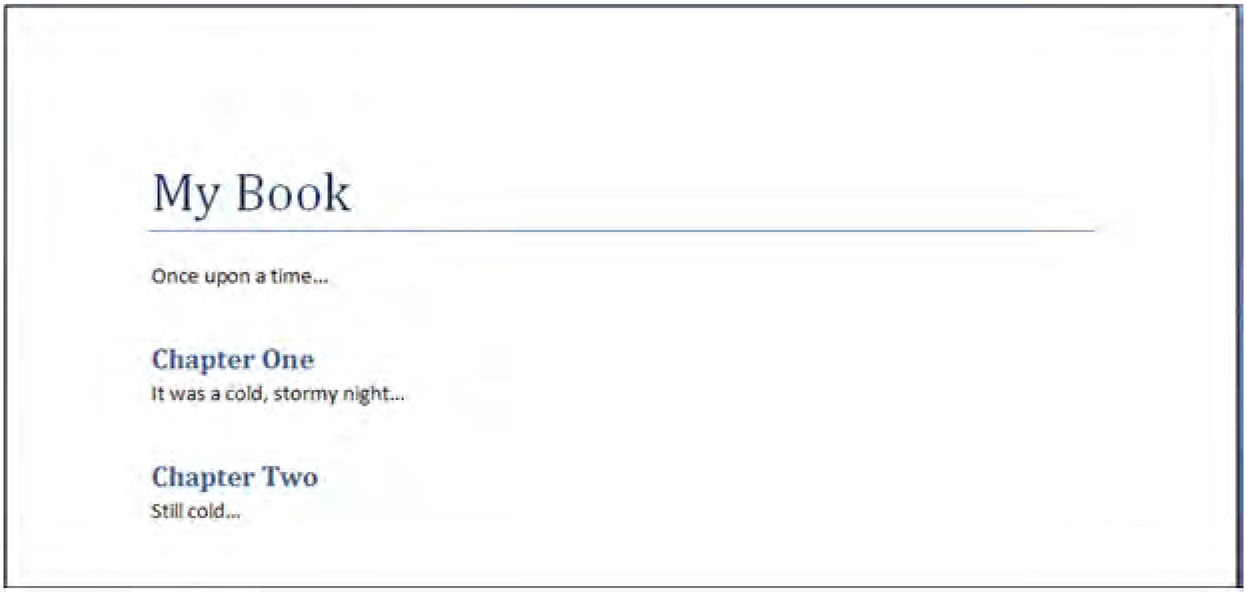
Within a WordprocessingML document, it is sometimes necessary to break a large document into two or more separate WordprocessingML document files, allowing each of these files to be distributed, edited, and handled independently.

[*Example*: A book might consist of five chapters, each edited by a separate author. The editor for the book would therefore desire to create six WordprocessingML documents - one for each author to work on their chapter, and a main document which collates the content of the five chapters appropriately. *end example*]

When a WordprocessingML document is comprised of other WordprocessingML documents in this way, the resulting documents are called a master document and its subdocuments.

- A *subdocument* is a WordprocessingML document - there is no specific information in a document which classifies it as such, other than that it is incorporated into another document.
- A *master document* is a document which incorporates one or more subdocuments (as well as optional WordprocessingML content) to create a larger document

[*Example*: Consider a WordprocessingML document which is being used to write a book:



My Book

Once upon a time...

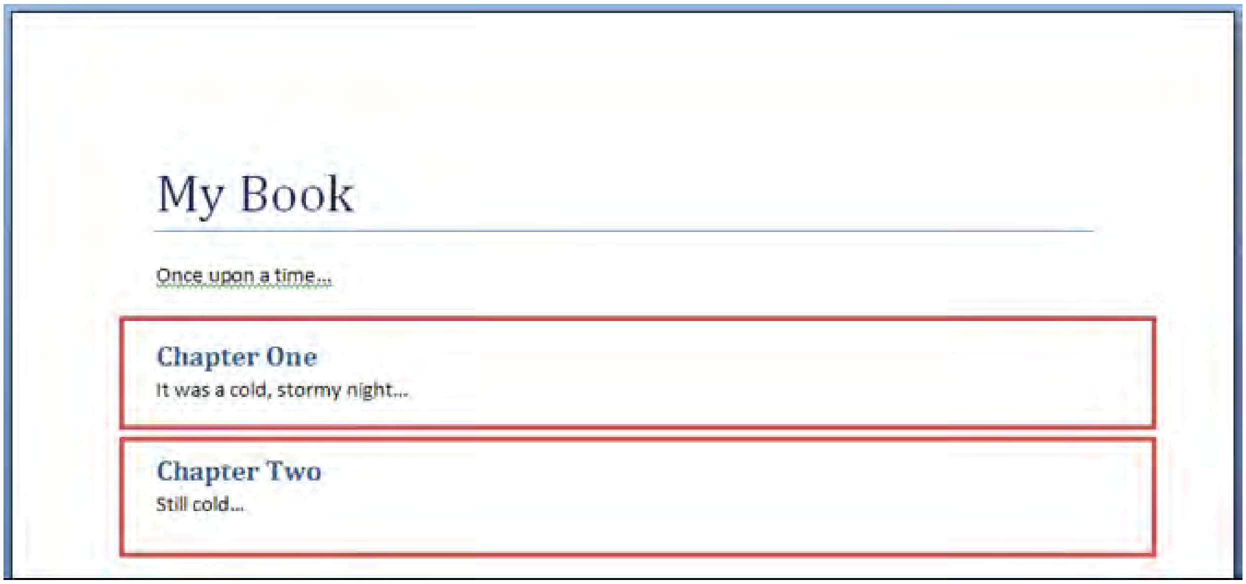
Chapter One

It was a cold, stormy night...

Chapter Two

Still cold...

To allow this document to be written by multiple authors, each chapter in the book is placed in a separate file (the sections highlighted in red below):



My Book

Once upon a time...

Chapter One

It was a cold, stormy night...

Chapter Two

Still cold...

The result is three WordprocessingML documents:

- A master document (containing the title of the book, the first paragraph, and references to the subdocuments for each chapter)
- Two subdocuments (one for each chapter)

end example]

17.17.1.1 subDoc (Anchor for Subdocument Location)

This element specifies a location within a master document for the insertion of the contents of a specified subdocument. The specified subdocument's contents should appear at the specified location within the master document as needed, but shall remain part of the separate file specified by the subdocument location. The location of the subdocument shall be specified by the relationship whose Id attribute matches the id attribute on this element.

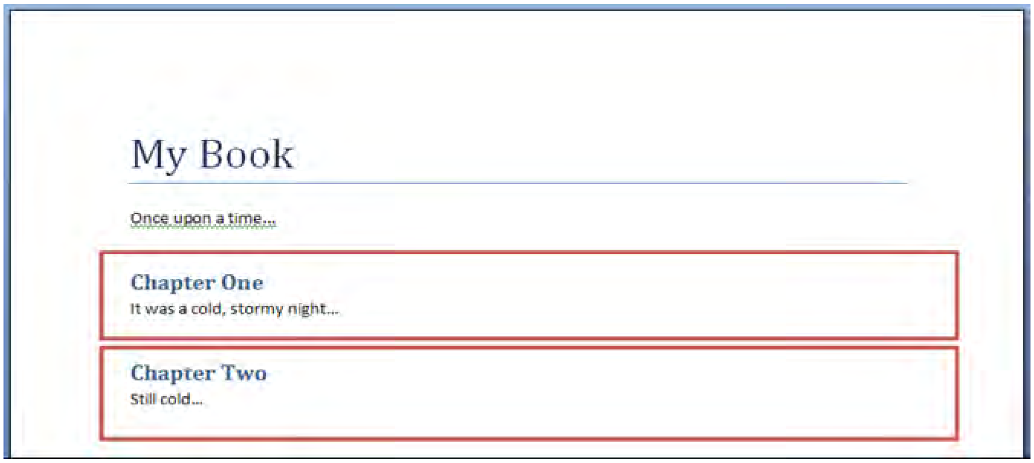
If the relationship type of the relationship specified by this element is not http://purl.oclc.org/ooxml/officeDocument/relationships/subDocument, is not present, or does not have a TargetMode attribute value of `External`, then the document shall be considered non-conformant.

When a subdocument is displayed at the specified location within the master document, the following logic determines how the styles and formatting of the subdocument and associated master document shall be handled:

- All direct formatting in the subdocument is retained.
- For each style in each subdocument:
  - If a style with the same styleId attribute value exists in the master document, display the contents of the subdocument using the style in the master document.

- Otherwise, import the style into the master document and display the content using the original style.
- If a subsequent subdocument also contains a style with the same styleId attribute value, the latter's style is ignored and the content is displayed using the version of the style imported into the master document.

[*Example*: Consider a book consisting of three chapters, two of which have been divided into subdocuments as follows (the red rectangle indicates the bounds of each subdocument's contents):



My Book

Once upon a time…

Chapter One

It was a cold, stormy night…

Chapter Two

Still cold…

The resulting master document would consist of its own WordprocessingML content as well as subdocument anchors in the appropriate locations:

```
<w:body>
  <w:p>
    …
    <w:r>
      <w:t>My Book</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:r>
      <w:t>Once upon a time…</w:t>
    </w:r>
  </w:p>
  <w:p>
    <w:subDoc r:id="subDocRel1" />
  </w:p>
```

```
    <w:p>
      <w:subDoc r:id="subDocRel2" />
    </w:p>
    <w:sectPr>
      …
    </w:sectPr>
  </w:body>
```

The two subDoc elements specify that the subdocuments targeted by the relationships with an ID of subDocRel1 and subDocRel2 must be imported in that order after the content of the first two paragraphs of content. Examining the contents of the corresponding relationship part item, we can see the targets for those relationships:

```
  <Relationships … >
    …
    <Relationship Id="subDocRel1" TargetMode="External"
  Type="http://purl.oclc.org/ooxml/officeDocument/relationships/subDocument"
  Target="Chapter1.docx" />
    <Relationship Id="subDocRel2" TargetMode="External"
  Type="http://purl.oclc.org/ooxml/officeDocument/relationships/subDocument"
  Target="Chapter2.docx" />
    …
  </Relationships>
```

The corresponding relationship part item shows that the two files to be imported are located in the same location as the current file and name `Chapter1.docx` and `Chapter2.docx` respectively. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part. The specified relationship shall match the relationship type required by the parent element: • http://purl.oclc.org/ooxml/officeDocument/relationships/customXml for the contentPart element • http://purl.oclc.org/ooxml/officeDocument/relationships/footer for the footerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/header for the headerReference element • http://purl.oclc.org/ooxml/officeDocument/relationships/font for the embedBold, embedBoldItalic, embedItalic, or embedRegular elements • http://purl.oclc.org/ooxml/officeDocument/relationships/printerSettings for the printerSettings element • http://purl.oclc.org/ooxml/officeDocument/relationships/hyperlink for the longDesc or hyperlink element

Attributes	Description
	[*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rel) is located in §A.1. *end note*]

17.17.2 Alternative Format Import

When generating WordprocessingML documents, it is sometimes necessary to include existing document content (henceforth called *external content*) within the document. External content in a document is typically included because it was stored in a format other than the WordprocessingML format defined by ISO/IEC 29500.

As described in §11.3.1, conforming producers shall not create WordprocessingML packages which utilize this mechanism.

In order to facilitate the inclusion of such content without requiring its conversion as a prerequisite to its inclusion in a document, WordprocessingML includes the facility for applications to implement the import of external content in any format as part of a WordprocessingML document. This functionality, called external content import, allows the inclusion of content of an arbitrary content type within the WordprocessingML package, which shall then be opened and merged into the main document when the package is consumed by applications which understand that content type.

[*Example*: Consider a WordprocessingML document which is being created based on the following existing HTML content:

```
<html … >
  <body style="margin-left:200px;margin-top:50px">
    <p>Paragraph one.</p>
    <blockquote style="border:5px solid #00FFFF">Paragraph in a
blockquote.</blockquote>
    <p>Paragraph two.</p>
  </body>
</html>
```

This content could be converted to its WordprocessingML equivalents using the XML syntax defined by ISO/IEC 29500, or a more basic tool can use the external content import to include the HTML document within a WordprocessingML package, allowing a subsequent consumer of that content to import the resulting HTML. When the resulting WordprocessingML package is opened, the HTML document must be read (if it is an alternate format understood by the consuming application) and migrated into the appropriate location in the main WordprocessingML document. *end example*]

17.17.2.1 altChunk (Anchor for Imported External Content)

This element specifies a location within a document for the insertion of the contents of a specified file containing external content to be imported into the main WordprocessingML document. The specified file's contents should appear at the specified location within the document, and can henceforth be emitted as regular WordprocessingML without distinction to its origin. The location of the external content to be imported shall be specified by the relationship whose Id attribute matches the id attribute on this element.

If the relationship type of the relationship specified by this element is not http://purl.oclc.org/ooxml/officeDocument/relationships/aFChunk, is not present, or does not have a TargetMode attribute value of `Internal`, then the document shall be considered non-conformant. If an application cannot process external content of the content type specified by the targeted part, then it should ignore the specified alternate content but continue to process the file. If possible, it should also provide some indication that unknown content was not imported.

[*Example*: Consider a WordprocessingML document consisting of contents which must be imported from the following HTML document:

```
<html … >
  <body style="margin-left:200px;margin-top:50px">
    <p>Paragraph one.</p>
    <blockquote style="border:5px solid #00FFFF">Paragraph in a 
blockquote.</blockquote>
    <p>Paragraph two.</p>
  </body>
</html>
```

The resulting WordprocessingML host document would consist of its own WordprocessingML content as well as an external content import anchor in the appropriate location:

```
<w:body>
  <w:altChunk r:id="altChunk1" />
  <w:p/>
  <w:sectPr>
    …
  </w:sectPr>
</w:body>
```

The altChunk element specifies that the external content targeted by the relationship with an ID of `altChunk1` must be imported at the beginning of the document. Examining the contents of the corresponding relationship part item, we can see the targets for that relationship:

```
<Relationships … >
  …
  <Relationship Id="altChunk1" TargetMode="Internal"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/aFChunk" Target="import.htm"
/>
  …
</Relationships>
```

The corresponding relationship part item shows that the file to be imported is located next to the main document and is named `import.htm`. *end example*]

Attributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a specified part containing alternate content for import. If the specified relationship does not match the relationship type required by the parent element, then this document shall be considered to be non-conformant. [*Example*: Consider an XML element which has the following id attribute: `<… r:id="rId10" />` The markup specifies the associated relationship part with relationship ID `rId1` contains the corresponding relationship information for the parent XML element. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_AltChunk) is located in §A.1. *end note*]

17.17.2.2 altChunkPr (External Content Import Properties)

This element specifies the set of properties which shall be applied to the import of the external content specified by the parent altChunk element. Within ISO/IEC 29500, only one property is specified.

[*Example*: Consider a WordprocessingML document consisting of contents which contains an external content import anchor in the appropriate location:

```
<w:body>
  <w:altChunk r:id="altChunk1">
    <w:altChunkPr>
      <w:matchSrc w:val="false" />
    </w:altChunkPr>
  </w:altChunk>
  <w:p/>
```

```
    <w:sectPr>
      …
    </w:sectPr>
  </w:body>
```

The altChunkPr element specifies the set of properties applied to the external content import when importing the specified content. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_AltChunkPr) is located in §A.1. *end note*]

17.17.2.3 matchSrc (Keep Source Formatting on Import)

This element specifies if any style definitions present in the imported content shall be overridden by identical styles present in the host WordprocessingML document. If this element's val attribute is `true`, then any style exists in both the imported content and main document shall be maintained on the imported content by redefining the style name and/or ID as needed. Conversely, if this element's val attribute is false, any style which exists in both the imported content and main document shall apply the style form the main document in place of the style in the imported content.

If this element is omitted, then styles from the main document shall override identical styles from the imported content.

[*Example*: Consider a WordprocessingML document consisting of contents which contains an external content import anchor in the appropriate location:

```
  <w:body>
    <w:altChunk r:id="altChunk1">
      <w:altChunkPr>
        <w:matchSrc w:val="true" />
      </w:altChunkPr>
    </w:altChunk>
    <w:p/>
    <w:sectPr>
      …
    </w:sectPr>
  </w:body>
```

The matchSrc element has a val attribute value of `true`, which specifies that conflicting styles must be maintained when importing the specified content. For example, if the `Heading 1` style was defined in both places, then applications must ensure that the resulting document does not lose either instance of its formatting as appropriate. *end example*]

This element's content model is defined by the common boolean property definition in §17.17.4.

17.17.3 Roundtripping Alternate Content

Office Open XML defines a mechanism for the storage of content which is not defined by ISO/IEC 29500, for example extensions developed by future software applications which leverage the Office Open XML formats. This mechanism allows for the storage of a series of alternative representations of content, of which the consuming application should use the first alternative whose requirements are met.

[*Example*: Consider an application which creates a new paragraph property intended to make the colors of its text change colors randomly when it is displayed. This functionality is not defined in ISO/IEC 29500, and so the application might choose to create an alternative representation setting a different manual color on each character for clients which do not understand this extension using an AlternateContent block as follows:

```
<ve:AlternateContent xmlns:ve="…">
  <ve:Choice Requires="colors" xmlns:colors="urn:randomTextColors">
    <w:p>
      <w:pPr>
        <colors:random colors:val="true" />
      </w:pPr>
      <w:r>
        <w:t>Random colors!</w:t>
      </w:r>
    </w:p>
  </ve:Choice>
  <ve:Fallback>
    <w:p>
      <w:r>
        <w:rPr>
          <w:color w:val="FF0000" />
        </w:rPr>
        <w:t>R</w:t>
      </w:r>
      <w:r>
        <w:rPr>
          <w:color w:val="00FF00" />
        </w:rPr>
        <w:t>a</w:t>
      </w:r>
      …
    </w:p>
  </ve:Fallback>
</ve:AlternateContent>
```

©ISO/IEC 2012 – All rights reserved

The Choice element that requires the new color extensions uses the random element in its namespace, and the Fallback element allows clients that do not support this namespace to see an appropriate alternative representation. *end example*]

These alternate content blocks can occur at any location within a WordprocessingML document, and applications shall handle and process them appropriately (taking the appropriate choice).

However, WordprocessingML does not explicitly define a set of locations where applications shall attempt to store and roundtrip all non-taken choices whenever possible. This behavior is therefore application-defined.

[*Example*: If an application does not understand the colors extension, the resulting file (if alternate choices are to be preserved would appear as follows:

```
<ve:AlternateContent xmlns:ve="…">
  <ve:Choice Requires="colors" xmlns:colors="urn:randomTextColors">
    …
  </ve:Choice>
  <ve:Fallback>
    …
  </ve:Fallback>
</ve:AlternateContent>
```

The file would then appear as follows after the choice is processed:

```
<w:p>
  <w:r>
    <w:rPr>
      <w:color w:val="FF0000" />
    </w:rPr>
    <w:t>R</w:t>
  </w:r>
  <w:r>
    <w:rPr>
      <w:color w:val="00FF00" />
    </w:rPr>
    <w:t>a</w:t>
  </w:r>
  …
</w:p>
```

The state of the alternate choices (preserved or not) is dependent on the application hosting the file. Preserving the content involves storing each non-taken choice while the file is being edited, and writing out the file with an AlternateContent block when it is resaved. *end example*]

17.17.4 Boolean Property (CT_OnOff)

This common complex type specifies a boolean attribute used throughout WordprocessingML.

Attributes	Description
val (On/Off Value)	Specifies a binary value for the property defined by the parent XML element. A value of `1` or `true` specifies that the property shall be explicitly applied. This is the default value for this attribute, and is implied when the parent element is present, but this attribute is omitted. A value of `0` or `false` specifies that the property shall be explicitly turned off. [*Example*: For example, consider the following on/off property: `<… w:val="false"/>` The val attribute explicitly declares that the property is `false`. *end example*] The possible values for this attribute are defined by the ST_OnOff simple type (§22.9.2.7).

[*Note*: The W3C XML Schema definition of this complex type's content model (CT_OnOff) is located in §A.1. *end note*]

17.18 Simple Types

This is the complete list of simple types dedicated to WordprocessingML.

17.18.1 ST_AnnotationVMerge (Table Cell Vertical Merge Revision Type)

This simple type specifies the possible values for the vertical merge setting which applied to a table cell by a cell merge (or split) revision.

[*Example*: Consider a two row by two column table in which the cells in the second column are merged, and this change is tracked as a revision. The annotation on the last cell in the table would appear as follows:

```
<w:tc>
  <w:tcPr>
    <w:cellMerge … w:vMerge="cont" />
  </w:tcPr>
  …
</w:tc>
```

The vmerge attribute value of `cont` specifies that the revision on the table cell resulted in it being merged with the previous set of vertically merged cells above it (whether that was one cell or many). *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
cont (Vertically Merged Cell)	Specifies that the revision resulted in this cell being vertically merged with the cell above it.
rest (Vertically Split Cell)	Specifies that the revision resulted in this cell being vertically split from the one above it.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_AnnotationVMerge) is located in §A.1. *end note*]

17.18.2 ST_Border (Border Styles)

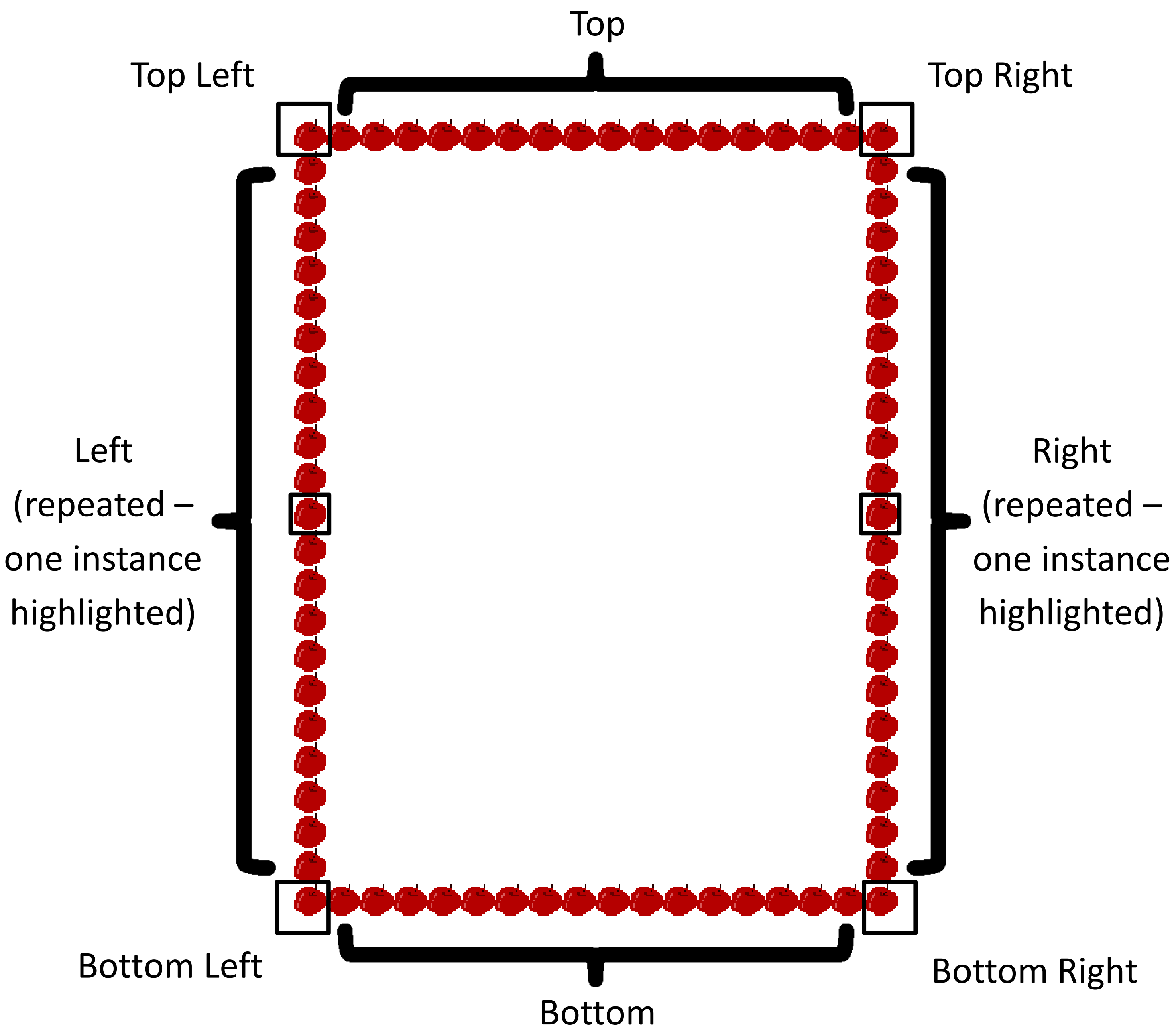
This simple type specifies the kinds of borders which can be specified for WordprocessingML objects which have a border.

Borders can be separated into two types:

- *Line borders*, which specify a pattern to be used when drawing a line around the specified object.
- *Art borders*, which specify a repeated image to be used when drawing a border around the specified object.

Line borders can be specified on any object which allows a border, however, art borders can only be used as a border at the page level - the borders under the pgBorders element (§17.6.10).

For art borders, each border definition is specified by the combination of eight images, as follows:



For the edge borders (top, bottom, left, and right), the border image is repeated as necessary to span the distance between the two intersecting corners. Electronic normative definitions of each piece of border art are included in Annex F.

[*Example*: Consider a left border resulting in the following WordprocessingML:

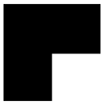
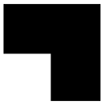
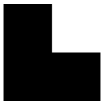
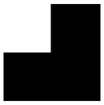
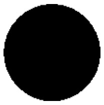
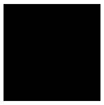
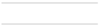
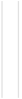
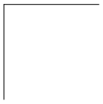
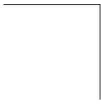
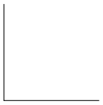
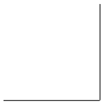
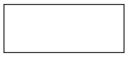
```
<w:left w:val="single" …/>
```

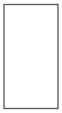
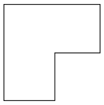
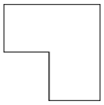
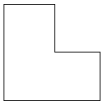
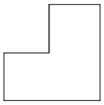
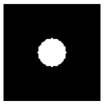
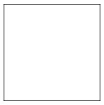
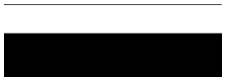
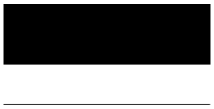
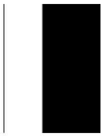
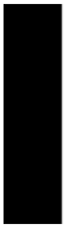
This border's val is `single`, indicating that the border style is a single line border. *end example*]

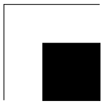
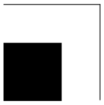
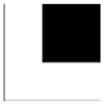
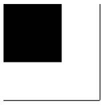
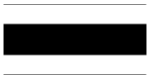
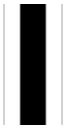
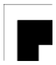
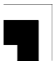
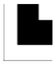
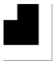
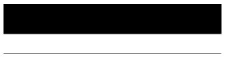
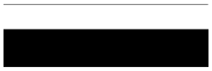
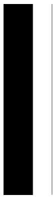
This simple type's contents are a restriction of the W3C XML Schema string datatype.

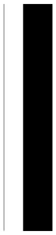
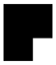
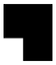
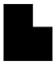
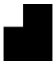
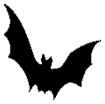
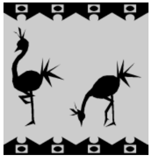
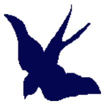
This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
apples (Apples Art Border)	Specifies an art border using the following images: All Sides:
archedScallops (Arched Scallops Art Border)	Specifies an art border using the following images: All Sides:
babyPacifier (Baby Pacifier Art Border)	Specifies an art border using the following images: All Sides:
babyRattle (Baby Rattle Art Border)	Specifies an art border using the following images: All Sides:
balloons3Colors (Three Color Balloons Art Border)	Specifies an art border using the following images: All Sides:
balloonsHotAir (Hot Air Balloons Art Border)	Specifies an art border using the following images: All Sides:
basicBlackDashes (Black Dash Art Border)	Specifies an art border using the following images: Top and Bottom: Left and Right:

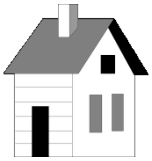
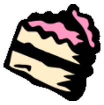
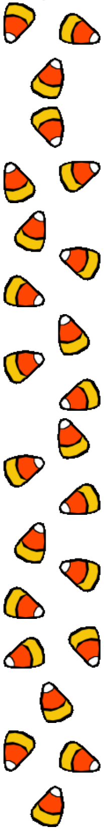
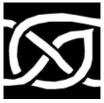
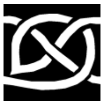
Enumeration Value	Description
	Top Left:  Top Right:  Bottom Left:  Bottom Right: 
basicBlackDots (Black Dot Art Border)	Specifies an art border using the following images: All Sides: 
basicBlackSquares (Black Square Art Border)	Specifies an art border using the following images: All Sides: 
basicThinLines (Thin Line Art Border)	Specifies an art border using the following images: Top and Bottom:  Left and Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
basicWhiteDashes (White Dash Art Border)	Specifies an art border using the following images: Top and Bottom: 

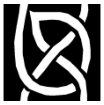
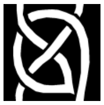
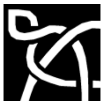
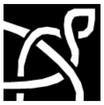
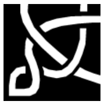
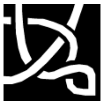
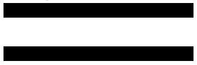
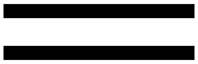
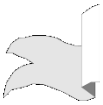
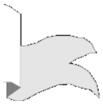
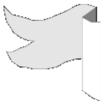
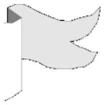
Enumeration Value	Description
	Left and Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
basicWhiteDots (White Dot Art Border)	Specifies an art border using the following images: All Sides: 
basicWhiteSquares (White Square Art Border)	Specifies an art border using the following images: All Sides: 
basicWideInline (Wide Inline Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right: 

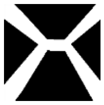
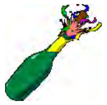
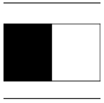
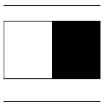
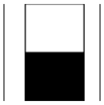
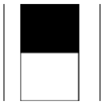
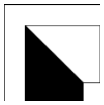
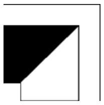
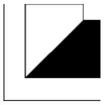
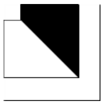
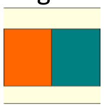
Enumeration Value	Description
	Top Left:  Top Right:  Bottom Left:  Bottom Right: 
basicWideMidline (Wide Midline Art Border)	Specifies an art border using the following images: Top and Bottom:  Left and Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
basicWideOutline (Wide Outline Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left: 

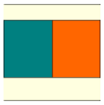
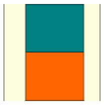
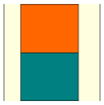
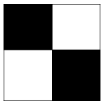
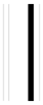
Enumeration Value	Description
	Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
bats (Bats Art Border)	Specifies an art border using the following images: All Sides: 
birds (Birds Art Border)	Specifies an art border using the following images: Top, Bottom, Top Left, Top Right, Bottom Left and Bottom Right:  Left and Right: [*Note*: Second image is blank. *end note*]
birdsFlight (Birds Flying Art Border)	Specifies an art border using the following images: All Sides: 
cabins (Cabin Art Border)	Specifies an art border using the following images:

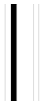
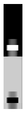
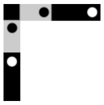
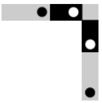
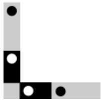
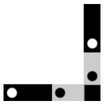
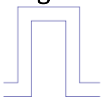
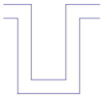
©ISO/IEC 2012 – All rights reserved

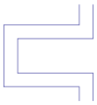
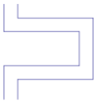
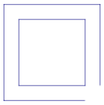
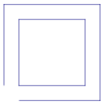
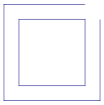
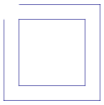
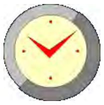
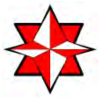
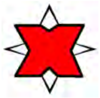
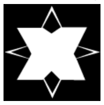
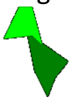
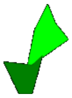
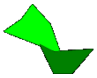
Enumeration Value	Description
	All Sides: 
cakeSlice (Cake Art Border)	Specifies an art border using the following images: All Sides: 
candyCorn (Candy Corn Art Border)	Specifies an art border using the following images:  Top: Bottom: Left: Right: Top Left: Top Right: Bottom Left: Bottom Right:
celticKnotwork (Knot Work Art Border)	Specifies an art border using the following images: Top:  Bottom: 

Enumeration Value	Description
	Left:  Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
certificateBanner (Certificate Banner Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left: [Note: Image is blank. end note] Right: [Note: Image is blank. end note] Top Left:  Top Right:  Bottom Left:  Bottom Right: 

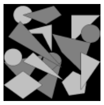
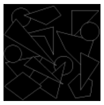
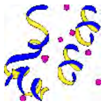
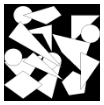
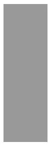
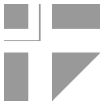
Enumeration Value	Description
chainLink (Chain Link Art Border)	Specifies an art border using the following images: All Sides: 
champagneBottle (Champagne Bottle Art Border)	Specifies an art border using the following images: All Sides: 
checkedBarBlack (Black and White Bar Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
checkedBarColor (Color Checked Bar Art Border)	Specifies an art border using the following images: Top: 

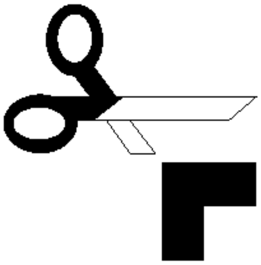
Enumeration Value	Description
	Bottom:  Left:  Right:  Top Left: Top Right: Bottom Left: Bottom Right:
checkered (Checkerboard Art Border)	Specifies an art border using the following images: All Sides: 
christmasTree (Christmas Tree Art Border)	Specifies an art border using the following images: All Sides:
circlesLines (Circles And Lines Art Border)	Specifies an art border using the following images: Top: Bottom: Left: 

Enumeration Value	Description
	Right:  Top Left: Top Right: Bottom Left: Bottom Right:
circlesRectangles (Circles and Rectangles Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
classicalWave (Wave Art Border)	Specifies an art border using the following images: Top:  Bottom: 

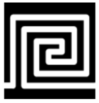
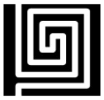
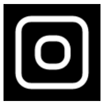
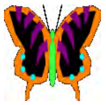
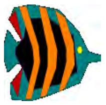
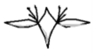
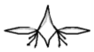
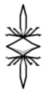
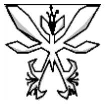
Enumeration Value	Description
	Left:  Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
clocks (Clocks Art Border)	Specifies an art border using the following images: All Sides: 
compass (Compass Art Border)	Specifies an art border using the following images: Top and Bottom:  Left and Right:  Top Left, Top Right, Bottom Left, and Bottom RIght: 
confetti (Confetti Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left: 

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	Right: Top Left: Top Right: Bottom Left: Bottom Right:
confettiGrays (Confetti Art Border)	Specifies an art border using the following images: All Sides: 
confettiOutline (Confetti Art Border)	Specifies an art border using the following images: All Sides: 
confettiStreamers (Confetti Streamers Art Border)	Specifies an art border using the following images: All Sides: 
confettiWhite (Confetti Art Border)	Specifies an art border using the following images: All Sides: 
cornerTriangles (Corner Triangle Art Border)	Specifies an art border using the following images: Top, Bottom, Left, and Right:  Top Left: 

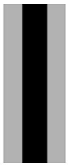
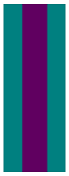
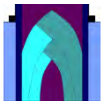
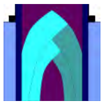
Enumeration Value	Description
	 Top Right: Bottom Left: Bottom Right:
couponCutoutDashes (Dashed Line Art Border)	Specifies an art border using the following images: Top, Bottom, Left, and Right:  Top Left: Top Right: Bottom Left: Bottom Right:
couponCutoutDots (Dotted Line Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left:

©ISO/IEC 2012 – All rights reserved

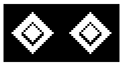
Enumeration Value	Description
	Top Right: Bottom Left: Bottom Right:
crazyMaze (Maze Art Border)	Specifies an art border using the following images: Top and Bottom:  Left and RIght:  Top Left, Top Right, Bottom Left and Bottom Right: 
creaturesButterfly (Butterfly Art Border)	Specifies an art border using the following images: All Sides: 
creaturesFish (Fish Art Border)	Specifies an art border using the following images: All Sides: 
creaturesInsects (Insects Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left and Right:  Top Left and Top Right: 

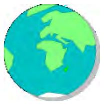
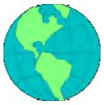
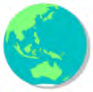
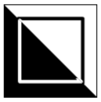
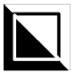
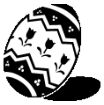
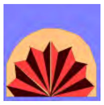
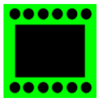
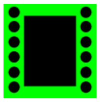
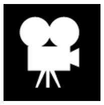
Enumeration Value	Description
	Bottom Left and Bottom Right:
creaturesLadyBug (Ladybug Art Border)	Specifies an art border using the following images: All Sides:
crossStitch (Cross-stitch Art Border)	Specifies an art border using the following images: All Sides:
cup (Cupid Art Border)	Specifies an art border using the following images: All Sides:
custom (Custom Defined Art Border)	Specifies a custom art border using the parent element's attributes to reference one or more customer art border images.
dashDotStroked (Dash Dot Strokes Line Border)	Specifies a line border consisting of a line with a series of alternating thin and thick strokes around the parent object. [*Example*: *end example*]
dashed (Dashed Line Border)	Specifies a line border consisting of a dashed line around the parent object. [*Example*: - - - - - - - - - - - - - - *end example*]
dashSmallGap (Dashed Line Border)	Specifies a line border consisting of a dashed line with small gaps around the parent object. [*Example*: -------------------------

©ISO/IEC 2012 – All rights reserved

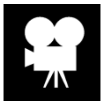
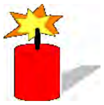
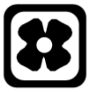
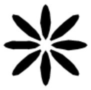
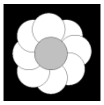
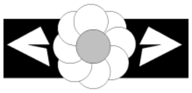
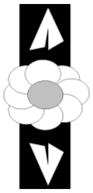
Enumeration Value	Description
	end example]
decoArch (Archway Art Border)	Specifies an art border using the following images: Top and Bottom:  Left and Right: Top Left and Top Right: Bottom Left and Bottom Right:
decoArchColor (Color Archway Art Border)	Specifies an art border using the following images: Top and Bottom:  Left and Right: Top Left:  Top Right:  Bottom Left and Bottom Right:
decoBlocks (Blocks Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right:

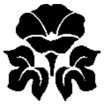
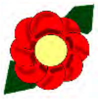
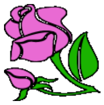
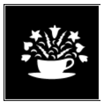
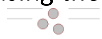
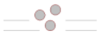
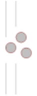
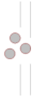
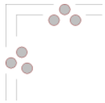
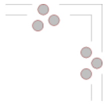
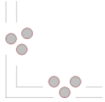
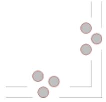
Enumeration Value	Description
	Top Left: Top Right: Bottom Left: Bottom Right:
diamondsGray (Gray Diamond Art Border)	Specifies an art border using the following images: All Sides:
dotDash (Dot Dash Line Border)	Specifies a line border consisting of a alternating dotted and dashed line around the parent object. [*Example*: -–-–-–-–-–-–-–-–-–- *end example*]
dotDotDash (Dot Dot Dash Line Border)	Specifies a line border consisting of a alternating dotted, dotted, dashed line around the parent object. [*Example*: --–--–--–--–--–--–--–--- *end example*]
dotted (Dotted Line Border)	Specifies a line border consisting of a dotted line around the parent object. [*Example*: .... *end example*]
double (Double Line Border)	Specifies a line border consisting of a double line around the parent object.

Enumeration Value	Description
	[*Example*: *end example*]
doubleD (Double D Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left: Top Right: Bottom Left: Bottom Right:
doubleDiamonds (Diamond Art Border)	Specifies an art border using the following images: All Sides: 
doubleWave (Double Wave Line Border)	Specifies a line border consisting of a double wavy line around the parent object. [*Example*: *end example*]

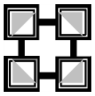
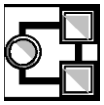
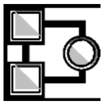
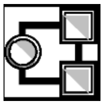
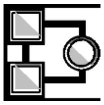
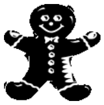
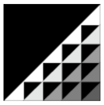
Enumeration Value	Description
earth1 (Earth Art Border)	Specifies an art border using the following images: All Sides: 
earth2 (Earth Art Border)	Specifies an art border using the following images: All Sides: 
earth3 (Earth Art Border)	Specifies an art border using the following images: All sides: 
eclipsingSquares1 (Shadowed Square Art Border)	Specifies an art border using the following images: All Sides: 
eclipsingSquares2 (Shadowed Square Art Border)	Specifies an art border using the following images: All Sides: 
eggsBlack (Painted Egg Art Border)	Specifies an art border using the following images: All Sides: 
fans (Fans Art Border)	Specifies an art border using the following images: All Sides: 
film (Film Reel Art Border)	Specifies an art border using the following images: Top and Bottom:  Left and Right:  Top Left and Bottom Left: 

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	Top Right and Bottom Right: 
firecrackers (Firecracker Art Border)	Specifies an art border using the following images: All Sides: 
flowersBlockPrint (Flowers Art Border)	Specifies an art border using the following images: All Sides: 
flowersDaisies (Daisy Art Border)	Specifies an art border using the following images: All Sides: 
flowersModern1 (Flowers Art Border)	Specifies an art border using the following images: All Sides: 
flowersModern2 (Flowers Art Border)	Specifies an art border using the following images: Top and Bottom:  Left and Right:  Top Left: Top Right: Bottom Left: Bottom Right:

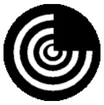
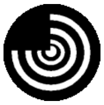
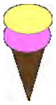
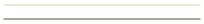
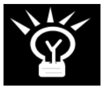
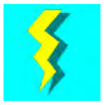
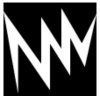
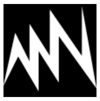
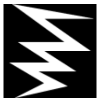
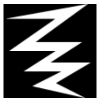
Enumeration Value	Description
flowersPansy (Pansy Art Border)	Specifies an art border using the following images: All Sides: 
flowersRedRose (Red Rose Art Border)	Specifies an art border using the following images: All Sides: 
flowersRoses (Roses Art Border)	Specifies an art border using the following images: All Sides: 
flowersTeacup (Flowers in a Teacup Art Border)	Specifies an art border using the following images: All Sides: 
flowersTiny (Small Flower Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 

©ISO/IEC 2012 – All rights reserved

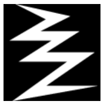
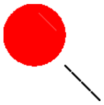
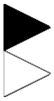
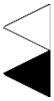
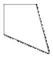
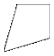
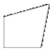
Enumeration Value	Description
gems (Gems Art Border)	Specifies an art border using the following images: Top, Bottom, Left and Right  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
gingerbreadMan (Gingerbread Man Art Border)	Specifies an art border using the following images: All Sides: 
gradient (Triangle Gradient Art Border)	Specifies an art border using the following images: All Sides: 
handmade1 (Handmade Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left:

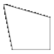
Enumeration Value	Description
	Top Right: Bottom Left: Bottom Right:
handmade2 (Handmade Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left: Top Right: Bottom Left: Bottom Right:
heartBalloon (Heart-Shaped Balloon Art Border)	Specifies an art border using the following images: All Sides:
heartGray (Gray Heart Art Border)	Specifies an art border using the following images:

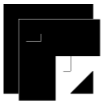
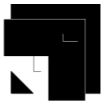
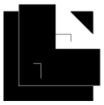
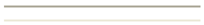
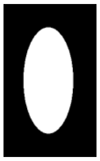
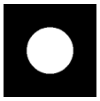
Enumeration Value	Description
	All Sides:
hearts (Hearts Art Border)	Specifies an art border using the following images: All Sides:
heebieJeebies (Pattern Art Border)	Specifies an art border using the following images: All Sides:
holly (Holly Art Border)	Specifies an art border using the following images: All Sides:
houseFunky (House Art Border)	Specifies an art border using the following images: All Sides:
hypnotic (Circular Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left: Top Right:

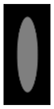
Enumeration Value	Description
	Bottom Left:  Bottom Right: 
iceCreamCones (Ice Cream Cone Art Border)	Specifies an art border using the following images: All Sides: 
inset (Inset Line Border)	Specifies a line border consisting of an inset set of lines around the parent object. [*Example*:  *end example*]
lightBulb (Light Bulb Art Border)	Specifies an art border using the following images: All Sides: 
lightning1 (Lightning Art Border)	Specifies an art border using the following images: All Sides: 
lightning2 (Lightning Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left and Top Left:  Right and Top RIght: 

©ISO/IEC 2012 – All rights reserved

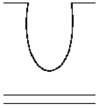
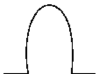
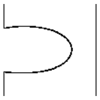
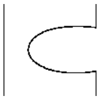
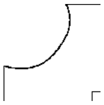
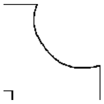
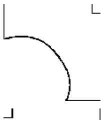
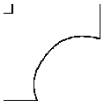
Enumeration Value	Description
	Bottom Left:  Bottom Right:
mapleLeaf (Maple Leaf Art Border)	Specifies an art border using the following images: All Sides:
mapleMuffins (Muffin Art Border)	Specifies an art border using the following images: All Sides:
mapPins (Map Pins Art Border)	Specifies an art border using the following images: All Sides: 
marquee (Marquee Art Border)	Specifies an art border using the following images: Top and Bottom: Left and Right: Top Left, Top Right, Bottom Left, and Bottom Right:
marqueeToothed (Marquee Art Border)	Specifies an art border using the following images: Top: Bottom: Left:  Right:  Top Left:  Top Right:  Bottom Left: 

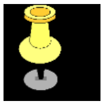
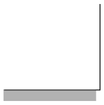
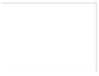
Enumeration Value	Description
	Bottom Right: 
moons (Moon Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left: Top Right: Bottom Left: Bottom Right:
mosaic (Mosaic Art Border)	Specifies an art border using the following images: All Sides:
musicNotes (Musical Note Art Border)	Specifies an art border using the following images: All Sides:
nil (No Border)	Specifies that no border shall be applied to the current item.
none (No Border)	Specifies that no border shall be applied to the current item.
northwest (Patterned Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right:

Enumeration Value	Description
	Top Left:  Top Right:  Bottom Left:  Bottom Right:
outset (Outset Line Border)	Specifies a line border consisting of an outset set of lines around the parent object. [*Example*:  *end example*]
ovals (Oval Art Border)	Specifies an art border using the following images: Top and Bottom: Right and Left:  Top Left, Top Right, Bottom Left and Bottom Right: 
packages (Package Art Border)	Specifies an art border using the following images: All Sides:
palmsBlack (Black Palm Tree Art Border)	Specifies an art border using the following images: All Sides:

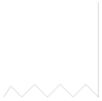
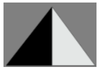
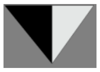
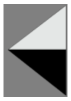
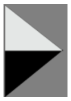
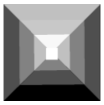
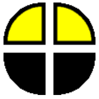
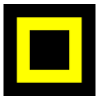
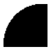
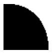
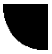
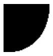
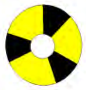
Enumeration Value	Description
palmsColor (Color Palm Tree Art Border)	Specifies an art border using the following images: All Sides:
paperClips (Paper Clip Art Border)	Specifies an art border using the following images: All Sides:
papyrus (Papyrus Art Border)	Specifies an art using the following images: Top and Bottom: Right and Left:  Top Left and Top Right: Bottom Left and Bottom Right:
partyFavor (Party Favor Art Border)	Specifies an art border using the following images: All Sides:
partyGlass (Party Glass Art Border)	Specifies an art border using the following images: All Sides:
pencils (Pencils Art Border)	Specifies an art border using the following images: All Sides:
people (Character Art Border)	Specifies an art border using the following images: All Sides:
peopleHats (Character With Hat Art Border)	Specifies an art border using the following images:

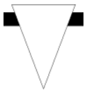
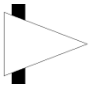
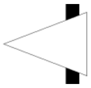
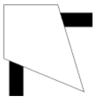
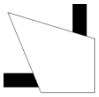
©ISO/IEC 2012 – All rights reserved

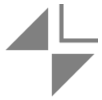
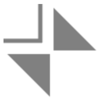
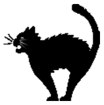
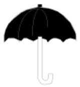
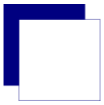
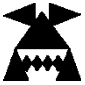
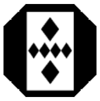
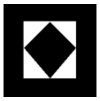
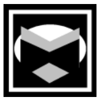
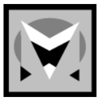
Enumeration Value	Description
	All Sides:
peopleWaving (Waving Character Border)	Specifies an art border using the following images: All Sides:
poinsettias (Poinsettia Art Border)	Specifies an art border using the following images: All Sides:
postageStamp (Postage Stamp Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
pumpkin1 (Pumpkin Art Border)	Specifies an art border using the following images: All Sides:

Enumeration Value	Description
pushPinNote1 (Push Pin Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left:  Top Right: Bottom Left: Bottom Right: 
pushPinNote2 (Push Pin Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left: Top Right:  Bottom Left:

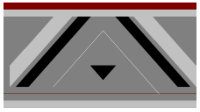
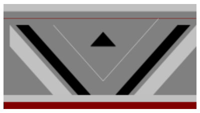
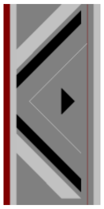
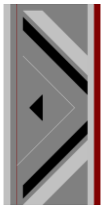
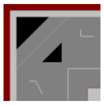
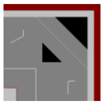
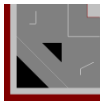
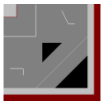
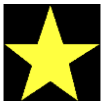
©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	Bottom Right: 
pyramids (Pyramid Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right:  Top Left, Top Right, Bottom Left, and Bottom RIght:
pyramidsAbove (Pyramid Art Border)	Specifies an art border using the following images: All Sides: 
quadrants (Quadrants Art Border)	Specifies an art border using the following images: Top and Bottom:  Left and RIght:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
rings (Rings Art Border)	Specifies an art border using the following images: All Sides: 
safari (Safari Art Border)	Specifies an art border using the following images:

Enumeration Value	Description
	All Sides:
sawtooth (Saw tooth Art Border)	Specifies an art border using the following images: Top:  Bottom: Left:  Right:  Top Left:  Top Right: Bottom Left: Bottom Right: 
sawtoothGray (Gray Saw tooth Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left: Top Right:

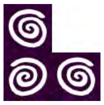
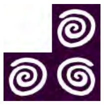
Enumeration Value	Description
	Bottom Left:  Bottom Right: 
scaredCat (Scared Cat Art Border)	Specifies an art border using the following images: All Sides: 
seattle (Umbrella Art Border)	Specifies an art border using the following images: All Sides: 
shadowedSquares (Shadowed Squares Art Border)	Specifies an art border using the following images: All Sides: 
shapes1 (Black and White Shapes Art Border)	Specifies an art border using the following images: Top and Bottom:  Right and Left:  Top Left, Top Right, Bottom Left and Bottom Right: 
shapes2 (Black and White Art Border Two)	Specifies an art border using the following images: Top, Bottom, Left and Right:  Top Left, Top Right, Bottom Left and Bottom Right: 
sharksTeeth (Shark Tooth Art Border)	Specifies an art border using the following images:

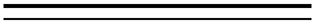
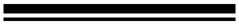
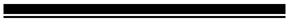
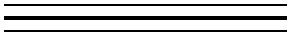
Enumeration Value	Description
	All Sides:
shorebirdTracks (Bird Tracks Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left, Top Right, Bottom Left and Bottom Right:
single (Single Line Border)	Specifies a line border consisting of a single line around the parent object. [Example: end example]
skyrocket (Rocket Art Border)	Specifies an art border using the following images: All Sides:
snowflakeFancy (Snowflake Art Border)	Specifies an art border using the following images: All Sides:
snowflakes (Snowflake Art Border)	Specifies an art border using the following images: All Sides:
sombrero (Sombrero Art Border)	Specifies an art border using the following images: All Sides:

Enumeration Value	Description
southwest (Southwest-themed Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
stars (Stars Art Border)	Specifies an art border using the following images: All Sides: 

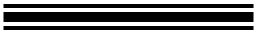
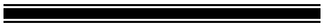
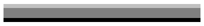
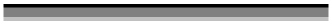
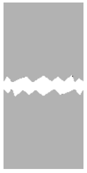
©ISO/IEC 2012 – All rights reserved

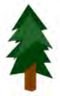
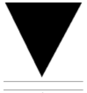
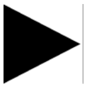
Enumeration Value	Description
stars3d (3-D Stars Art Border)	Specifies an art border using the following images: All Sides:
starsBlack (Stars Art Border)	Specifies an art border using the following images: All Sides:
starsShadowed (Stars With Shadows Art Border)	Specifies an art border using the following images: All Sides:
starsTop (Stars On Top Art Border)	Specifies an art border using the following images: Top, Top Left, and Top Right: Bottom, Bottom Left, and Bottom Right: Left and Right:
sun (Sun Art Border)	Specifies an art border using the following images: All Sides:
swirligig (Whirligig Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left: Top Right:

Enumeration Value	Description
	Bottom Left:  Bottom Right: 
thick (Single Line Border)	Specifies a line border consisting of a single line around the parent object. [*Example*: *end example*]
thickThinLargeGap (Thick, Thin Line Border)	Specifies a line border consisting of a thick line contained within a thin line with a large sized intermediate gap around the parent object. [*Example*: *end example*]
thickThinMediumGap (Thick, Thin Line Border)	Specifies a line border consisting of a thick line contained within a thin line with a medium sized intermediate gap around the parent object. [*Example*: *end example*]
thickThinSmallGap (Thick, Thin Line Border)	Specifies a line border consisting of a thick line contained within a thin line with a small intermediate gap around the parent object. [*Example*: *end example*]
thinThickLargeGap (Thin, Thick Line Border)	Specifies a line border consisting of a thin line contained within a thick line contained within a thick

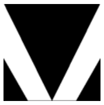
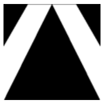
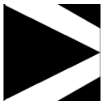
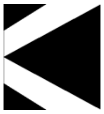
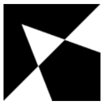
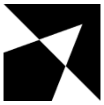
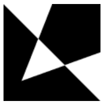
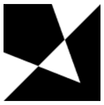
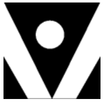
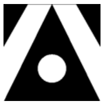
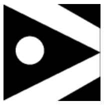
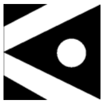
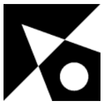
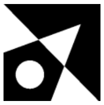
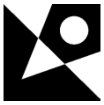
Enumeration Value	Description
	thin with a large sized intermediate gap between each around the parent object. [*Example*:  *end example*]
thinThickMediumGap (Thin, Thick Line Border)	Specifies a line border consisting of a thin line contained within a thick line contained within a thick thin with a medium sized intermediate gap between each around the parent object. [*Example*:  *end example*]
thinThickSmallGap (Thin, Thick Line Border)	Specifies a line border consisting of a thin line contained within a thick line contained within a thick thin with a small intermediate gap between each around the parent object. [*Example*:  *end example*]
thinThickThinLargeGap (Thin, Thick, Thin Line Border)	Specifies a line border consisting of a thin line contained within a thick line, contained within a thin line with a large sized intermediate gap around the parent object. [*Example*:  *end example*]
thinThickThinMediumGap (Thin, Thick, Thin Line Border)	Specifies a line border consisting of a thin line contained within a thick line, contained within a thin line with a medium sized intermediate gap around the parent object.

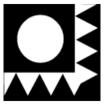
©ISO/IEC 2012 – All rights reserved

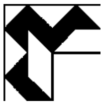
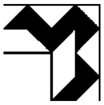
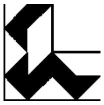
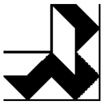
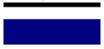
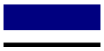
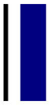
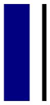
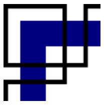
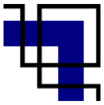
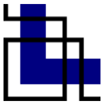
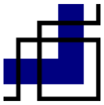
Enumeration Value	Description
	[*Example*:  *end example*]
thinThickThinSmallGap (Thin, Thick, Thin Line Border)	Specifies a line border consisting of a thin line contained within a thick line, contained within a thin line with a small intermediate gap around the parent object. [*Example*:  *end example*]
threeDEmboss (3D Embossed Line Border)	Specifies a line border consisting of three staged gradient lines around the parent object, getting darker towards the object. [*Example*:  *end example*]
threeDEngrave (3D Engraved Line Border)	Specifies a line border consisting of three staged gradient lines around the parent object, getting darker away from the object. [*Example*:  *end example*]
tornPaper (Torn Paper Art Border)	Specifies an art border using the following images: Top, Top Left and Top Right  Bottom, Bottom Left and Bottom RIght: Left and RIght: [*Note*: Image is blank. *end note*]

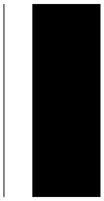
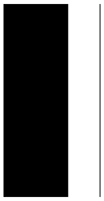
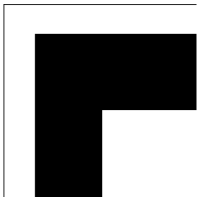
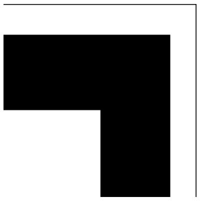
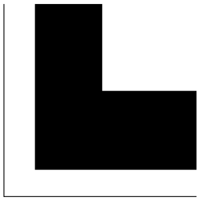
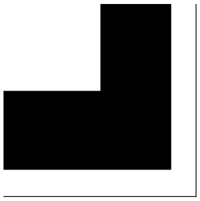
Enumeration Value	Description
tornPaperBlack (Black Torn Paper Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left: Top Right: Bottom Left: Bottom Right:
trees (Tree Art Border)	Specifies an art border using the following images: All Sides: 
triangle1 (Triangle Art Border One)	Specifies an art border using the following images: Top:  Bottom: Left:  Right: Top Left, Top Right, Bottom Left, and Bottom Right:
triangle2 (Triangle Art Border Two)	Specifies an art border using the following images:

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	Top:  Bottom:  Left:  Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
triangleCircle1 (Triangle and Circle Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right:  Top Left:  Top Right:  Bottom Left: 

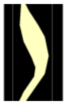
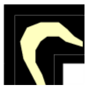
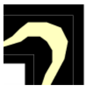
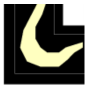
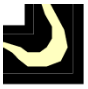
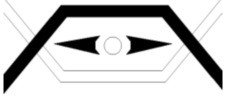
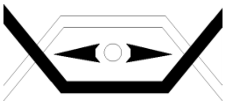
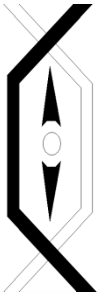
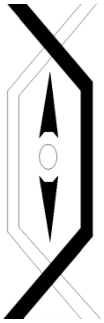
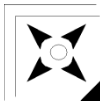
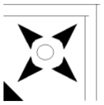
Enumeration Value	Description
	Bottom Right:
triangleCircle2 (Triangle and Circle Art Border Two)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left: Top Right: Bottom Left: Bottom Right: 
triangleParty (Triangle Art Border)	Specifies an art border using the following images: All Sides:
triangles (Triangles Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right:

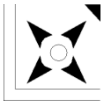
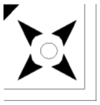
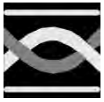
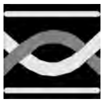
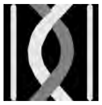
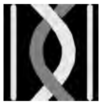
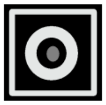
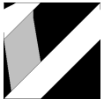
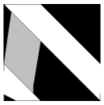
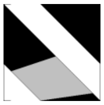
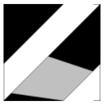
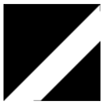
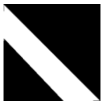
Enumeration Value	Description
	Top Left:  Top Right:  Bottom Left:  Bottom Right: 
triple (Triple Line Border)	Specifies a line border consisting of a triple line around the parent object. [*Example*: *end example*]
twistedLines1 (Twisted Lines Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 

Enumeration Value	Description
twistedLines2 (Twisted Lines Art Border)	Specifies an art border using the following images: Top: Bottom: Left:  Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
vine (Vine Art Border)	Specifies an art border using the following images:

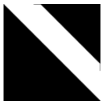
©ISO/IEC 2012 – All rights reserved

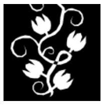
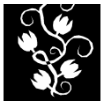
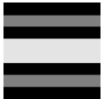
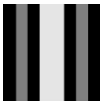
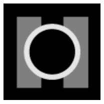
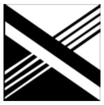
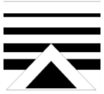
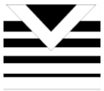
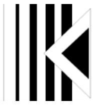
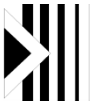
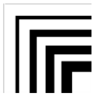
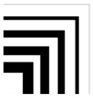
Enumeration Value	Description
	Top: Bottom: Left: Right: Top Left: Top Right: Bottom Left: Bottom Right:
wave (Wavy Line Border)	Specifies a line border consisting of a wavy line around the parent object. [*Example*: *end example*]
waveline (Wavy Line Art Border)	Specifies an art border using the following images: Top: Bottom: Left:

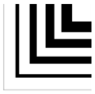
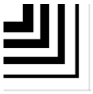
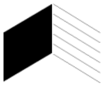
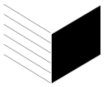
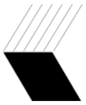
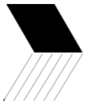
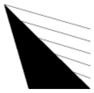
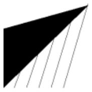
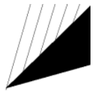
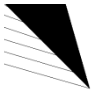
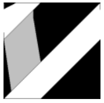
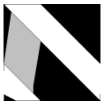
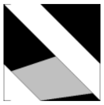
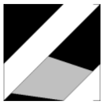
Enumeration Value	Description
	Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right: 
weavingAngles (Weaving Angles Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right:  Top Left:  Top Right: 

Enumeration Value	Description
	Bottom Left:  Bottom Right: 
weavingBraid (Weaving Braid Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right:  Top Left, Top Right, Bottom Left, and Botton RIght: 
weavingRibbon (Weaving Ribbon Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right:  Top Left:  Top Right: 

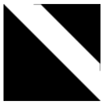
©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	Bottom Left:  Bottom Right:
weavingStrips (Weaving Strips Art Border)	Specifies an art border using the following images: Top: Bottom: Left: Right: Top Left: Top Right: Bottom Left: Bottom Right:
whiteFlowers (White Flowers Art Border)	Specifies an art border using the following images: Top and Bottom: Left: Right: Top Left: Top Right:

Enumeration Value	Description
	Bottom Left:  Bottom Right: 
woodwork (Woodwork Art Border)	Specifies an art border using the following images: Top and Bottom:  Left and Right:  Top Left, Top Right, Bottom Left, and Botton RIght: 
xIllusions (Crisscross Art Border)	Specifies an art border using the following images: All Sides: 
zanyTriangles (Triangle Art Border)	Specifies an art border using the following images: Top:  Bottom:  Left:  Right:  Top Left:  Top Right: 

Enumeration Value	Description
	 Bottom Left:  Bottom Right:
zigZag (Zigzag Art Border)	Specifies an art border using the following images:  Top:  Bottom:  Left:  Right:  Top Left:  Top Right:  Bottom Left:  Bottom Right:
zigZagStitch (Zigzag stitch)	Specifies an art border using the following images:  Top:  Bottom:  Left:  Right:

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	Top Left: Top Right: Bottom Left:  Bottom Right:

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Border) is located in §A.1. *end note*]

17.18.3 ST_BrClear (Line Break Text Wrapping Restart Location)

This simple type specifies the set of possible restart locations which can be used as to determine the next available line when a break's type attribute has a value of `textWrapping`. This property only affects the restart location when the current run is being displayed on a line which does not span the full text extents due to the presence of a floating object (see enumeration values for details).

[*Example*: Consider a text wrapping break character which should force the restart location to the next line which spans the full width of the text extents of the page (there are no floating objects which interrupt the line).

This line break is of style `textWrapping`, since it must only advance to the next line, but the clear value must specify that this restart location must ignore all lines which are not of the full line width by specifying a value of `all`, as follows:

```
<w:br w:type="textWrapping" w:clear="all" />
```

This break must therefore not use the next available line, but rather the next available line ignoring all lines which do not span the full text width. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
all (Restart On Next Full Line)	Specifies that the text wrapping break shall advance the text to the next line in the WordprocessingML document which spans the full width of the line (i.e. the next line which is not interrupted by any floating

Enumeration Value	Description
	objects when those objects are positioned on the page at display time. [*Note*: This setting is typically used to place a single line of text next to a floating object for use as a caption. *end note*]
left (Restart In Next Text Region Unblocked on the Left)	Specifies that the text wrapping break shall behave as follows when this line intersects a floating object: If the parent paragraph is left-to-right: • If this is the leftmost region of text flow currently on this line, • If a floating object occurs to the left of the break, advance the text to the next available line that does not have a floating object on the left. • Otherwise, advance the text to the next position on the line where text can be displayed • Otherwise, treat this as a text wrapping break of type none. If the parent paragraph is right to left: If the object occurs to the left of the break, advance the text to the next available line that does not have a floating object on the left. • Otherwise, treat this as a text wrapping break of type none. In either case, if this line does not intersect a floating object, then treat this break as a text wrapping break of type none.
none (Restart On Next Line)	Specifies that the text wrapping break shall advance the text to the next line in the WordprocessingML document, regardless of its position left to right or the presence of any floating objects which intersect with the line, This is the setting for a typical line break in a document.
right (Restart In Next Text Region Unblocked on the Right)	Specifies that the text wrapping break shall behave as follows when this line intersects a floating object:

Enumeration Value	Description
	If the parent paragraph is left-to-right: • If the object occurs to the right of the break, advance the text to the next available line that does not have a floating object on the right. • Otherwise, treat this as a text wrapping break of type none. If the parent paragraph is right to left: • If this is the rightmost region of text flow currently on this line, • If a floating object occurs to the right of the break, advance the text to the next available line that does not have a floating object on the right. • Otherwise, advance the text to the next position on the line where text can be displayed • Otherwise, treat this as a text wrapping break of type none. In either case, if this line does not intersect a floating object, then treat this break as a text wrapping break of type none.

[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_BrClear) is located in §A.1. *end note*]

17.18.4 ST_BrType (Break Types)

This simple type specifies the possible kinds of break characters in a WordprocessingML document. The break type determines the next location where text shall be placed after this manual break is applied to the text contents (see enumeration values for details).

[*Example*: Consider a manual break which must advance the text to the next text column in the document, rather than just the next available line. This break would therefore be specified as follows:

```
<w:br w:type="column"/>
```

The type attribute specifies a value of `column`, which means that the break must force the next character in the document to be restarted on the next line in a new text column in the document. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
column (Column Break)	Specifies that the current break shall restart itself on the next column available on the current page when the document is displayed in page view. If the current section is not divided into columns, or the column break occurs in the last column on the current page when displayed, then the restart location for text shall be the next page in the document.
page (Page Break)	Specifies that the current break shall restart itself on the next page of the document when the document is displayed in page view. Page breaks shall be ignored when present in frames unless the showBreaksInFrames element (Part 4, §9.7.3.36) is present in the document's compatibility settings.
textWrapping (Line Break)	Specifies that the current break shall restart itself on the next line in the document when the document is displayed in page view. The determine of the next line shall be done subject to the value of the clear attribute on the specified break character.

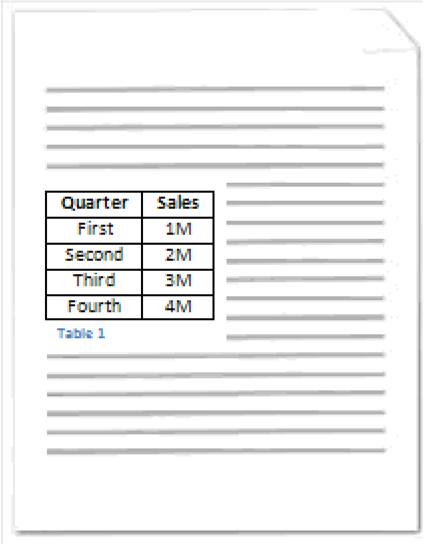
[*Note*: The W3C XML Schema definition of this simple type's content model (ST_BrType) is located in §A.1. *end note*]

17.18.5 ST_CaptionPos (Automatic Caption Positioning Values)

This simple type specifies the possible values can be used for the position of an automatically inserted caption on an object within this document. These values specify the position a given caption shall be take relative to the object it is used to label.

[*Example*: Consider a WordprocessingML document which should have all automatically inserted captions placed below the objects they are captioning, for example:

©ISO/IEC 2012 – All rights reserved



Quarter	Sales
First	1M
Second	2M
Third	3M
Fourth	4M

This requirement is specified using the following WordprocessingML in the document settings:

```
<w:captions>
  <w:caption w:name="Table" w:pos="below" w:numFmt="decimal" />
</w:captions>
```

The pos attribute has a value of `below`, specifying that the caption must be placed below the newly inserted objects. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
above (Position Caption Above Object)	Specifies that an automatically inserted caption shall be positioned above the object that it is used to label.
below (Position Caption Below Object)	Specifies that an automatically inserted caption shall be positioned below the object that it is used to label.
left (Position Caption Left Of Object)	Specifies that an automatically inserted caption shall be positioned to the left of the object that it is used to label (the position where text typed immediately before the object would appear).
right (Position Caption Right Of Object)	Specifies that an automatically inserted caption shall be positioned to the right of the object that it is used to label (the position where text typed immediately after the object would appear).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CaptionPos) is located in §A.1. *end note*]

17.18.6 ST_ChapterSep (Chapter Separator Types)

This simple type specifies the character which shall be used to separate the chapter number from the page number for page numbers in a given section, when chapter numbers are being displayed.

[*Example*: Consider a section in a document in which the chapter must be separated from the page number using a colon character. This constraint would be specified using the following WordprocessingML:

```
<w:pgNumType w:chapSep="colon" w:chapStyle="1" />
```

The chapSep attribute declares that the chapter and page number must be separated by a `colon` (e.g. 1:1 for chapter one, page one). *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
colon (Colon Chapter Separator)	Specifies that a colon character shall be used to separate the chapter number from the page number when page numbers are displayed. [Example: 1:1 for page one, section one. end example]
emDash (Em Dash Chapter Separator)	Specifies that an em dash character shall be used to separate the chapter number from the page number when page numbers are displayed. [*Example*: 1—1 for page one, section one. *end example*]
enDash (En Dash Chapter Separator)	Specifies that an en dash character shall be used to separate the chapter number from the page number when page numbers are displayed. [*Example*: 1–1 for page one, section one. *end example*]
hyphen (Hyphen Chapter Separator)	Specifies that a non-breaking hyphen character shall be used to separate the chapter number from the page number when page numbers are displayed. [*Example*: 1-1 for page one, section one. *end example*]
period (Period Chapter Separator)	Specifies that a period character shall be used to separate the chapter number from the page number when page numbers are displayed. [*Example*: 1.1 for page one, section one. *end example*]

©ISO/IEC 2012 – All rights reserved

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_ChapterSep) is located in §A.1. *end note*]

17.18.7 ST_CharacterSpacing (Character-Level Whitespace Compression Settings)

This simple type specifies the possible ways in which full-width characters in the current WordprocessingML document can be compressed to remove additional whitespace when the contents of this document are displayed, specifically by specifying the set(s) of characters which can be compressed to remove additional whitespace.

[*Example*: Consider the WordprocessingML below:

```
<w:characterSpacingControl w:val="doNotCompress" />
```

The characterSpacingControl element has a val attribute value of `doNotCompress`, which specifies that no character compression shall be applied to any character when the document is displayed. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
compressPunctuation (Compress Whitespace From Punctuation Characters)	Specifies that only whitespace characters shall have whitespace compression applied to them.
compressPunctuationAndJapaneseKana (Compress Whitespace From Both Japanese Kana And Punctuation Characters)	Specifies that whitespace and Japanese kana characters shall have whitespace compression applied to them.
doNotCompress (Do Not Compress Whitespace)	Specifies that characters shall not have whitespace compression applied to them.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CharacterSpacing) is located in §A.1. *end note*]

17.18.8 ST_CombineBrackets (Two Lines in One Enclosing Character Type)

This simple type specifies the kind of bracket character which shall be used to enclose the two lines in one text within the current run when displayed

[*Example*: Consider a paragraph with the text `two lines in one`, which must be displayed within a single logical line in the document and enclosed in curly brackets. This constraint would be specified as follows in the WordprocessingML:

```
<w:r>
  <w:rPr>
    <w:eastAsianLayout w:id="1" w:combine="on" w:combineBrackets="curly"/>
  </w:rPr>
```

```
    <w:t>two lines in one</w:t>
  </w:r>
```

The resulting text would be displayed on two sub lines within the other text on this line and enclosed within curly brackets when displayed. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
angle (Angle Brackets)	Specifies that angle bracket characters shall be used to enclose the contents of the current run's two lines in one text. [*Example*: <…> *end example*]
curly (Curly Brackets)	Specifies that curly bracket characters shall be used to enclose the contents of the current run's two lines in one text. [*Example*: {…} *end example*]
none (No Enclosing Brackets)	Specifies that no characters shall be used to enclose the contents of the current run's two lines in one text.
round (Round Brackets)	Specifies that round bracket characters shall be used to enclose the contents of the current run's two lines in one text. [*Example*: (…) *end example*]
square (Square Brackets)	Specifies that square bracket characters shall be used to enclose the contents of the current run's two lines in one text. [*Example*: […] *end example*]

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CombineBrackets) is located in §A.1. *end note*]

17.18.9 ST_DateTime (Standard Date and Time Storage Format)

This simple type specifies that its contents contain a date in the standard XML Schema xsd:dateTime format, whose contents are interpreted based on the context of the parent XML element.

[*Example*: Consider the following WordprocessingML fragment:

```
  <w:date w:fullDate="01-01-2006T12:00:00Z">
  …
</w:date>
```

In this case, the date in the realDate attribute is the full date associated with the parent date picker structured document. In every case, the value of this simple type is interpreted in the context of the parent element or attribute. *end example*]

This simple type's contents are a restriction of the W3C XML Schema dateTime datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DateTime) is located in §A.1. *end note*]

17.18.10 ST_DecimalNumber (Decimal Number Value)

This simple type specifies that its contents contain a whole decimal number (positive or negative), whose contents are interpreted based on the context of the parent XML element.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:pPr>
  <w:divId w:val="1512645511" />
</w:pPr>
```

The value of the val attribute is the ID of the associated HTML `div`.

However, consider the following fragment:

```
<w:ilvl w:val="1">
  …
</w:ilvl>
```

In this case, the decimal number in the val attribute is the ID of the associated numbering level. In each case, the value is interpreted in the context of the parent element. *end example*]

This simple type's contents are a restriction of the W3C XML Schema integer datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DecimalNumber) is located in §A.1. *end note*]

17.18.11 ST_DecimalNumberOrPercent (Percentage Measurement)

This simple type specifies that its contents will contain a percentage-based value. See the union's member types for details.

This simple type is a union of the following types:

- The ST_Percentage simple type (§22.9.2.9).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DecimalNumberOrPercent) is located in §A.1. *end note*]

17.18.12 ST_Direction (Bidirectional Direction Types)

This simple type specifies the possible values for bidirectional settings within a WordprocessingML document.

[*Example*: Consider the following bidirectional setting; in this case, a bidirectional override:

```
<w:bdo w:val="rtl">
  …
</w:bdo>
```

The val attribute explicitly declares that the override is applied right to left via the val attribute value of rtl. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

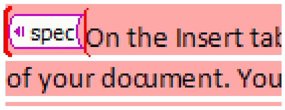
Enumeration Value	Description
ltr (Left toRight)	Specifies a left-to-right direction for the property defined by the parent XML element.
rtl (Right to Left)	Specifies a right-to-left direction for the property defined by the parent XML element.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Direction) is located in §A.1. *end note*]

17.18.13 ST_DisplacedByCustomXml (Location of Custom XML Markup Displacing an Annotation)

This simple type specifies the possible values for the location of a single custom XML element's start and/or end tag relative to the location of an annotation tag in document order. This enumeration shall be used to specify that the parent annotation's placement shall be directly linked with the location of the physical presentation of a custom XML element in the document.

[*Example*: Consider a paragraph with block level custom XML markup and two comment anchor annotations (one before and one after the custom XML element's physical representation), as follows:



Since all three of these items are around the entire paragraph, they are stored outside of the paragraph. However, in order to ensure that their relative positions are stored correctly, any annotation which must be displaced by the physical custom XML element specifies this information, resulting in the following WordprocessingML:

```
…
<w:commentRangeStart w:id="0" />
<w:commentRangeStart w:id="1" w:displaced byCustomXml="next" />
<w:customXml w:element="spec" … >
  <w:p>
    …
  </w:p>
</w:customXml>
…
```

The displacedByCustomXml attribute specifies that even though all three of these items are around the paragraph and is moved inside the paragraph to be represented physically, the comment with ID 0 must be inside the custom XML, but the comment with ID 1 must be displaced to stay outside of the relative location of the next custom XML element (the spec element). *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
next (Displaced by Next Custom XML Markup Tag)	Specifies that this annotation anchor shall be displaced by the physical representation of the next element of custom XML markup in the document. If no custom XML markup exists in the same paragraph and after this anchor, then this setting shall be ignored.
prev (Displaced by Previous Custom XML Markup Tag)	Specifies that this annotation anchor shall be displaced by the physical representation of the previous element of custom XML markup in the document. If no custom XML markup exists in the same paragraph and directly before this anchor, then this setting shall be ignored.

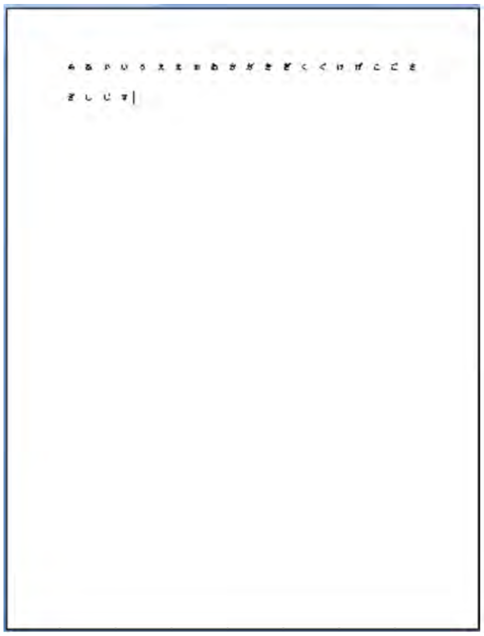
[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_DisplacedByCustomXml) is located in §A.1. *end note*]

17.18.14 ST_DocGrid (Document Grid Types)

Specifies the kind of the current document grid, which defines the grid behavior.

The grid can define a grid which snaps all East Asian characters to grid positions, but leaves Latin text with its default spacing; a grid which adds the specified character pitch to all characters on each row; or a grid which affects only the line pitch for the current section.

[*Example*: Consider the document discussed above with the document grid defined to allow 20 characters per line, and 20 lines per page by snapping characters to the grid as follows:



あぃいぅうぇえぉおかがきぎくぐけげこごさ
ざしじす

This document has a type attribute of type ST_DocGrid and value `snapToChars`, which specifies that the grid must force East Asian characters to fit 20 to a line. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
default (No Document Grid)	Specifies that no document grid shall be applied to the contents of the current section in the document.
lines (Line Grid Only)	Specifies that the parent section shall have additional line pitch added to each line within it (as specified on the docGrid element (§17.6.5)) in order to maintain the specified number of lines per page.
linesAndChars (Line and Character Grid)	Specifies that the parent section shall have both the additional line pitch and character pitch added to each line and character within it (as specified on the docGrid element (§17.6.5)) in order to maintain a specific number of lines per page and characters per line.

Enumeration Value	Description
	When this value is set, the input specified via the user interface can be allowed in exact number of line/character pitch units.
snapToChars (Character Grid Only)	Specifies that the parent section shall have both the additional line pitch and character pitch added to each line and character within it (as specified on the docGrid element (§17.6.5)) in order to maintain a specific number of lines per page and characters per line. When this value is set, the input specified via the user interface can be restricted to the number of lines per page and characters per line, with the consumer or producer translating this information based on the current font data to get the resulting line and character pitch values

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DocGrid) is located in §A.1. *end note*]

17.18.15 ST_DocPartBehavior (Insertion Behavior Types)

This simple type specifies the possible sets of behaviors which can be applied to the contents of a single glossary document entry (§17.12.5) when it is added to the main document story of a WordprocessingML document.

[*Example*: Consider the WordprocessingML fragment for a glossary document entry containing a single run, defined as follows:

```
<w:docPart>
  <w:docPartPr>
    <w:behaviors>
      <w:behavior w:val="p"/>
    </w:behaviors>
    …
  </w:docPartPr>
  <w:docPartBody>
    <w:p>
      <w:r>
        <w:t>Sample entry.</w:t>
      </w:r>
    </w:p>
  </w:docPartBody>
</w:docPart>
```

The behavior element of type ST_DocPartBehavior has a value of p, which specifies that the contents of the parent glossary document entry must be inserted in their own paragraph when they are added to the contents of a document. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
content (Insert Content At Specified Location)	Specifies that when the glossary document entry is inserted into the main document contents of the document, it shall be inserted normally as defined above. This includes ensuring that the final paragraph which is included in the part is not inserted, and its run content is added to the paragraph into which the current part is being inserted.
p (Ensure Entry Is In New Paragraph)	Specifies that the glossary document entry shall be added into its own unique paragraph, by failing to remove the last paragraph from the entry's contents when they are added to the document.
pg (Ensure Entry Is On New Page)	Specifies that the glossary document entry shall be added into its own new page, by preceding the entry with a blank paragraph whose only content is a page break character.

[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_DocPartBehavior) is located in §A.1. *end note*]

17.18.16 ST_DocPartGallery (Entry Gallery Types)

This simple type specifies possible settings for the predefined gallery into which a glossary document part shall be classified. This classification, although its enumeration values can be interpreted to imply semantics around the contents of the parent glossary document entry, shall only be used to classify and sort this entry (via an application or a user interface).

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:category>
    <w:name w:val="Internal Memo Covers" />
    <w:gallery w:val="coverPg" />
  </w:category>
  …
</w:docPartPr>
```

The gallery element with a value of coverPg specifies that the gallery categorization applied to the current entry, for the purposes of classification or user interface sorting, puts this entry into the Cover Pages classification. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
any (All Galleries)	Specifies that this glossary document entry shall be associated with all possible gallery classification values.
autoTxt (AutoText Gallery)	Specifies that this glossary document entry shall be associated with the AutoText gallery classification.
bib (Bibliography Gallery)	Specifies that this glossary document entry shall be associated with the Bibliography gallery classification.
coverPg (Cover Page Gallery)	Specifies that this glossary document entry shall be associated with the Cover Page gallery classification.
custAutoTxt (Custom AutoText Gallery)	Specifies that this glossary document entry shall be associated with the Custom AutoText gallery classification.
custBib (Custom Bibliography Gallery)	Specifies that this glossary document entry shall be associated with the Custom Bibliography gallery classification.
custCoverPg (Custom Cover Page Gallery)	Specifies that this glossary document entry shall be associated with the Custom Cover Page gallery classification.
custEq (Custom Equation Gallery)	Specifies that this glossary document entry shall be associated with the Custom Equation gallery classification.
custFtrs (Custom Footer Gallery)	Specifies that this glossary document entry shall be associated with the Custom Footer gallery classification.
custHdrs (Custom Header Gallery)	Specifies that this glossary document entry shall be associated with the Custom Header gallery classification.

Enumeration Value	Description
custom1 (Custom 1 Gallery)	Specifies that this glossary document entry shall be associated with the Custom 1 gallery classification.
custom2 (Custom 2 Gallery)	Specifies that this glossary document entry shall be associated with the Custom 2 gallery classification.
custom3 (Custom 3 Gallery)	Specifies that this glossary document entry shall be associated with the Custom 3 gallery classification.
custom4 (Custom 4 Gallery)	Specifies that this glossary document entry shall be associated with the Custom 4 gallery classification.
custom5 (Custom 5 Gallery)	Specifies that this glossary document entry shall be associated with the Custom 5 gallery classification.
custPgNum (Custom Page Number Gallery)	Specifies that this glossary document entry shall be associated with the Custom Page Number gallery classification.
custPgNumB (Custom Page Number At Bottom Gallery)	Specifies that this glossary document entry shall be associated with the Custom Page Number At Bottom gallery classification.
custPgNumMargins (Custom Page Number At Margins Gallery)	Specifies that this glossary document entry shall be associated with the Custom Page Number At Margins gallery classification.
custPgNumT (Custom Page Number At Top Gallery)	Specifies that this glossary document entry shall be associated with the Custom Page Number At Top gallery classification.
custQuickParts (Custom Quick Parts Gallery)	Specifies that this glossary document entry shall be associated with the Custom Quick Parts gallery classification.
custTblOfContents (Custom Table of Contents Gallery)	Specifies that this glossary document entry shall be associated with the Custom Table of Contents gallery classification.
custTbls (Custom Table Gallery)	Specifies that this glossary document entry shall be associated with the Custom Tables gallery classification.
custTxtBox (Custom Text Box Gallery)	Specifies that this glossary document entry shall be associated with the Custom Text Box gallery classification.
custWatermarks (Custom Watermark Gallery)	Specifies that this glossary document entry shall be associated with the Custom Watermark gallery classification.
default (No Gallery Classification)	Specifies that this glossary document entry shall not have a gallery classification.
docParts (Document Parts Gallery)	Specifies that this glossary document entry shall be associated with the Document Parts gallery

Enumeration Value	Description
	classification.
eq (Equations Gallery)	Specifies that this glossary document entry shall be associated with the Equations gallery classification.
ftrs (Footers Gallery)	Specifies that this glossary document entry shall be associated with the Footers gallery classification.
hdrs (Headers Gallery)	Specifies that this glossary document entry shall be associated with the Headers gallery classification.
pgNum (Page Numbers Gallery)	Specifies that this glossary document entry shall be associated with the Page Numbers gallery classification.
pgNumB (Page Numbers At Bottom Gallery)	Specifies that this glossary document entry shall be associated with the Page Numbers At Bottom gallery classification.
pgNumMargins (Page Numbers At Margins Gallery)	Specifies that this glossary document entry shall be associated with the Page Numbers At Margins gallery classification.
pgNumT (Page Numbers At Top Gallery)	Specifies that this glossary document entry shall be associated with the Page Numbers At Top gallery classification.
placeholder (Structured Document Tag Placeholder Text Gallery)	Specifies that this glossary document entry shall be associated with the Structured Document Tag Placeholder Text gallery classification.
tblOfContents (Table of Contents Gallery)	Specifies that this glossary document entry shall be associated with the Table of Contents gallery classification.
tbls (Table Gallery)	Specifies that this glossary document entry shall be associated with the Tables gallery classification.
txtBox (Text Box Gallery)	Specifies that this glossary document entry shall be associated with the Text Box gallery classification.
watermarks (Watermark Gallery)	Specifies that this glossary document entry shall be associated with the Watermark gallery classification.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DocPartGallery) is located in §A.1. *end note*]

17.18.17 ST_DocPartType (Entry Types)

This simple type specifies the possible entry types which can be applied to the properties of a single glossary document entry (§17.12.5). Each of these entry types can, based on their values, influence the visibility and behavior of the parent glossary document entry.

[*Example*: Consider the following WordprocessingML fragment for the properties of a single glossary document entry:

```
<w:docPartPr>
  <w:types>
    <w:type w:val="bbPlcHdr" />
  </w:types>
  …
</w:docPartPr>
```

The type element with a value of `bbPlcHdr` specifies that the parent glossary document entry must be treated as if it was the placeholder text for one or more structured document tags in the document. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
autoExp (Automatically Replace Name With Content)	Specifies that the type of the current glossary document entry shall allow the entry to be automatically inserted into the document whenever its name is entered into an application.
bbPlcHdr (Structured Document Tag Placeholder Text)	Specifies that the type of the current glossary document entry shall be structured document tag placeholder text.
formFld (Form Field Help Text)	Specifies that the type of the current glossary document entry shall be form field help text.
none (No Type)	Specifies no type information for the current glossary document entry.
normal (Normal)	Specifies that the type of the current glossary document entry shall be normal (i.e. a regular glossary document entry).
speller (AutoCorrect Entry)	Specifies that the type of the current glossary document entry shall be associated with the spelling and grammar tools.
toolbar (AutoText User Interface Entry)	Specifies that the type of the current glossary document entry shall be associated with a special grouping of entries associated with a single piece of user interface.

[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_DocPartType) is located in §A.1. *end note*]

17.18.18 ST_DocProtect (Document Protection Types)

This simple type specifies the possible set of editing restrictions which can be enforced on a given WordprocessingML document.

[*Example*: Consider a WordprocessingML document that contains the following WordprocessingML specifying that hosting applications must enforce read-only protection for a given document:

```
<w:documentProtection w:edit="readOnly" w:enforcement="1" />
```

The edit attribute has a value of `readOnly` and an enforcement attribute with a value of `1`, specifying that read-only document protection must be enforced on the given document. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
comments (Allow Editing of Comments)	Specifies that the edits made to this document shall be restricted to: • The insertion and deletion of comments within the document • The editing of regions delimited by range permissions which match the editing rights of the user account which is performing the editing.
forms (Allow Editing of Form Fields)	Specifies that the edits made to this document shall be restricted to: • The editing of form fields in sections where the formProt element (§17.6.6) has a value of `true`. • No restrictions in sections where the formProt element has a value of `false`.
none (No Editing Restrictions)	Specifies that no editing restrictions have been applied to the document.
readOnly (Allow No Editing)	Specifies that the edits made to this document shall be restricted to: • The editing of regions delimited by range permissions which match the editing rights of the user account which is performing the editing.
trackedChanges (Allow Editing With Revision Tracking)	Specifies that the edits made to this document shall be tracked as revisions. This value shall imply the presence of the trackRevisions element (§17.15.1.89), and applications shall not allow that element's state to be changed to `false`.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DocProtect) is located in §A.1. *end note*]

17.18.19 ST_DocType (Document Classification Values)

This simple type specifies the possible classifications that can be used for a WordprocessingML document.

The following possible values for this type are reserved:

Value	Description
eMail (E-Mail Message)	Specifies that this document shall be classified as an e-mail message.
letter (Letter)	Specifies that this document shall be classified as a letter.
notSpecified (Default Document)	Specifies that this document shall be classified as a default document.

[*Example*: Consider a set of WordprocessingML documents which should be classified as 'letters'. This classification would be specified using the following WordprocessingML in the document settings of these documents:

```
<w:documentType w:val="letter" />
```

The documentType element's val attribute is equal to `letter`, specifying that the hosting application must apply the behaviors it has specified for letters to the given WordprocessingML document. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DocType) is located in §A.1. *end note*]

17.18.20 ST_DropCap (Text Frame Drop Cap Location)

This simple type specifies the location which shall be used to position a drop cap text frame when the contents of that text frame are displayed in the anchor paragraph at display time.

[*Note*: Although a drop cap is simply a text frame, the values of this simple type are used to determine how the cap should be positioned relative to the following non-frame paragraph in relative terms (see enumeration values), rather than relying on absolute sizing. *end note*]

[*Example*: Consider the following paragraph containing a text frame which should be positioned as a drop cap:

```
<w:p>
  <w:pPr>
    <w:framePr w:dropCap="margin" w:lines="3" w:hSpace="432" w:wrap="around"
w:vAnchor="text" w:hAnchor="page" />
```

```
    </w:pPr>
    <w:r>
      <w:t>A</w:t>
    </w:r>
  </w:p>
```

The dropCap attribute specifies a value of `margin`, so this drop cap is placed outside of the text margin before the start of the current text. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
drop (Drop Cap Inside Margin)	Specifies that the drop cap text frame shall be positioned inside the text margin on the anchor paragraph when this text frame is displayed in the document.
margin (Drop Cap Outside Margin)	Specifies that the drop cap text frame shall be positioned outside of the text margin on the anchor paragraph when this text frame is displayed in the document.
none (Not Drop Cap)	Specifies that this text frame is not a drop cap text frame.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DropCap) is located in §A.1. *end note*]

17.18.21 ST_EdGrp (Range Permision Editing Group)

This simple type specifies the set of possible aliases (or editing groups) which can be used as aliases to determine if the current user shall be allowed to edit a single range defined by a range permission within a document. This mechanism simply provides a set of predefined editing groups which can be associated with user accounts by applications in any desired manner.

[*Example*: Consider a range permission defined as follows:

```
<w:permStart w:id="0" w:edGrp="editors" … />
…
<w:permEnd w:id="0" />
```

The edGrp attribute value of `editors` specifies that only user(s) who the current application associates with the editors group must be allowed to edit the contents between the start and end markers when document protection is being enforced. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
administrators (Administrator Group)	Specifies that users associated with the Administrators group shall be allowed to edit range permissions using this editing group when document protection is enabled.
contributors (Contributors Group)	Specifies that users associated with the Contributors group shall be allowed to edit range permissions using this editing group when document protection is enabled.
current (Current Group)	Specifies that users associated with the Current group shall be allowed to edit range permissions using this editing group when document protection is enabled.
editors (Editors Group)	Specifies that users associated with the Editors group shall be allowed to edit range permissions using this editing group when document protection is enabled.
everyone (All Users Have Editing Permissions)	Specifies that all users that open the document shall be allowed to edit range permissions using this editing group when document protection is enabled.
none (No Users Have Editing Permissions)	Specifies that none of the users that open the document shall be allowed to edit range permissions using this editing group when document protection is enabled.
owners (Owners Group)	Specifies that users associated with the Owners group shall be allowed to edit range permissions using this editing group when document protection is enabled.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_EdGrp) is located in §A.1. *end note*]

17.18.22 ST_EdnPos (Endnote Positioning Location)

This simple type specifies the possible positions of endnotes in a document.

[*Example*: Consider a document in which endnotes must be positioned at the end of the section. The section properties for this section must be declared as follows:

```
<w:settings>
  <w:endnotePr>
    <w:pos w:val="sectEnd" />
  </w:endnotePr>
  …
</w:settings>
```

©ISO/IEC 2012 – All rights reserved

The val attribute is sectEnd, therefore the position of endnotes is specified to be at the end of the section. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
docEnd (Endnotes Positioned at End of Document)	Specifies that all endnotes shall be placed at the end of the current document, regardless of which section they are referenced within.
sectEnd (Endnotes Positioned at End of Section)	Specifies that endnotes shall be placed at the end of the section in which they are referenced. An endnote which is never referenced is never displayed.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_EdnPos) is located in §A.1. *end note*]

17.18.23 ST_EighthPointMeasure (Measurement in Eighths of a Point)

This simple type specifies that its contents contain a positive whole number, whose contents consist of a measurement in eighths of a point (equivalent to 1/576th of an inch).

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of 24 whose simple type is ST_EighthPointMeasure. This attribute value specifies a size in eighths of a point (24 eighths of a point = 3 points). *end example*]

This simple type's contents are a restriction of the ST_UnsignedDecimalNumber datatype (§22.9.2.16).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_EighthPointMeasure) is located in §A.1. *end note*]

17.18.24 ST_Em (Emphasis Mark Type)

This simple type specifies possible types of emphasis marks which can be displayed for each non-space character in a run. This character is rendered above or below the character glyph as specified by enumeration values.

[*Example*: Consider a run of text which must have a dot underneath each character as an emphasis mark. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:em w:val="dot"/>
</w:rPr>
```

This run explicitly declares that the emphasis mark type is dot, so the contents of this run has a dot emphasis mark above each character. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
circle (Circle Emphasis Mark Above Characters)	Specifies that the emphasis mark is a circle character which shall be rendered above each character in this run using Unicode character 0x02DA when the language of the text is not Traditional Chinese. For that language Unicode character 0x3002 shall be used instead, positioned beneath the characters.
comma (Comma Emphasis Mark Above Characters)	Specifies that the emphasis mark is a comma character which shall be rendered above each character in this run, using Unicode character 0x3001.
dot (Dot Emphasis Mark Above Characters)	Specifies that the emphasis mark is a dot character which shall be rendered above each character in this run using Unicode character 0x02D9 whenever the language of the text is not Japanese, Simplified Chinese, or Traditional Chinese. For those three languages, the emphasis mark shall be rendered as follows: • Japanese = Unicode character 0xFF0E (dot beneath characters) • Simplified Chinese = Unicode character 0xFF0E (dot beneath characters) • Traditional Chinese = Unicode character 0x2027
none (No Emphasis Mark)	Specifies that there shall be no emphasis mark for any character in this run.
underDot (Dot Emphasis Mark Below Characters)	Specifies that the emphasis mark is a dot character which shall be rendered below each character in this run using Unicode character 0xFF0E.

[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_Em) is located in §A.1. *end note*]

17.18.25 ST_FFHelpTextVal (Help Text Value)

This simple type specifies the format of optional help text which can be associated with the parent form field.

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
<w:r>
  <w:fldChar w:fldCharType="begin">
    <w:ffData>
      <w:helpText w:type="text" w:val="Example help text."/>
      <w:ddList>
        …
      </w:ddList>
    </w:ffData>
  </w:fldChar>
</w:r>
<w:r>
  <w:instrText> FORMDROPDOWN </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

The helpText element specifies the help text for the parent form field - in this case, literal help text consisting of the string `Example help text.` *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a maximum length of 256 Unicode scalar values.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FFHelpTextVal) is located in §A.1. *end note*]

17.18.26 ST_FFName (Form Field Name Value)

This simple type specifies the format of the name which can be associated with the parent form field.

[*Example*: Consider the following WordprocessingML fragment for a text box form field:

```
<w:r>
  <w:fldChar w:fldCharType="begin">
    <w:ffData>
      <w:name w:val=”FirstName” />
      <w:textInput>
        …
      </w:textInput>
    </w:ffData>
  </w:fldChar>
</w:r>
```

```
<w:r>
  <w:instrText> FORMTEXT </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="separate"/>
</w:r>
<w:r>
  <w:t>1</w:t>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

The name element specifies that the name of the current form field is `FirstName`. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a maximum length of 65 Unicode scalar values.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FFName) is located in §A.1. *end note*]

17.18.27 ST_FFStatusTextVal (Status Text Value)

This simple type specifies the format of optional status text which can be associated with the parent form field.

[*Example*: Consider the following WordprocessingML fragment for a drop-down list form field:

```
<w:r>
  <w:fldChar w:fldCharType="begin">
    <w:ffData>
      <w:statusText w:type="text" w:val="Example status text."/>
      <w:ddList>
        …
      </w:ddList>
    </w:ffData>
  </w:fldChar>
</w:r>
<w:r>
  <w:instrText> FORMDROPDOWN </w:instrText>
</w:r>
<w:r>
  <w:fldChar w:fldCharType="end"/>
</w:r>
```

The statusText element specifies the status text for the parent form field - in this case, literal text consisting of the string `Example status text.` *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a maximum length of 140 characters.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FFStatusTextVal) is located in §A.1. *end note*]

17.18.28 ST_FFTextType (Text Box Form Field Type Values)

This simple type specifies the possible types of the contents of a text box form field.

[*Example*: Consider the following WordprocessingML fragment for the properties of a text box form field:

```
<w:ffData>
  <w:textInput>
    <w:type w:val="number" />
    <w:maxLength w:val="4" />
    <w:format w:val="0.00" />
  </w:textInput>
</w:ffData>
```

The type element specifies that the contents of this form field should be handled as a `number` by an application. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
calculated (Field Calculation)	Specifies that the contents of this text box form field shall be the result of the field calculation specified by the corresponding default element (§17.16.10). This field should not be directly editable when the editing of form fields is enabled.
currentDate (Current Date Display)	Specifies that the contents of this text box form field shall be the current date when the field is updated.
currentTime (Current Time Display)	Specifies that the contents of this text box form field shall be the current time when the field is updated.
date (Date)	Specifies that the contents of this text box form field shall be treated as a date.
number (Number)	Specifies that the contents of this text box form field

Enumeration Value	Description
	shall be treated as a number value.
regular (Text Box)	Specifies that this text form field is a plain text field (no additional content restrictions).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FFTextType) is located in §A.1. *end note*]

17.18.29 ST_FldCharType (Complex Field Character Type)

This simple type specifies the possible values for the type of a single complex field character in the document.

[*Example*: Consider the following WordprocessingML for a complex field character:

```
…
<w:fldChar w:fldCharType="separate" />
…
```

The type attribute value of `separate` specifies that this is a complex field separator character; therefore it is being used to separate the field codes from the field contents in a complex field. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
begin (Start Character)	Specifies that the character is a start character, which defines the start of a complex field.
end (End Character)	Specifies that the character is an end character, which defines the end of a complex field.
separate (Separator Character)	Specifies that the character is a separator character, which defines the end of the field codes and the start of the field result for a complex field.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FldCharType) is located in §A.1. *end note*]

17.18.30 ST_FontFamily (Font Family Value)

This simple type specifies possible values for the font family of a font.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Calibri">
  <w:family w:val="swiss" />
  …
</w:font>
```

The family element specifies via its val attribute value of `swiss` that this font is part of the Swiss family. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
auto (No Font Family)	Specifies that information about a font's font family does not exist.
decorative (Novelty Font)	Specifies the Novelty font family.
modern (Monospace Font)	Specifies a monospace font with or without serifs (monospace fonts are usually modern).
roman (Proportional Font With Serifs)	Specifies a proportional font with serifs.
script (Script Font)	Specifies a script font designed to mimic the appearance of handwriting.
swiss (Proportional Font Without Serifs)	Specifies a proportional font without serifs.

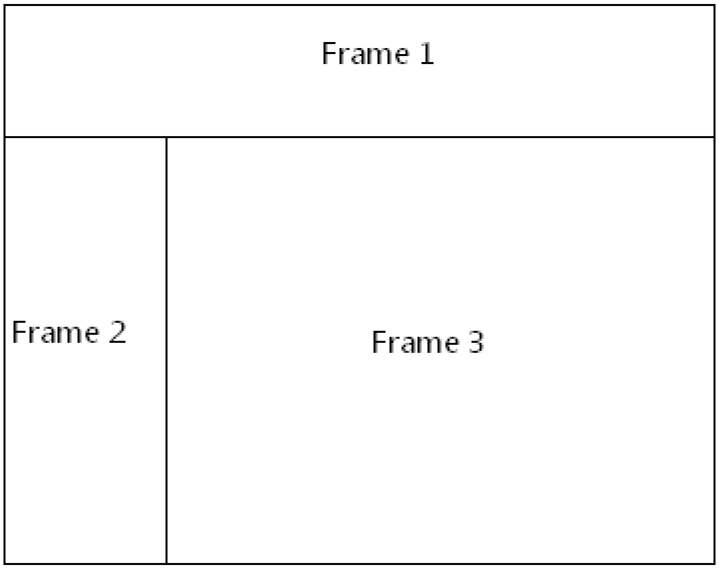
[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_FontFamily) is located in §A.1. *end note*]

17.18.31 ST_FrameLayout (Frameset Layout Order)

This simple type specifies the possible order in which the frames (and nested framesets) in a frameset can be displayed. When a frameset is created, it can only contain frames which are stacked in one direction:

- Vertically (one on top of another)
- Horizontally (one next to another)

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  <w:frameLayout w:val="rows" />
  <w:frame>
    …
  </w:frame>
  <w:frameset>
    <w:frameLayout w:val="cols" />
    <w:frame>
      …
    </w:frame>
    <w:frame>
      …
    </w:frame>
  </w:frameset>
</w:frameset>
```

The frameLayout element specifies that the outer frameset is consists of the single frame and the child frameset stacked vertically, and an inner nested frameset consisting of two frames stacked horizontally. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
cols (Stack Frames Horizontally)	Specifies that the frames in the frameset shall be

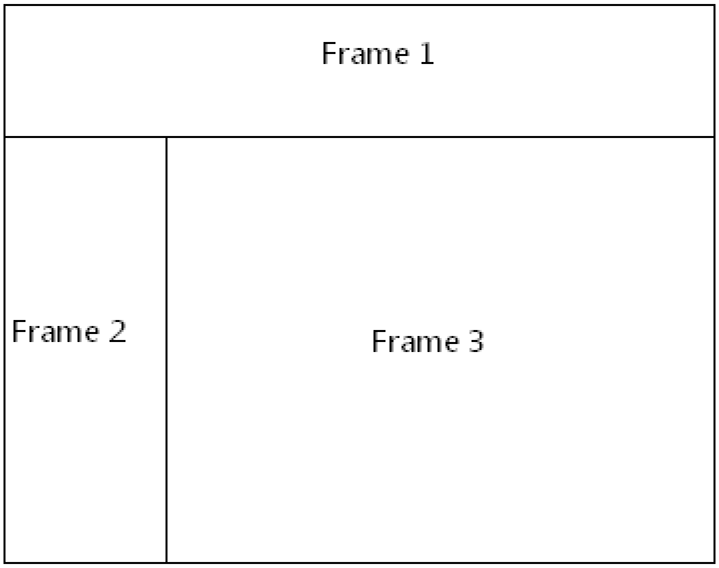
Enumeration Value	Description
	stacked horizontally next to each other in left to right order.
none (Do Not Stack Frames)	Specifies that no frames shall be shown in the frameset.
rows (Stack Frames Vertically)	Specifies that the frames in the frameset shall be stacked vertically next to each other in top to bottom order.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FrameLayout) is located in §A.1. *end note*]

17.18.32 ST_FrameScrollbar (Frame Scrollbar Visibility)

This simple type specifies the possible settings for when a scrollbar shall be visible for the contents of the current frame.

[*Example*: Consider a WordprocessingML document which serves as the frameset container for a frameset consisting of the following three frames:



The frameset properties for this document are specified by the following WordprocessingML within the web page settings:

```
<w:frameset>
  …
  <w:frameset>
```

```
      …
      <w:frame>
        <w:name w:val="Frame 2" />
        <w:scrollbar w:val="auto" />
      </w:frame>
      …
    </w:frameset>
  </w:frameset>
```

The scrollbar element has a val attribute of auto, which specifies that the frame must only display a scrollbar when it is needed to display all of its content. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

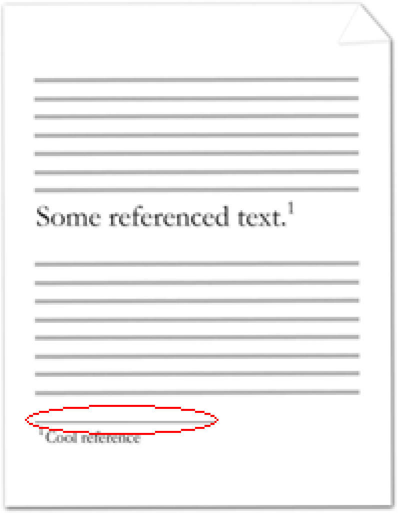
Enumeration Value	Description
auto (Automatically Show Scrollbar As Needed)	Specifies that the scrollbar for a frame shall automatically be hidden and/or displayed as needed based on the length of the contents.
off (Never Show Scrollbar)	Specifies that the scrollbar for a frame shall always be hidden.
on (Always Show Scrollbar)	Specifies that the scrollbar for a frame shall always be displayed (even when not needed).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FrameScrollbar) is located in §A.1. *end note*]

17.18.33 ST_FtnEdn (Footnote or Endnote Type)

This simple type specifies the possible types of footnotes and endnotes which can be specified in a WordprocessingML document.

[*Example*: Consider a document with a single footnote at the bottom of the first page. This footnote must be separated from the text by the separator footnote (the footnote explicitly used to separate text from the footnote list (circled in red below):



This footnote type would be declared as follows in the WordprocessingML:

```
<w:footnote w:type="separator" w:id="0">
  …
</w:footnote>
```

In this example, the footnote has an attribute value of `separator`, specifies when this footnote must be used. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
continuationNotice (Continuation Notice Separator)	Specifies that this footnote or endnote is a continuation notice footnote or endnote. *Continuation notice footnotes and endnotes* are used when the footnotes or endnotes exceed the length allowed on a single page. When this happens, this footnote or endnote shall be placed on the bottom of each page where the note shall continue to indicate that fact to the reader.
continuationSeparator (Continuation Separator)	Specifies that this footnote or endnote is a continuation separator footnote or endnote. *Continuation separator footnotes and endnotes* are used when the footnotes or endnotes exceed the length allowed on a single page. When this happens, this footnote or endnote shall be placed between the main text contents and the continued footnotes/endnotes on all subsequent pages of the

Enumeration Value	Description
	document.
normal (Normal Footnote/Endnote)	Specifies that this footnote or endnote is a normal footnote or endnote, and can be referenced by main document content.
separator (Separator)	Specifies that this footnote or endnote is a separator footnote or endnote. *Separator footnotes and endnotes* are used to indicate the separation between the main document's content and the footnotes or endnotes to indicate that fact to the reader.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FtnEdn) is located in §A.1. *end note*]

17.18.34 ST_FtnPos (Footnote Positioning Location)

This simple type specifies the position of footnotes in the document.

[*Example*: Consider a document in which footnotes must be positioned beneath their text. The footnote properties for this document must be declared as follows:

```
<w:sectPr>
  <w:footnotePr>
    <w:pos w:val="beneathText" />
  </w:footnotePr>
  …
</w:sectPr>
```

The val attribute is `beneathText`, therefore the position of footnotes is specified to be beneath the page's text. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
beneathText (Footnotes Positioned Beneath Text)	Specifies that footnotes shall be displayed immediately after the last line of text on the page on which the note reference mark appears.
docEnd (Footnotes Positioned At End of Document)	Specifies that all footnotes shall be placed at the end of the current document, regardless of which section they are referenced within.
pageBottom (Footnotes Positioned at Page Bottom)	Specifies that footnotes shall be displayed at the

Enumeration Value	Description
	bottom margin of the page on which the note reference mark appears.
sectEnd (Footnotes Positioned At End of Section)	Specifies that all footnotes shall be placed at the end of the section in which they are referenced. A footnote which is never referenced is never displayed.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FtnPos) is located in §A.1. *end note*]

17.18.35 ST_HAnchor (Horizontal Anchor Location)

This simple type specifies the horizontal position to which the parent object has been anchored in the document. This anchor position shall be used as the base location to determine the final horizontal position of the object in the document.

[*Example*: Consider a text frame which should be positioned one inch to the right of its column in a left-to-right document. This text frame would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:framePr … w:x="1440" w:hAnchor="margin" />
</w:pPr>
```

These frame horizontal anchor properties specify that they are relative to the anchor paragraph's `margin` (the text margin excluding any indents). *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
margin (Relative To Margin)	Specifies that the parent object shall be horizontally anchored to the text margins. This shall be used to specify that any horizontal positioning values shall be calculated with respect to the location of the text margin.
page (Relative to Page)	Specifies that the parent object shall be horizontally anchored to the page edge. This shall be used to specify that any horizontal positioning values shall be calculated with respect to the location of the edge of the page.

Enumeration Value	Description
text (Relative to Text Extents)	Specifies that the parent object shall be horizontally anchored to the text extents. This shall be used to specify that any horizontal positioning values shall be calculated with respect to the location of the edge of the text in the anchor paragraph (including text indentations on that paragraph within the text margins).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_HAnchor) is located in §A.1. *end note*]

17.18.36 ST_HdrFtr (Header or Footer Type)

This simple type specifies the possible types of headers and footers which can be specified for a given header or footer reference in a document. This value determines the page(s) on which the current header or footer shall be displayed.

[*Example*: Consider a WordprocessingML section which specifies the following header reference:

```
<w:headerReference r:id="rId10" w:type="first" />
```

The resulting section must use the specified header part for the `first` page. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
default (Default Header or Footer)	Specifies that this header or footer shall appear on every page in this section which is not overridden with a specific even or `first` page header/footer. In a section with all three types specified, this header/footer type shall be used on all odd numbered pages (counting from the first page in the section, not the section numbering).
even (Even Numbered Pages Only)	Specifies that this header or footer shall appear on all even numbered pages in this section (counting from the first page in the section, not the section numbering). The appearance of this header or footer is contingent on the setting of the evenAndOddHeaders element (§17.10.1).

Enumeration Value	Description
first (First Page Only)	Specifies that this header or footer shall appear on the first page in this section. The appearance of this header or footer is contingent on the setting of the titlePg element (§17.10.6).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_HdrFtr) is located in §A.1. *end note*]

17.18.37 ST_HeightRule (Height Rule)

This simple type specifies the logic which shall be used to calculate the height of the parent object when it is displayed in the document.

[*Example*: Consider the following table row:

```
<w:trPr>
  <w:trHeight w:hRule="atLeast" w:val="2189" />
</w:trPr>
```

The val attribute specifies a value of 2189 twentieths of a point, so this table row is a minimum of 2189 twentieths of a point high regardless of its contents, since its hRule value is set to `atLeast`. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
atLeast (Minimum Height)	Specifies that the height of the parent object shall be at least the value specified, but can be expanded to fit its content as needed.
auto (Determine Height Based On Contents)	Specifies that the height of the parent object shall be automatically determined by the size of its contents, with no predetermined minimum or maximum size.
exact (Exact Height)	Specifies that the height of the parent object shall be exactly the value specified, regardless of the size of the contents of the object. If the contents are too large for the specified height, then they shall be clipped.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_HeightRule) is located in §A.1. *end note*]

17.18.38 ST_HexColor (Color Value)

This simple type specifies that its contents contain one of the following:

- A color values in RRGGBB format (ST_HexColorRGB)
- The enumeration value auto (ST_HexColorAuto)

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider a border color with value auto, as follows:

```
<w:bottom … w:color="auto"/>
```

This color therefore can be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*]

This simple type is a union of the following types:

- The ST_HexColorRGB simple type (§22.9.2.5).
- The ST_HexColorAuto simple type (§17.18.39).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_HexColor) is located in §A.1. *end note*]

17.18.39 ST_HexColorAuto ('Automatic' Color Value)

This simple type specifies that its contents contain the enumeration value auto. This value shall be used to specify an automatically determined color value, the meaning of which is interpreted based on the context of the parent XML element.

[*Example*: Consider a border color with value auto, as follows:

```
<w:bottom … w:color="auto"/>
```

This color therefore can be automatically be modified by a consumer as appropriate, for example, in order to ensure that the border can be distinguished against the page's background color. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
auto (Automatically Determined Color)	Specifies that the color value can automatically be defined when this document is processed, based on the display context.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_HexColorAuto) is located in §A.1. *end note*]

17.18.40 ST_HighlightColor (Text Highlight Colors)

This simple type specifies the possible values for highlighting colors which can be applied as a background behind the contents of a text run.

[*Example*: Consider a run within a paragraph which has yellow text highlighting using the highlight element. This formatting is specified using the following WordprocessingML:

```
<w:rPr>
  <w:highlight w:val="yellow" />
</w:rPr>
```

The resulting run would have `yellow` highlighting visible over its contents. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
black (Black Highlighting Color)	Specifies that the text highlighting color for this run shall be black. The hexadecimal RGB value for this setting shall be `000000`.
blue (Blue Highlighting Color)	Specifies that the text highlighting color for this run shall be blue. The hexadecimal RGB value for this setting shall be `0000FF`.
cyan (Cyan Highlighting Color)	Specifies that the text highlighting color for this run shall be cyan. The hexadecimal RGB value for this setting shall be `00FFFF`.
darkBlue (Dark Blue Highlighting Color)	Specifies that the text highlighting color for this run shall be dark blue. The hexadecimal RGB value for this setting shall be `00008B`.
darkCyan (Dark Cyan Highlighting Color)	Specifies that the text highlighting color for this run shall be dark cyan. The hexadecimal RGB value for this setting shall be `008B8B`.
darkGray (Dark Gray Highlighting Color)	Specifies that the text highlighting color for this run shall be dark gray.

Enumeration Value	Description
	The hexadecimal RGB value for this setting shall be `A9A9A9`.
darkGreen (Dark Green Highlighting Color)	Specifies that the text highlighting color for this run shall be dark green. The hexadecimal RGB value for this setting shall be `006400`.
darkMagenta (Dark Magenta Highlighting Color)	Specifies that the text highlighting color for this run shall be dark magenta. The hexadecimal RGB value for this setting shall be `800080`.
darkRed (Dark Red Highlighting Color)	Specifies that the text highlighting color for this run shall be dark red. The hexadecimal RGB value for this setting shall be `8B0000`.
darkYellow (Dark Yellow Highlighting Color)	Specifies that the text highlighting color for this run shall be dark cyan. The hexadecimal RGB value for this setting shall be `808000`.
green (Green Highlighting Color)	Specifies that the text highlighting color for this run shall be green. The hexadecimal RGB value for this setting shall be `00FF00`.
lightGray (Light Gray Highlighting Color)	Specifies that the text highlighting color for this run shall be light gray. The hexadecimal RGB value for this setting shall be `D3D3D3`.
magenta (Magenta Highlighting Color)	Specifies that the text highlighting color for this run shall be magenta. The hexadecimal RGB value for this setting shall be `FF00FF`.
none (No Text Highlighting)	Specifies that this text run shall have no text highlighting applied to its contents.
red (Red Highlighting Color)	Specifies that the text highlighting color for this run shall be red.

Enumeration Value	Description
	The hexadecimal RGB value for this setting shall be `FF0000`.
white (White Highlighting Color)	Specifies that the text highlighting color for this run shall be white. The hexadecimal RGB value for this setting shall be `FFFFFF`.
yellow (Yellow Highlighting Color)	Specifies that the text highlighting color for this run shall be yellow. The hexadecimal RGB value for this setting shall be `FFFF00`.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_HighlightColor) is located in §A.1. *end note*]

17.18.41 ST_Hint (Font Type Hint)

Specifies the font type which shall be used to format any ambiguous characters in the current run.

There are certain characters which are not explicitly stored in the document, and can be mapped into multiple categories of the four mentioned above. This attribute shall be used to arbitrate that conflict, and determine how ambiguities in this run shall be handled. [*Note*: This is primarily used to handle the formatting on the paragraph mark glyph, and other characters that are not stored as text in the WordprocessingML document. *end note*]

[*Example*: Consider the run representing the paragraph mark glyph, which is not stored as a physical character. Since this could therefore be formatted with any of the fonts specified for the run, this ambiguity is resolved using the following WordprocessingML:

```
<w:pPr>
  <w:rPr>
    <w:rFonts w:hint="eastAsia" />
  </w:rPr>
</w:pPr>
```

The hint attribute specifies that the run must use the `eastAsia` font (theme or not, whichever is in use for East Asian text) as applied for this run. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
cs (Complex Script Font)	Specifies that the font hint for this text run shall be to use the Complex Script font defined on the run via the style hierarchy.
default (High ANSI Font)	Specifies that the font hint for this text run shall be to use the High ANSI font defined on the run via the style hierarchy.
eastAsia (East Asian Font)	Specifies that the font hint for this text run shall be to use the East Asian font defined on the run via the style hierarchy.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Hint) is located in §A.1. *end note*]

17.18.42 ST_HpsMeasure (Measurement in Half-Points)

This simple type specifies that its contents contain either:

- A positive whole number, whose contents consist of a measurement in half-points (equivalent to 1/144th of an inch), or
- A positive decimal number immediately followed by a unit identifier.

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of 72 whose simple type is ST_HpsMeasure. This attribute value specifies a size of one-half of an inch or 36 points (72 halves of a point = 36 points = 0.5 inches). *end example*]

[*Example*: Consider an attribute value of `12.7mm` whose type is ST_HpsMeasure. This attribute value specifies a size of 0.0127 meter or one-half of an inch or 36 points. *end example*]

This simple type is a union of the following types:

- The ST_PositiveUniversalMeasure simple type (§22.9.2.12).
- The ST_UnsignedDecimalNumber simple type (§22.9.2.16).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_HpsMeasure) is located in §A.1. *end note*]

17.18.43 ST_InfoTextType (Help or Status Text Type)

This simple type specifies the possible values for the type of help or status text which can be associated with a form field.

[*Example*: Consider the following WordprocessingML fragment for a form field:

```
<w:ffData>
  <w:helpText w:type="text" w:val="Example help text." />
</w:ffData>
```

The type attribute has a value of `text`, which specifies that the text in the val attribute is the literal help text for this form field. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
autoText (Glossary Document Entry)	Specifies that the value specified by the parent XML element's val attribute shall be interpreted as the name of a glossary document entry whose contents contain the help or status text.
text (Literal Text)	Specifies that the value specified by the parent XML element's val attribute shall be interpreted as the literal text for the help or status text.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_InfoTextType) is located in §A.1. *end note*]

17.18.44 ST_Jc (Horizontal Alignment Type)

This simple type specifies all types of alignment which are available to be applied to objects in a WordprocessingML document.

[*Example*: Consider a paragraph which is aligned to the trailing edge of text flow. This requirement would be specified as follows in the WordprocessingML markup:

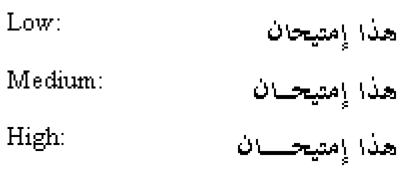
```
<w:pPr>
  <w:jc w:val="end" />
</w:pPr>
```

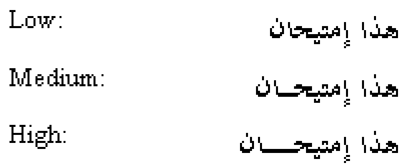
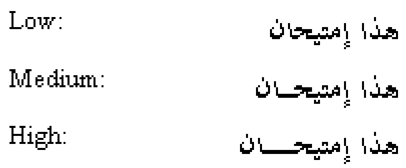
The val attribute's value of end specifies that the content must be right-aligned on the page for a left-to-right paragraph, and left justfied for a right-to-left paragraph. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
both (Justified)	Specifies that the text shall be justified between both of the text margins in the document. The `lowKashida` setting shall also be applied to Arabic

Enumeration Value	Description
	text when this setting is applied. This type of justification shall only affect the inter-word spacing on each line, and not the inter-character spacing between each word when justifying its contents.
center (Align Center)	Specifies that the text shall be centered on the line between both of the text margins in the document.
distribute (Distribute All Characters Equally)	Specifies that the text shall be justified between both of the text margins in the document. This type of justification shall equally affect the inter-word spacing on each line as well as the inter-character spacing between each word when justifying its contents - that is, an equal amount of additional character pitch shall be added to all characters on the line.
end (Align to Trailing Edge)	Specifies that the text shall be aligned on the trailing text margin in the document (right for left-to-right paragraphs; left for right-to-left paragraphs).
highKashida (Widest Kashida Length)	Specifies that the kashida length for text in the current paragraph shall be extended to its widest possible length. This setting only affects *kashidas*, which are special characters used to extend the joiner between two Arabic characters. [*Note*: They are typically used to improve the appearance of justified text by visually lengthening words rather than increasing the spacing between words. *end note*] [*Example*: The following example illustrates each type of kashida:  Low: هذا إمتحان Medium: هذا إمتحـان High: هذا إمتحــــان *end example*]
lowKashida (Low Kashida Length)	Specifies that the kashida length for text in the current paragraph shall be extended to a slightly longer length. This setting shall also be applied to Arabic text when

Enumeration Value	Description
	the both setting is applied. This setting only affects *kashidas*, which are special characters used to extend the joiner between two Arabic characters. [*Note*: They are typically used to improve the appearance of justified text by visually lengthening words rather than increasing the spacing between words. *end note*] [*Example*: The following example illustrates each type of kashida:  Low: هذا إمتيحان Medium: هذا إمتيحـان High: هذا إمتيحــــان *end example*]
mediumKashida (Medium Kashida Length)	Specifies that the kashida length for text in the current paragraph shall be extended to a medium length determined by the consumer. This setting only affects *kashidas*, which are special characters used to extend the joiner between two Arabic characters. [*Note*: They are typically used to improve the appearance of justified text by visually lengthening words rather than increasing the spacing between words. *end note*] [*Example*: The following example illustrates each type of kashida:  Low: هذا إمتيحان Medium: هذا إمتيحـان High: هذا إمتيحــــان *end example*]
numTab (Align to List Tab)	Specifies that the text shall be aligned to the list tab, which is the tab stop after the numbering for the current paragraph. If the current paragraph has no numbering, this setting has no effect.

Enumeration Value	Description
	[*Note*: This justification style is used for backwards compatibility with earlier word processors, and should be avoided in favor of hanging paragraph indentation. *end note*]
start (Align To Leading Edge)	Specifies that the text shall be aligned on the leading text margin in the document (left for left-to-right paragraphs; right for right-to-left paragraphs).
thaiDistribute (Thai Language Justification)	Specifies that the text shall be justified with an optimization for Thai. This type of justification shall affect both the inter-word spacing on each line, and the inter-character spacing between each word when justifying its contents, unlike both justification. This difference is created in that the inter-character space is increased slightly in order to ensure that the additional space created by the justification is reduced. [*Note*: This setting is different from justification in that the reduction in inter-character spacing would be inappropriate in Western languages. *end note*]

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Jc) is located in §A.1. *end note*]

17.18.45 ST_JcTable (Table Alignment Type)

This simple type specifies all types of alignment that are available to be applied to tables in a WordprocessingML document.

[*Example*: Consider a table row that is right aligned. This requirement would be specified as follows in the WordprocessingML markup:

```
<w:trPr>
  <w:jc w:val="end" />
</w:trPr>
```

The val attribute's value of end specifies that the table is right-aligned on the page (assuming the table is aligned left-to-right. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

©ISO/IEC 2012 – All rights reserved

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
center (Align Center)	Specifies that the table shall be centered on the line between both of the text margins in the document.
end (Align to Trailing Edge)	Specifies that the table shall be aligned to the trailing edge of the text flow – the right text margin (for a left-to-right table); or the left text margin (for a right-to-left table) in the document. (See §17.4.1)
start (Align to Starting Edge)	Specifies that the table shall be aligned to the leading edge of the text flow – the left text margin (for a left-to-right table); or the right text margin (for a right-to-left table) in the document. (See §17.4.1)

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_JcTable) is located in §A.1. *end note*]

17.18.46 ST_LevelSuffix (Content Between Numbering Symbol and Paragraph Text)

This simple type specifies the types of content which shall be possible between a given numbering level's text and the text of every numbered paragraph which references that numbering level.

[*Example:* Consider the numbered paragraph below:

1. Test

In this example, a space exists between the numbering symbol `1.` and the numbered paragraph text `Test`. The space would be specified in WordprocessingML as follows:

```
<w:lvl w:ilvl="0">
  …
  <w:suff w:val="space" />
  …
</w:lvl>
```

The suff element with an attribute value of space specifies that the character between the numbering's level text and the paragraph text must be a space. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
nothing (Nothing Between Numbering and Text)	Specifies that no character shall be displayed between the numbering level's text and the contents of the paragraph when displaying the numbered paragraph.

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
space (Space Between Numbering and Text)	Specifies that a space character shall be displayed between the numbering level's text and the contents of the paragraph when displaying the numbered paragraph.
tab (Tab Between Numbering and Text)	Specifies that a tab character shall be displayed between the numbering level's text and the contents of the paragraph when displaying the numbered paragraph. This tab shall follow normal tab stop rules to determine its length.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_LevelSuffix) is located in §A.1. *end note*]

17.18.47 ST_LineNumberRestart (Line Numbering Restart Position)

This simple type specifies when the line numbering in the parent section shall be reset to its restart value. The line numbering increments for each line (even if the line number itself is not displayed) until it reaches the restart point specified by this element.

[*Example*: Consider the line numbering used on each page of this document, which specifies that line numbering must restart at the top of each new page. This line numbering setting would be defined using the following WordprocessingML:

```
<w:lnNumType w:restart="newPage" … />
```

The restart attribute is of type ST_LineNumberRestart, and a value of newPage specifies that the line numbers must restart at the top of each page to the value specified by the start attribute. In this case, newPage is the default, so this value could have been omitted entirely. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
continuous (Continue Line Numbering From Previous Section)	Specifies that line numbering for the parent section shall continue from the line numbering from the end of the previous section, if any.
newPage (Restart Line Numbering on Each Page)	Specifies that line numbering for the parent section shall restart to the starting value whenever a new page is displayed.
newSection (Restart Line Numbering for Each Section)	Specifies that line numbering for the parent section shall restart to the starting value whenever the parent

Enumeration Value	Description
	begins.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_LineNumberRestart) is located in §A.1. *end note*]

17.18.48 ST_LineSpacingRule (Line Spacing Rule)

This simple type specifies the logic which shall be used to calculate the line spacing of the parent object when it is displayed in the document.

[*Example*: Consider the following WordprocessingML paragraph:

```
<w:pPr>
  <w:spacing w:line="276" w:lineRule="auto" />
</w:pPr>
```

This paragraph specifies that the spacing in each line should be automatically calculated using 1.15 times (276 divided by 240) the normal single spacing calculation. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
atLeast (Minimum Line Height)	Specifies that the height of the line shall be at least the value specified, but might be expanded to fit its content as needed.
auto (Automatically Determined Line Height)	Specifies that the line spacing of the parent object shall be automatically determined by the size of its contents, with no predetermined minimum or maximum size.
exact (Exact Line Height)	Specifies that the height of the line shall be exactly the value specified, regardless of the size of the contents of the contents. If the contents are too large for the specified height, then they shall be clipped as necessary.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_LineSpacingRule) is located in §A.1. *end note*]

17.18.49 ST_Lock (Locking Types)

This simple type specifies the possible set of locking behaviors that can be applied to the contents of the nearest ancestor structured document tag when the contents of this document are edited by an application (whether through a user interface or directly).

[*Example*: Consider the following plain text structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:lock w:val="sdtLocked"/>
    …
    <w:text/>
  </w:sdtPr>
  …
</w:sdt>
```

This plain text structured document tag's properties contain a lock element, specifying locking behaviors for the structured document tag. Since the locking val attribute value is sdtLocked, this locking setting must specify that the contents of the structured document tag can be edited, but the structured document tag itself must not be deleted from the document. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
contentLocked (Contents Cannot Be Edited At Runtime)	Specifies that the editing restriction applied to the nearest ancestor structured document tag shall be as follows: • This structured document tag's contents shall not be editable • This structured document tag can be deleted in its entirety (but only entirely, no sub portion of it can be deleted)
sdtContentLocked (Contents Cannot Be Edited At Runtime And SDT Cannot Be Deleted)	Specifies that the editing restriction applied to the nearest ancestor structured document tag shall be as follows: • This structured document tag's contents shall not be editable • This structured document tag shall not be deleted in its entirety
sdtLocked (SDT Cannot Be Deleted)	Specifies that the editing restriction applied to the nearest ancestor structured document tag shall be as follows: • This structured document tag's contents shall

Enumeration Value	Description
	be editable - This structured document tag shall not be deleted in its entirety
unlocked (No Locking)	Specifies that no special locking behaviors shall be applied to the nearest ancestor structured document tag. The default behaviors as specified on the lock element (§17.5.2.23) shall be used.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Lock) is located in §A.1. *end note*]

17.18.50 ST_LongHexNumber (Eight Digit Hexadecimal Value)

This simple type specifies a number value specified as a four octet (eight digit) hexadecimal number, whose contents are interpreted based on the context of the parent XML element.

[*Example*: Consider the following value for a node of simple type ST_LongHexNumber: `00BE2C6C`.

This value is permitted, as it contains four hexadecimal octets, each an encoding of an octet of the actual decimal number value. *end example*]

This simple type's contents are a restriction of the W3C XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a length of exactly 8 hexadecimal digit(s).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_LongHexNumber) is located in §A.1. *end note*]

17.18.51 ST_MacroName (Script Subroutine Name Value)

This simple type specifies a subroutine in a scripting language which can be executed based on the context of the parent XML element. The language and location of this subroutine can be determined using any method desired by an application.

[*Example*: Consider the following WordprocessingML fragment for the properties of a form field:

```
<w:ffData>
  <w:exitMacro w:val="TestExitFunction" />
</w:ffData>
```

The exitMacro element specifies that any application which processes this file should attempt to locate and execute a scripting subroutine called `TestExitFunction` when the contents of the form field are exited. If this subroutine cannot be located or executed, then this setting is silently ignored. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a maximum length of 33 Unicode scalar values.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_MacroName) is located in §A.1. *end note*]

17.18.52 ST_MailMergeDataType (Mail Merge Data Source Type Values)

This simple type specifies the data source access mechanism used to connect to the data source for a mail merge. This setting is purely a suggestion of the data source access mechanism which shall be used, and can be ignored in favor of an alternative mechanism if one is present.

Although this any access mechanism value can be specified, the following values are reserved:

Value	Description
native (Office Data Source Object Data Source)	Specifies that a given merged WordprocessingML document has been connected to an external data source via the data stored in the Office Data Source Object (ODSO) interface (§17.14.25).
odbc (Open Database Connectivity Data Source)	Specifies that a given merged WordprocessingML document has been connected to an external data source via the Open Database Connectivity interface.
query (Query Data Source)	Specifies that a given merged WordprocessingML document has been connected to an external data source using an external query file.
soap (SOAP Data Source)	Specifies that a given WordprocessingML document has been connected to a data source using SOAP.
spreadsheet (Spreadsheet Data Source)	Specifies that a given WordprocessingML document has been connected to a spreadsheet.
textFile (Text File Data Source)	Specifies that a given WordprocessingML document has been connected to a text file.
xQuery (Text File Data Source)	Specifies that a given WordprocessingML document has been connected to an external data source via XQuery.
xmlFile (XML File Data Source)	Specifies that a given WordprocessingML document has been connected to an XML file.

[*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document:

```
<w:dataType w:val="odbc" />
```

The dataType element's val attribute is equal to odbc, specifying that the given merged WordprocessingML document has been connected to an external data source via the Open Database Connectivity interface. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_MailMergeDataType) is located in §A.1. *end note*]

17.18.53 ST_MailMergeDest (Merged Document Destination Types)

This simple type specifies the possible results which can be generated when a mail merge is carried out on a given WordprocessingML source document. In other words, this element is used to specify what is to be done with the merged documents that result from populating the fields within a given merged WordprocessingML document with data from the specified external data source.

[*Example*: Consider a WordprocessingML source document containing the following WordprocessingML:

```
<w:mailMerge>
  <w:destination w:val="newDocument" />
  …
</w:mailMerge>
```

The destination element's val attribute is set to `newDocument`, specifying that when the mail merge is carried out, the source document must be used to generate a specified number of new documents, which can be handled as appropriate. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
email (Send Merged Documents as E-mail Messages)	Specifies that conforming hosting applications shall generate emails using the documents that result from populating the fields within a given merged WordprocessingML document with data from the specified external data source.
fax (Send Merged Documents as Faxes)	Specifies that conforming hosting applications shall generate faxes using the documents that result from populating the fields within a given merged WordprocessingML document with data from the specified external data source.
newDocument (Send Merged Documents to New	Specifies that conforming hosting applications shall

Enumeration Value	Description
Documents)	generate new documents by populating the fields within a given merged WordprocessingML document with data from the specified external data source.
printer (Send Merged Documents to Printer)	Specifies that conforming hosting applications shall print the documents that result from populating the fields within a given merged WordprocessingML document with external data from the specified external data source.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_MailMergeDest) is located in §A.1. *end note*]

17.18.54 ST_MailMergeDocType (Source Document Types)

This simple types specifies the possible types for a given WordprocessingML source document.

[*Example*: Consider the WordprocessingML below:

```
<w:mailMerge>
  <w:mainDocumentType w:val="formLetters" />
  …
</w:mailMerge>
```

In this example, the source document is of the `formLetters` type, as specified by the mainDocumentType element's val attribute being equal to `formLetters`. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
catalog (Catalog Source Document)	Specifies that the mail merge source document is of the catalog type.
email (E-Mail Source Document)	Specifies that the mail merge source document is of the e-mail message type.
envelopes (Envelope Source Document)	Specifies that the mail merge source document is of the envelope type.
fax (Fax Source Document)	Specifies that the mail merge source document is of the fax type.
formLetters (Form Letter Source Document)	Specifies that the mail merge source document is of the form letter type.
mailingLabels (Mailing Label Source Document)	Specifies that the mail merge source document is of the mailing label type.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_MailMergeDocType) is located in §A.1. *end note*]

17.18.55 ST_MailMergeOdsoFMDFieldType (Merge Field Mapping Types)

This simple types specifies the possible types used to indicate if a given mail merge field has been mapped to a column in the given external data source.

[*Example*: Consider the WordprocessingML below:

```
<w:odso>
  …
  <w:fieldMapData>
    <w:type w:val="dbColumn" />
    <w:name w:val="Country" />
    <w:mappedName w:val="Country or Region" />
    <w:column w:val="9" />
    …
  </w:fieldMapData>
</w:odso>
```

In this example, the `country` column within the given external data source must be mapped to the mail merge field `Country or Region`, as specified by the type element's val attribute being equal to `dbColumn`. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
dbColumn (Field Mapping to Data Source Column)	Specifies that the mail merge field has been mapped to a column in the given external data source.
null (Field Not Mapped)	Specifies that the mail merge field has not been mapped to a column in the given external data source.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_MailMergeOdsoFMDFieldType) is located in §A.1. *end note*]

17.18.56 ST_MailMergeSourceType (Mail Merge ODSO Data Source Types)

This simple type specifies the type of external data source to be connected to via as part of the ODSO connection information for this mail merge. This setting is purely a suggestion of the data source type which is being used for this mail merge, and can be ignored in favor of an alternative mechanism if one is present.

[*Example*: Consider the following WordprocessingML fragment for a mail merge source or merged document:

```
<w:type w:val="database" />
```

The type element's val attribute is equal to database, specifying that the given merged WordprocessingML document has been connected to an external data source via the ODSO settings, and that the resulting data source was a database. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
addressBook (Address Book Data Source)	Specifies that a given merged WordprocesingML document has been connected to an address book of contacts.
database (Database Data Source)	Specifies that a given merged WordprocessingML document has been connected to a database.
document1 (Alternate Document Format Data Source)	Specifies that a given merged WordprocessingML document has been connected to another document format supported by the producing application. The format of this document is application-defined and outside the scope of ISO/IEC 29500.
document2 (Alternate Document Format Data Source Two)	Specifies that a given merged WordprocessingML document has been connected to another document format supported by the producing application. The format of this document is application-defined and outside the scope of ISO/IEC 29500.
email (E-Mail Program Data Source)	Specifies that a given merged WordprocessingML document has been connected to an e-mail application.
legacy (Legacy Document Format Data Source)	Specifies that a given merged WordprocessingML document has been connected to a legacy document format supported by the producing application. The format of this legacy document is application-defined and outside the scope of ISO/IEC 29500.
master (Aggregate Data Source)	Specifies that a given merged WordprocessingML document has been connected to a data source which aggregates other data sources.
native (Native Data Souce)	Specifies that a given merged WordprocessingML document has been connected to another document format native to the producing application. The format of this document is application-defined and outside the scope of ISO/IEC 29500.
text (Text File Data Source)	Specifies that a given merged WordprocessingML

Enumeration Value	Description
	document has been connected to a text file.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_MailMergeSourceType) is located in §A.1. *end note*]

17.18.57 ST_Merge (Merged Cell Type)

This element specifies the way in which a cell shall be included in a merged group of cells (horizontally or vertically) within the parent table.

[*Example*: Consider a table with three rows and two columns with the last column completely vertically merged:

The second cell in the first row starts a vertical merge that is completed in the last cell, resulting in the following WordprocessingML:

```
<w:tbl>
  …
  <w:tr>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      …
    </w:tc>
    <w:tc>
      <w:tcPr>
        <w:vMerge w:val="restart"/>
      </w:tcPr>
      …
    </w:tc>
  </w:tr>
```

```
    <w:tr>
      <w:tc>
        …
      </w:tc>
      <w:tc>
        …
      </w:tc>
      <w:tc>
        <w:tcPr>
          <w:vMerge w:val="continue"/>
        </w:tcPr>
        …
      </w:tc>
    </w:tr>
    <w:tr>
      <w:tc>
        …
      </w:tc>
      <w:tc>
        …
      </w:tc>
      <w:tc>
        <w:tcPr>
          <w:vMerge w:val="continue"/>
        </w:tcPr>
        …
      </w:tc>
    </w:tr>
  </w:tbl>
```

The val attribute of type ST_Merge on the vMerge element defines the cells that are vertically merged, and how each cell is merged together. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
continue (Continue Merged Region)	Specifies that the current cell continues a previously existing merged group of cells in the parent table. If the previous cell in the document (horizontally or vertically) does not either begin or continue a set of merged cells, then this value shall be ignored (i.e. a group of merged cells shall start with a merge whose

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	ST_Merge value is restart).
restart (Start/Restart Merged Region)	Specifies that the current cell starts (or restarts) a group of merged cells in the parent table. After this value, all following cells which have a value of continue shall be merged into this merged cell group.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Merge) is located in §A.1. *end note*]

17.18.58 ST_MultiLevelType (Numbering Definition Type)

This simple type specifies the possible types of numbering which can be defined by a given abstract numbering type. This information shall only be used by a consumer to determine user interface behaviors for this numbering definition, and shall not be used to limit the behavior of the list (i.e. a list with multiple levels marked as singleLevel shall not be prevented from using levels 2 through 9).

[*Example*: Consider the WordprocessingML below:

```
<w:abstractNum w:abstractNumId="8">
  …
  <w:multiLevelType w:val="singleLevel" />
  …
</w:abstractNum>
```

This abstract numbering definition is specified to be of the singleLevel type by the multiLevelType element. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
hybridMultilevel (Hybrid Multilevel Numbering Definition)	Specifies that this numbering definition defines a numbering format consisting of a multiple levels, each of a potentially different kind (bullets vs. level text).
multilevel (Multilevel Numbering Definition)	Specifies that this numbering definition defines a numbering format consisting of a multiple levels, each of the same kind (bullets vs. level text).
singleLevel (Single Level Numbering Definition)	Specifies that this numbering definition defines a numbering format consisting of a single level only.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_MultiLevelType) is located in §A.1. *end note*]

17.18.59 ST_NumberFormat (Numbering Format)

This simple type specifies the numbering format which shall be used for a group of automatically numbered objects,

[*Example*: A value of `lowerLetter` for page numbering indicates that a consumer must use lowercase letters for each page in this section: a,b,c… *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
aiueo (AIUEO Order Half-Width Katakana)	Specifies that the sequence shall consist of one or more occurrences of a single half-width Katakana character from the set listed below, in the traditional a-i-u-e-o order. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–46 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–46 is U+FF71–U+FF9C, U+FF66, and U+FF9D, respectively. For values greater than 46, the sequence begins again, iterating through the same 46 values, repeating this pattern as necessary. [*Example*: The numbering for these items should be represented by the following pattern: ｱ, ｲ, ｳ, …, ｦ, ﾝ, ｱｱ, ｲｲ, ｳｳ, … *end example*]
aiueoFullWidth (AIUEO Order Full-Width Katakana)	Specifies that the sequence shall consist of one or more occurrences of a single full-width katakana character from the set listed below, in the traditional a-i-u-e-o order. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–46 and then repeats those same characters using the logic defined below to construct all other values.

Enumeration Value	Description
	The set of characters used by this numbering format for values 1–46 is U+30A2, U+30A4, U+30A6, U+30A8, U+30AA, U+30AB, U+30AD, U+30AF, U+30B1, U+30B3, U+30B5, U+30B7, U+30B9, U+30BB, U+30BD, U+30BF,U+30C1,U+30C4, U+30C6, U+30C8, U+30CA, U+30CB, U+30CC, U+30CD, U+30CE, U+30CF, U+30D2, U+30D5, U+30D8, U+30DB, U+30DE, U+30DF,U+30E0, U+30E1, U+30E2, U+30E4, U+30E6, U+30E8, U+30E9, U+30EA, U+30EB, U+30EC, U+30ED,U+30EF, U+30F0, U+30F1, U+30F2, and U+30F3, respectively. For values greater than 46, the sequence begins again, iterating through the same 46 values, repeating this pattern as necessary. [*Example*: The numbering for the items should be represented by the following pattern: ア, イ, ウ, …, ヲ, ン, アア, イイ, ウウ, … *end example*]
arabicAbjad (Arabic Abjad Numerals)	Specifies that the sequence shall consist of one or more occurrences of a single ascending Abjad numeral from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–28 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–28 is U+0623, U+0628, U+062C, U+062F, U+0647, U+0648, U+0632, U+062D, U+0637, U+064A, U+0643, U+0644, U+0645, U+0646, U+0633, U+0639, U+0641, U+0635, U+0642, U+0631, U+0634, U+062A, U+062B, U+062E,U+0630, U+0636,U+063A, and U+0638, respectively. For values greater than 28, the text displayed shall be constructed as follows: 1. Repeatedly subtract the size of the set (28) from the value until the result is equal to or less than the size of the set. 2. The remainder determines which character to use from the set above, and that character is written once and then written once and then

Enumeration Value	Description
	written once and then repeated the number of times the size of the set was subtracted from the original value. [*Example*: The numbering for the items should be represented by the following pattern: أ, ب, ج, ..., ظ, غ, أأ, ب ب, ج ج, ..., ظ ظ, غ غ, أ أ أ, ب ب ب, ج ج ج, ... *end example*]
arabicAlpha (Arabic Alphabet)	Specifies that the sequence shall consist of one or more occurrences of a single character in the Arabic alphabet from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–28 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–28 is U+0623, U+0628, U+062A, U+062B, U+062C, U+062D, U+062E, U+062F, U+0630, U+0631, U+0632, U+0633, U+0634, U+0635, U+0636, U+0637, U+0638, U+0639, U+063A, U+0641, U+0642, U+0643, U+0644, U+0645, U+0646, U+0647, U+0648, and U+064A, respectively. For values greater than 28, the text displayed shall be constructed as follows: 1. Repeatedly subtract the size of the set (28) from the value until the result is equal to or less than the size of the set. 2. The remainder determines which character to use from the set above, and that character is written once and then written once and then written once and then repeated the number of times the size of the set was subtracted from the original value. [*Example*: The numbering for the items should be represented by the following pattern: أ, ب, ت, ..., و, ي, أ أ, ب ب, ت ت, ..., و و, ي ي, أأأ, ب ب ب, ت ت ت, ... *end example*]
bahtText (Thai Baht Text)	Specifies that the sequence shall consist of a numeric value from the Thai counting system, with บาทถ้วน

Enumeration Value	Description
	appended to the result. The portion of the sequence which is prepended to the static text shall be the equivalent value in the thaiCounting format, defined below. [*Example*: The values for the items should be represented by the following pattern: หนึ่งบาทถ้วน, สองบาทถ้วน, สามบาทถ้วน. *end example*]
bullet (Bullet)	Specifies that the sequence shall consist of the bullet character defined by the lvlText element (§17.9.11). [*Example*: ● *end example*]
cardinalText (Cardinal Text)	Specifies that the sequence shall consist of cardinal text of the run language. This sequence is a set of strings each of which is the textual representation, in the language of the lang element (§17.3.2.20), of a different unique position in that sequence. [*Example*: The numbering for the items in Spanish should be represented by the following pattern: Uno, Dos, Tres, …, Nueve, Diez, Once, … Diecineuve, Veinte, Veintiuno, … *end example*]
chicago (Chicago Manual of Style)	Specifies that the sequence shall consist of one or more occurrences of a single character from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–4 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–4 is U+002A, U+2020, U+2021, and U+00A7, respectively. For values greater than 4, the text displayed shall be constructed as follows: 1. Repeatedly subtract the size of the set (4) from the value until the result is equal to or less than the size of the set. 2. The remainder determines which character to

Enumeration Value	Description
	use from the set above, and that character is written once and then written once and then repeated the number of times the size of the set was subtracted from the original value. [*Example*: The first nine items in this format are: *, †, ‡, §, **, ††, ‡‡, §§, ***. *end example*]
chineseCounting (Chinese Counting System)	Specifies that the sequence shall consist of one or more occurrences of a single ascending number from the Chinese counting system, from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–10 and then repeats those same characters using the logic defined below to construct all other values. ○ represents the number zero. The set of characters used by this numbering format for values 0–10 is U+25CB, U+4E00, U+4E8C, U+4E09, U+56DB, U+4E94, U+516D, U+4E03, U+516B, U+4E5D, and U+5341, respectively. For values greater than 10, the text displayed shall be constructed as follows: 1. Divide the value by 10 and write the symbol that represents the remainder. If the quotient is less than 10, then write 十 to the left of the symbol, which represents the remainder. 2. Divide the quotient of the previous division by 10, and write the symbol, which represents the remainder, to the left of the existing characters. 3. Repeat step 2 until the remaining value is equal to zero. [*Example*: The numbering for the items should be represented by the following pattern: 一, 二, 三, …, 九, 十, 十一, 十二, …, 十九, 二十, 二十一, …, 九十九, 一○○, 一○一, … *end example*]
chineseCountingThousand (Chinese Counting Thousand System)	Specifies that the sequence shall consist of one or more occurrences of a single sequential number from the Chinese counting thousand system, from the set listed below.

©ISO/IEC 2012 – All rights reserved

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–10, 100, 1,000, and 10,000, and then repeats those same characters using the logic defined below to construct all other values.

The set of characters used by this numbering format is U+96F6, U+4E00, U+4E8C, U+4E09, U+56DB, U+4E94, U+516D, U+4E03, U+516B, U+4E5D, U+5341, U+767E, U+5343, and U+4E07.

To construct a value that is beyond the set, but less than one hundred thousand, work from largest groups to smallest following these steps:

1. Create as many groups as possible that contain ten thousand in each group.
• Write down the symbol representing that value (1–9):
<table>
<tr><th>Digit</th><th>Character</th></tr>
<tr><td>1</td><td>一 (U+4E00)</td></tr>
<tr><td>2</td><td>二 (U+4E8C)</td></tr>
<tr><td>3</td><td>三 (U+4E09)</td></tr>
<tr><td>4</td><td>四 (U+56DB)</td></tr>
<tr><td>5</td><td>五 (U+4E94)</td></tr>
<tr><td>6</td><td>六 (U+516D)</td></tr>
<tr><td>7</td><td>七 (U+4E03)</td></tr>
<tr><td>8</td><td>八 (U+516B)</td></tr>
<tr><td>9</td><td>九 (U+4E5D)</td></tr>
</table>
• If no groups are formed, do not write any characters.
• If groups were formed, write down the symbol representing ten thousand: 万

2. Repeat step 1 for groups of one thousand (千) using the corresponding symbol to indicate the groups (so five thousand would be 五千).
• If the original value was between 10,000 and 100,000 and If no groups are formed (and the number is not a multiple of</td></tr>
</table>

Enumeration Value	Description
	thousand) write the symbol 零 instead (so ten thousand and five would be 一万零五) 3. Repeat step 1 for groups of one hundred (百) using the corresponding symbol to indicate the groups (so five hundred would be 五百). - If the original value was between 1,000 and 10,000 and if no groups are formed (and the number is not a multiple of hundred) write the symbol 零 instead (so one thousand and five would be 一千零五) 4. Repeat step 1 for groups of ten (十) using the corresponding symbol to indicate the groups (so fifty would be 五十). - If the original value was between 100 and 1000 and if no groups are formed (and the number is not a multiple of ten) write the symbol 零 instead (so one hundred and five would be 一百零五) 5. Write down the symbol for the remaining number. If the number is larger than one hundred thousand but less than one hundred million, perform the cycle for the numbers above one thousand, but use two characters to represent each group. So, for example, groups of one hundred thousand are represented as ten ten thousands (一十万). An additional symbol for counting groups is introduced at one hundred million (because 10 million is one thousand ten thousands). [*Example*: The numbering for the items should be represented by the following pattern: 一, 二, 二, 三, …, 八, 九, 一十, 一十一, 一十二, …, 一十九, 二十, 二十一, …, 九九, 一百, 一百一, … *end example*]
chineseLegalSimplified (Chinese Legal Simplified Format)	Specifies that the sequence shall consist of one or more occurrences of a single sequential number from the Chinese simplified legal format, from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those are combined with additional characters to represent the corresponding power of ten.

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	The set of characters used by this numbering format for values 1–10 is U+96F6, U+58F9, U+8D30, U+53C1, U+8086, U+4F0D, U+9646, U+67D2, U+634C, and U+7396, respectively.

To construct a number that is less than one hundred thousand, work from largest groups to smallest following these steps:

1. Create as many groups as possible that contain ten thousand in each group.
   - Write down the symbol representing that value (1–9):

Digit	Character
1	壹(U+58F9)
2	贰 (U+8D30)
3	叁 (U+53C1)
4	肆(U+8086)
5	伍 (U+4F0D)
6	陆 (U+9646)
7	柒 (U+67D2)
8	捌 (U+634C)
9	玖 (U+7396)

   - If no groups are formed, do not write any characters.
   - If groups were formed, write down the symbol representing ten thousand: 万
2. Repeat step 1 for groups of one thousand (仟) using the corresponding symbol to indicate the groups (so five thousand would be 伍仟).
   - If the original value was between 10,000 and 100,000 and If no groups are formed (and the number is not a multiple of thousand) write the symbol 零 instead (so ten thousand and five would be 壹万零伍)
3. Repeat step 1 for groups of one hundred (佰) using the corresponding symbol to indicate the groups (so five hundred would be 伍佰).
   - If the original value was between 1,000 and 10,000 and if no groups are formed

Enumeration Value	Description
	(and the number is not a multiple of hundred) write the symbol 零 instead (so one thousand and five would be 壹仟零伍) 4. Repeat step 1 for groups of ten (拾) using the corresponding symbol to indicate the groups (so fifty would be 伍拾). - If the original value was between 100 and 1000 and If no groups are formed (and the number is not a multiple of ten) write the symbol 零 instead (so one hundred and five would be 壹佰零伍) 5. Write down the symbol for the remaining number. If the number is larger than one hundred thousand but less than one hundred million, perform the cycle for the numbers above one thousand but use two characters to represent each group. So, for example, groups of one million are represented as one hundred ten thousands (佰万). An additional symbol for counting groups is introduced at one hundred million (because 10 million is one thousand ten thousands). [*Example*: The numbering for the items should be represented by the following pattern: 壹, 贰叁, ..., 捌, 玖, 壹拾, 壹拾壹, 壹拾贰, …, 壹拾玖, 贰拾, 贰拾壹, …, 玖玖, 壹佰, 壹佰零壹, … *end example*]
chosung (Korean Chosung Numbering)	Specifies that the sequence shall consist of one or more occurrences of a single sequential number from the Korean Chosung format, from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–14 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–14 is U+3131, U+3134, U+3137, U+3139, U+3141, U+3142, U+3145, U+3147, U+3148, U+314A, U+314B, U+314C, U+314D, and U+314E, respectively. For values greater than 14, the text displayed shall be constructed as follows:

Enumeration Value	Description
	1. Repeatedly subtract the size of the set (14) from the value until the result is equal to or less than the size of the set. 2. The remainder determines which character to use from the set above, and that character is written once and then repeated the number of times the size of the set was subtracted from the original value. [*Example*: The numbering for the items should be represented by the following pattern: ㄱ ,ㄴ ,ㄷ, …, ㅍ, ㅎ, ㄱㄱ, ㄴㄴ, ㄷㄷ, … *end example*]
custom (Custom Defined Number Format)	Specifies a custom number format using the parent element's attribute to specify a number format using the mechanism defined by the XSLT format attribute.
decimal (Decimal Numbers)	Specifies that the sequence shall consist of decimal numbering. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and 0 (represents the number zero) to construct the remaining values. The set of characters used by this numbering format for values 0–9 is U+0030–U+0039, respectively. Continue the sequence by using the following steps: 1. Increment the rightmost position. 2. Every time the end of the set is reached, for a given position, increment the position to the immediate left (if there is no position to the immediate left, create a new position and start the sequence of the new position at 1) and reset the current position to 0. [*Example*: The numbering for the items should be represented by the following pattern: 1, 2, 3, …, 8, 9, 10, 11, 12, …, 18, 19, 20, 21, … *end example*]
decimalEnclosedCircle (Decimal Numbers Enclosed in a Circle)	Specifies that the sequence shall consist of decimal numbering enclosed in a circle, using the enclosed character.

Enumeration Value	Description
	This system uses a set of characters to represent the numbers 1–20. The set of characters used by this numbering format for values 1–20 is U+2460–U+2473, respectively. For values greater than the size of the set, the items fall back to the decimal format. [*Example*: The numbering for the items should be represented by the following pattern: ①, ②, ③, …, ⑲ , ⑳, 21, … *end example*]
decimalEnclosedCircleChinese (Decimal Numbers Enclosed in a Circle)	Identical to decimalEnclosedCircle.
decimalEnclosedFullstop (Decimal Numbers Followed by a Period)	Specifies that the sequence shall consist of decimal numbering followed by a period, using the appropriate character, as described below. This system uses a set of characters to represent the numbers 1–20. The set of characters used by this numbering format for values 1–20 is U+2488–U+249B, respectively. For values greater than the size of the set, the items fall back to the decimal format. [*Example*: The numbering for the items should be represented by the following pattern: ⒈, ⒉, ⒊, …, ⒚, ⒛, 21, … *end example*]
decimalEnclosedParen (Decimal Numbers Enclosed in Parenthesis)	Specifies that the sequence shall consist of decimal numbering enclosed in parentheses, using the appropriate character, as described below. This system uses a set of characters to represent the numbers 1–20. The set of characters used by this numbering format for values 1–20 is U+2474–U+2487, respectively. For values greater than the size of the set, the items fall back to the decimal format. [*Example*: The numbering for the items should be represented by the following pattern: ⑴, ⑵, ⑶, …, ⒆, ⒇, 21, 22, … *end example*]

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
decimalFullWidth (Full Width Arabic Numerals)	Specifies that the sequence shall consist of full-width Arabic numbering. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and ０ (represents the number zero) to construct the remaining values. The set of characters used by this numbering format for values 0–9 is U+FF10–U+FF19, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Divide the value by 10 and write the symbol which represents the remainder. 2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing position. 3. Repeat step 2 until the remaining value is equal to zero. [*Example*: The numbering for the items should be represented by the following pattern: １，２，３，…，８，９，１０，１１，１２，…，１８，１９，２０，２１，… *end example*]
decimalHalfWidth (Half Width Arabic Numerals)	Specifies that the sequence shall consist of half-width Arabic numbering. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and 0 (represents the number zero) to construct the remaining values. The set of characters used by this numbering format for values 0–9 is U+0030–U+0039, respectively. For values greater than the size of the set, the number is constructed by following these steps:

Enumeration Value	Description
	1. Divide the value by 10 and write the symbol which represents the remainder. 2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing position. 3. Repeat step 2 until the remaining value is equal to zero. [*Example*: The numbering for the items should be represented by the following pattern: 1, 2, 3, …, 8, 9, 10, 11, 12, …, 18, 19, 20, 21, … *end example*]
decimalZero (Initial Zero Arabic Numerals)	Specifies that the sequence shall consist of Arabic numbering with a zero added to numbers one through nine. To determine the text that is displayed for any value, this sequence specifies a set of paired characters (zero followed by an additional symbol) that represent positions 1–9 and then those same characters are combined with each other to construct the remaining values. The set of characters used by this numbering format for values 0–9 is U+0030–U+0039. For values greater than the size of the set, the number is constructed by following these steps: 1. Divide the value by 10 and write the symbol which represents the remainder. 2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing position. 3. Repeat step 2 until the remaining value is equal to zero. [*Example*: The numbering for the items should be represented by the following pattern: 01, 02, 03, …, 08, 09, 10, 11, 12, …, 18, 19, 20, 21, 22, …, 98, 99, 100, 101, … *end example*]
dollarText (Dollar Text)	Specifies that the sequence shall consist of a cardinal text value of the run language, with "and `00/100`" (also in the run language) appended to the result.

Enumeration Value	Description
	[*Note*: The latter text is fixed because values in numbering sequences are integer-based. *end note*] The cardinal text value which is prepended to the static text shall be the equivalent value in the cardinalText format, defined below. [*Example*: one and 00/100, two and 00/100, three and 00/100. *end example*]
ganada (Korean Ganada Numbering)	Specifies that the sequence shall consist of one or more occurrences of a single sequential number from the Korean Ganada format, from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–14 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–14 is U+AC00, U+B098, U+B2E4, U+B77C, U+B9C8, U+BC14, U+C0AC, U+C544, U+C790, U+CC28, U+CE74, U+D0C0, U+D30C, and U+D558, respectively. For values greater than 14, the text displayed shall be constructed as follows: 1. Repeatedly subtract the size of the set (14) from the value until the result is equal to or less than the size of the set. 2. The remainder determines which character to use from the set above, and that sequence of character is repeated the number of times the size of the set was subtracted from the original value. [*Example*: The numbering for the items should be represented by the following pattern: 가, 나, 다, …, 파, 하, 가가, 나나, 다다, … *end example*]
hebrew1 (Hebrew Letters)	Specifies that the sequence shall consist of Hebrew letters from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9, each multiple of ten (less than

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>
<p>100), each multiple of 100 (less than 1000), etc. which are then combined with each other to construct the remaining values.</p>
<p>The set of characters used by this numbering format is U+05D0–U+05D9, U+05DB, U+05DC, U+05DE, U+05E0–U+05E2, U+05E4, U+05E6–U+05EA, U+05E7–U+05E9, U+05EA, U+05DA, U+05DD, U+05DF, U+05E3, and U+05E5, respectively.</p>
<p>To construct the text for any value, convert it from its decimal equivalent following these steps (writing right to left):</p>
<ol>
<li>Replace the thousands digit with the appropriate symbol:
<table>
<tr><th>Digit</th><th>Symbol</th></tr>
<tr><td>1</td><td>א (U+05D0)</td></tr>
<tr><td>2</td><td>ב (U+05D1)</td></tr>
<tr><td>3</td><td>ג (U+05D2)</td></tr>
<tr><td>4</td><td>ד (U+05D3)</td></tr>
<tr><td>5</td><td>ה (U+05D4)</td></tr>
<tr><td>6</td><td>ו (U+05D5)</td></tr>
<tr><td>7</td><td>ז (U+05D6)</td></tr>
<tr><td>8</td><td>ח (U+05D7)</td></tr>
<tr><td>9</td><td>ט (U+05D8)</td></tr>
</table>
</li>
<li>Replace the hundreds digit with the appropriate symbol:
<table>
<tr><th>Digit</th><th>Symbol</th></tr>
<tr><td>1</td><td>ק (U+05E7)</td></tr>
<tr><td>2</td><td>ר (U+05E8)</td></tr>
<tr><td>3</td><td>ש (U+ 05E9)</td></tr>
<tr><td>4</td><td>ת (U+05EA)</td></tr>
<tr><td>5</td><td>ך (U+05DA)</td></tr>
<tr><td>6</td><td>ם (U+05DD)</td></tr>
<tr><td>7</td><td>ן (U+05DF)</td></tr>
<tr><td>8</td><td>ף (U+05E3)</td></tr>
</table>
</li>
</ol>
</td></tr>
</table>

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>9</td><td>ץ (U+05E5)</td></tr>
</table>

3. If the remainder is 15 or 16, replace them as follows and stop:

<table>
<tr><th>Value</th><th>Symbols</th></tr>
<tr><td>15</td><td>טו (U+05D8, U+05D5)</td></tr>
<tr><td>16</td><td>טז (U+05D8, U+05D6)</td></tr>
</table>

4. Otherwise, replace the tens digit with the appropriate symbol:

<table>
<tr><th>Digit</th><th>Symbol</th></tr>
<tr><td>1</td><td>י (U+05d9)</td></tr>
<tr><td>2</td><td>כ (U+05DB)</td></tr>
<tr><td>3</td><td>ל (U+ 05DC)</td></tr>
<tr><td>4</td><td>מ (U+05DE)</td></tr>
<tr><td>5</td><td>נ (U+05E0)</td></tr>
<tr><td>6</td><td>ס (U+05E1)</td></tr>
<tr><td>7</td><td>ע (U+05E2)</td></tr>
<tr><td>8</td><td>פ (U+05E4)</td></tr>
<tr><td>9</td><td>צ (U+05E6)</td></tr>
</table>

5. Replace the ones digit with the appropriate symbol:

<table>
<tr><th>Digit</th><th>Symbol</th></tr>
<tr><td>1</td><td>א (U+05d0)</td></tr>
<tr><td>2</td><td>ב (U+05d1)</td></tr>
<tr><td>3</td><td>ג (U+ 05d2)</td></tr>
<tr><td>4</td><td>ד (U+05d3)</td></tr>
<tr><td>5</td><td>ה (U+05d4)</td></tr>
<tr><td>6</td><td>ו (U+05D5)</td></tr>
<tr><td>7</td><td>ז (U+05D6)</td></tr>
<tr><td>8</td><td>ח (U+05d7)</td></tr>
<tr><td>9</td><td>ט (U+05D8)</td></tr>
</table>

6. Reordering a number group will not change its
</td></tr>
</table>

Enumeration Value	Description
	value. If a number spells out a Hebrew word with a negative or positive connotation, the number group can be rearranged. [*Example*: The numbering for the items should be represented by the following pattern: א, ב, ג, … טז, זי, חי, טי, כ, אכ, … *end example*]
hebrew2 (Hebrew Alphabet)	Specifies that the sequence shall consist of the Hebrew alphabet. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–22 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–22 is U+05D0–U+05D9, U+05DB, U+05DC, U+05DE, U+05E0–U+05E2, U+05E4, and U+05E6–U+05EA, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Repeatedly subtract the size of the set (22) from the value until the result is equal to or less than the size of the set. 2. Write the symbol represented by the result value. 3. Then the ת symbol is repeated (to the right of the first symbol) for each time the size of the set was subtracted from the original value. 4. Reordering a number group does not change its value. If a number spells out a Hebrew word with a negative or positive connotation, the number group can be rearranged. [*Example*: The numbering for the items should be represented by the following pattern: א, ב, ג, …, את, בת, … *end example*]
hex (Hexadecimal Numbering)	Specifies that the sequence shall consist of hexadecimal numbering. To determine the text that is displayed for any value, this sequence specifies a set of characters that

Enumeration Value	Description
	represent positions 1–15 and then those same characters are combined with each other and 0 (represents the number zero) to construct the remaining values. The set of characters used by this numbering format for values 0–15 is U+0030–U+0039, and U+0041–U+0046, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Divide the value by 16 and write the symbol which represents the remainder. 2. Divide the quotient of the previous division by 16 and write the symbol, which represents the remainder, to the left of the existing position. 3. Repeat step 2 until the remaining value is equal to zero. [*Example*: The numbering for the items should be represented by the following pattern: 1, 2, 3, …, E, F, 10,11, 12, …, 1E, 1F, 20, 21, … *end example*]
hindiConsonants (Hindi Consonants)	Specifies that the sequence shall consist of one or more occurrences of a single Hindi consonant from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–18 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–18 is U+0905–U+0914, U+0905 combined with U+0902, and U+0905 combined with U+0903, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Repeatedly subtract the size of the set (18) from the value until the result is equal to or less than the size of the set. 2. The result value determines which character to use, and the same character is written once

Enumeration Value	Description
	and then repeated for each time the size of the set was subtracted from the original value. [*Example*: The numbering for the items should be represented by the following pattern: अ, आ, इ, …, अं,अः, अअ, आआ, इइ, …, अंअं, अःअः, अअअ, आआआ, इइइ, … *end example*]
hindiCounting (Hindi Counting System)	Specifies that the sequence shall consist of sequential numbers from the Hindi counting system. This sequence is a set of strings each of which is the full name, in Hindi, of the next value in that sequence. [*Example*: The numbering for the items should be represented by the following pattern: एक, दो, तीन, चार, पाँच, छः, सात, आठ, नौ, दस, … *end example*]
hindiNumbers (Hindi Numbers)	Specifies that the sequence shall consist of one or more occurrences of a single Hindi number from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and ० (U+0966, which represents the number zero) to construct the remaining values. The set of characters used by this numbering format for values 1–9 is U+0967, U+0968, U+0969, U+096A, U+096B, U+096C, U+096D, U+096E, and U+096F, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Divide the value by 10 and write the symbol which represents the remainder. 2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing position. 3. Repeat step 2 until the remaining value is equal to zero.

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	[*Example*: The numbering for the items should be represented by the following pattern: १, २, ३, …, ८, ९, १०, ११, १२, …, १८, १९, २०, २१, … *end example*]
hindiVowels (Hindi Vowels)	Specifies that the sequence shall consist of one or more occurrences of a single Hindi vowel from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–37 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–37 is U+0915–U+0939, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Repeatedly subtract the size of the set (37) from the value until the result is equal to or less than the size of the set. 2. The result value determines which character to use, and the same character is written once and then repeated for each time the size of the set was subtracted from the original value. [*Example*: The numbering for the items should be represented by the following pattern: क, ख, ग, …, स, ह, कक, खख, गग, …, सस, हह, ककक, खखख, गगग, … *end example*]
ideographDigital (Ideographs)	Specifies that the sequence shall consist of sequential numerical ideographs, using the appropriate character, as described below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and 〇 (represents the number zero) to construct the remaining values. The set of characters used by this numbering format

Enumeration Value	Description
	for values 0–9 is U+3007, U+4E00, U+4E8C, U+4E09, U+56DB, U+4E94, U+516D, U+4E03, U+516B, and U+4E5D, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Divide the value by 10 and write the symbol which represents the remainder. 2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing position. 3. Repeat step 2 until the remaining value is equal to zero. [*Example*: The numbering for the items should be represented by the following pattern: 一, 二, 三, …, 八, 九, 一〇, 一一, 一二, …, 一八, 一九, 二〇, 二一, … *end example*]
ideographEnclosedCircle (Ideographs Enclosed in a Circle)	Specifies that the sequence shall consist of sequential numerical ideographs, using the appropriate character, as described below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–10. The set of characters used by this numbering format for values 1–10 is U+3220–U+3229, respectively. For values greater than the size of the set, the items fall back to the decimal format. [*Example*: The numbering for the items should be represented by the following pattern: ㈠, ㈡, ㈢, …, ㈨, ㈩, 11,12, … *end example*]
ideographLegalTraditional (Traditional Legal Ideograph Format)	Specifies that the sequence shall consist of sequential numerical traditional legal ideographs. This system uses a set of characters to represent the numbers 1–9 and then those are combined with additional characters to represent the corresponding power of ten.

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>
<p>The set of characters used by this numbering format is U+58F9, U+8CB3, U+53C3, U+8086, U+4F0D, U+9678, U+67D2, U+634C, U+7396, U+62FE, U+4F70, U+4EDF,and U+842C, respectively.</p>
<p>To construct a number that is less than hundred thousand, you work from largest groups to smallest following these steps:</p>
<ol>
<li>Create as many groups as possible that contain ten thousand in each group.
<ul>
<li>Write down the symbol representing that value (1–9):
<table>
<tr><th>Digit</th><th>Character</th></tr>
<tr><td>1</td><td>壹 (U+58F9)</td></tr>
<tr><td>2</td><td>貳 (U+8CB3)</td></tr>
<tr><td>3</td><td>參 (U+53C3)</td></tr>
<tr><td>4</td><td>肆 (U+8086)</td></tr>
<tr><td>5</td><td>伍 (U+4F0D)</td></tr>
<tr><td>6</td><td>陸 (U+9678)</td></tr>
<tr><td>7</td><td>柒 (U+67D2)</td></tr>
<tr><td>8</td><td>捌 (U+634C)</td></tr>
<tr><td>9</td><td>玖 (U+7396)</td></tr>
</table>
</li>
<li>If no groups are formed, do not write any characters.</li>
<li>If groups were formed, write down the symbol representing ten thousand (the power of ten represented by that position): 萬</li>
</ul>
</li>
<li>Repeat this for groups of one thousand (仟), one hundred (佰) and ten (拾) using the corresponding symbol to indicate the groups (so five hundred would be 伍佰 and fifty would be 伍拾).</li>
<li>Write down the symbol for the remaining number.</li>
</ol>
<p>If the number is larger than hundred thousand but less than one hundred million, you perform the cycle for the numbers above one thousand but use two characters to represent each group. So, for example,</p>
</td></tr>
</table>

Enumeration Value	Description
	groups of one million are represented as one hundred ten thousands (拾萬). An additional symbol for counting groups is introduced at one hundred million (because 10 million is one thousand ten thousands). [*Example*: The numbering for the items should be represented by the following pattern: 壹, 貳, 參, …, 捌, 玖, 壹拾, 壹拾壹, 壹拾貳, …, 壹拾玖, 貳拾, 貳拾壹, … *end example*]
ideographTraditional (Traditional Ideograph Format)	Specifies that the sequence shall consist of sequential numerical traditional ideographs. This system uses a set of characters (elements in the Chinese Sexagenary cycle) to represent the numbers 1–10. For values greater than the size of the set, the items fall back to the decimal format. The set of characters used by this numbering format for values 1–10 is U+7532, U+4E59, U+4E19, U+4E01, U+620A, U+5DF1, U+5E9A, U+8F9B, U+58EC, and U+7678, respectively. [*Example*: The numbering for the items should be represented by the following pattern: 甲, 乙, 丙, 丁, …, 壬, 癸, 11, 12, … *end example*]
ideographZodiac (Zodiac Ideograph Format)	Specifies that the sequence shall consist of sequential zodiac ideographs. This system uses a set of characters (animals in the Chinese Sexagenary cycle) to represent the numbers 1–12. For values greater than the size of the set, the items fall back to the decimal format. The set of characters used by this numbering format for values 1–12 is U+5B50, U+4E11, U+5BC5, U+536F, U+8FB0, U+5DF3, U+5348, U+672A, U+7533, U+9149, U+620C, and U+4EA5, respectively. [*Example*: The numbering for the items should be represented by the following pattern: 子, 丑, 寅, …, 戌, 亥, 13, 14, … *end example*]
ideographZodiacTraditional (Traditional Zodiac Ideograph Format)	Specifies that the sequence shall consist of sequential traditional zodiac ideographs.

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	This system uses a set of character pairs (all of the element-animal combinations of the Chinese sexagenary cycle) to represent the numbers 1–60 and then those same character pairs are repeated to construct the remaining values. The set of character pairs used by this numbering format for values 1–60 is U+7532, U+5B50; U+4E59, U+4E11; U+4E19, U+5BC5; U+4E01, U+536F; U+620A, U+8FB0; U+5DF1, U+5DF3; U+5E9A, U+5348; U+8F9B, U+672A; U+58EC, U+7533; U+7678, U+9149; U+7532, U+620D; U+4E59, U+4EA5; U+4E19, U+5B50; U+4E01, U+4E11; U+620A, U+5BC5; U+5DF1, U+536F; U+5E9A, U+8FB0; U+8F9B, U+5DF3; U+58EC, U+5348; U+7678, U+672A; U+7532, U+7533; U+4E59, U+9149; U+4E19, U+620D; U+4E01, U+4EA5; U+620A, U+5B50; U+5DF1, U+4E11; U+5E9A, U+5BC5; U+8F9B, U+536F; U+58EC, U+8FB0; U+7678, U+5DF3; U+7532, U+5348; U+4E59, U+672A; U+4E19, U+7533; U+4E01, U+9149; U+620A, U+620D; U+5DF1, U+4EA5; U+5E9A, U+5B50; U+8F9B, U+4E11; U+58EC, U+5BC5; U+7678, U+536F; U+7532, U+8FB0; U+4E59, U+5DF3; U+4E19, U+5348; U+4E01, U+672A; U+620A, U+7533; U+5DF1, U+9149; U+5E9A, U+620D; U+8F9B, U+4EA5; U+58EC, U+5B50; U+7678, U+4E11; U+7532, U+5BC5; U+4E59, U+536F; U+4E19, U+8FB0; U+4E01, U+5DF3; U+620A, U+5348; U+5DF1, U+672A; U+5E9A, U+7533; U+8F9B, U+9149; U+58EC, U+620D; U+7678, U+4EA5, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Repeatedly subtract the size of the set (60) from the value until the result is equal to or less than the size of the set. 2. The result value determines which character pair to use. [*Example*: The numbering for the items should be represented by the following pattern: 甲子, 乙丑, 丙寅, …, 壬戌, 癸亥, 甲子, 乙丑, 丙寅, … *end example*]
iroha (Iroha Ordered Katakana)	Specifies that the sequence shall consist of the iroha. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–48 and then repeats those same

Enumeration Value	Description
	characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–48 is U+FF72, U+FF9B, U+FF8A, U+FF86, U+FF8E, U+FF8D, U+FF84, U+FF81, U+FF98, U+FF87, U+FF99,U+FF66, U+FF9C, U+FF76, U+FF96, U+FF80, U+FF9A, U+FF7F, U+FF82, U+FF88, U+FF85, U+FF97, U+FF91, U+FF73, U+30F0, U+FF89, U+FF75, U+FF78, U+FF94, U+FF8F, U+FF79, U+FF8C, U+FF7A, U+FF74, U+FF83, U+FF71, U+FF7B, U+FF77, U+FF95, U+FF92, U+FF90, U+FF7C, U+30F1, U+FF8B, U+FF93, U+FF7E, U+FF7D, and U+FF9D, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Repeatedly subtract the size of the set (48) from the value until the result is equal to or less than the size of the set. 2. The result value determines which character to use. [*Example*: The numbering for the items should be represented by the following pattern: ｲ, ﾛ, ﾊ, …, ｽ, ﾝ, ｲ, ﾛ, ﾊ, … *end example*]
irohaFullWidth (Full-Width Iroha Ordered Katakana)	Specifies that the sequence shall consist of the full-width forms of the iroha. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–48 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–48 is U+30A4, U+30ED, U+30CF, U+30CB, U+30DB, U+30D8, U+30C8, U+30C1, U+30EA, U+30CC, U+30EB, U+30F2, U+30EF, U+30AB, U+30E8, U+30BF, U+30EC, U+30BD, U+30C4, U+30CD, U+30CA, U+30E9, U+30E0, U+30A6, U+30F0, U+30CE, U+30AA, U+30AF, U+30E4, U+30DE, U+30B1, U+30D5, U+30B3, U+30A8, U+30C6, U+30A2, U+30B5, U+30AD, U+30E6, U+30E1, U+30DF, U+30B7, U+30F1, U+30D2, U+30E2, U+30BB, U+30B9, and U+30F3, respectively.

©ISO/IEC 2012 – All rights reserved

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>
<p>For values greater than the size of the set, the number is constructed by following these steps:</p>
<ol>
<li>Repeatedly subtract the size of the set (48) from the value until the result is equal to or less than the size of the set.</li>
<li>The result value determines which character to use.</li>
</ol>
<p>[<i>Example</i>: The numbering for the items should be represented by the following pattern: イ, ロ, ハ, …, ス , ン, イ, ロ, ハ, … <i>end example</i>]</p>
</td></tr>
<tr><td>japaneseCounting (Japanese Counting System)</td><td>
<p>Specifies that the sequence shall consist of sequential numbers from the Japanese counting system.</p>
<p>This system uses a set of characters to represent the numbers 1–9 and then those are combined with additional characters to represent the corresponding power of ten.</p>
<p>The set of characters used by this numbering format is U+3007, U+4E00, U+4E8C, U+4E09, U+56DB, U+4E94, U+516D, U+4E03, U+516B, U+4E5D, U+5341, U+5343, and U+767E, respectively.</p>
<p>To construct a number that is less than ten thousand, you work from largest groups to smallest following these steps:</p>
<ol>
<li>Create as many groups as possible that contain one thousand in each group.
<ul>
<li>Write down the symbol representing that value (1–9):
<table>
<tr><th>Digit</th><th>Character</th></tr>
<tr><td>1</td><td>一 (U+4E00)</td></tr>
<tr><td>2</td><td>二 (U+4E8C)</td></tr>
<tr><td>3</td><td>三 (U+4E09)</td></tr>
<tr><td>4</td><td>四 (U+56DB)</td></tr>
<tr><td>5</td><td>五 (U+4E94)</td></tr>
<tr><td>6</td><td>六 (U+516D)</td></tr>
<tr><td>7</td><td>七 (U+4E03)</td></tr>
<tr><td>8</td><td>八 (U+516B)</td></tr>
</table>
</li>
</ul>
</li>
</ol>
</td></tr>
</table>

Enumeration Value	Description
	9 | 九 (U+4E5D) • If no groups are formed, do not write any characters. • If groups were formed, write down the symbol representing one thousand (the power of ten represented by that position): 千 2. Repeat this for groups of one hundred (百) and ten (十) using the corresponding symbol to indicate the groups (so five hundred would be 五百 and fifty would be 五十). 3. Write down the symbol for the remaining number. If the number is larger than ten thousand but less than one hundred million, you perform the cycle for the numbers above one thousand but use two characters to represent each group. So, for example, groups of one million are represented as one hundred ten thousands (百万). An additional symbol for counting groups is introduced at one hundred million (because 10 million is one thousand ten thousands). [*Example*: The numbering for the items should be represented by the following pattern: 一, 二, 三, …, 八, 九, 十, 十一, 十二, …, 十八, 十九, 二十, 二十一, … *end example*]
japaneseDigitalTenThousand (Japanese Digital Ten Thousand Counting System)	Specifies that the sequence shall consist of sequential numbers from the Japanese digital ten thousand counting system. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and 〇 (represents the number zero) to construct the remaining values. The set of characters used by this numbering format for values 0–9 is U+3007, U+4E00, U+4E8C, U+4E09, U+56DB, U+4E94, U+516D, U+4E03, U+516B, and U+4E5D, respectively. For values greater than the size of the set, the number is constructed by following these steps:

©ISO/IEC 2012 – All rights reserved

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>1. Divide the value by 10 and write the symbol which represents the remainder.
2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing position.
3. Repeat step 2 until the remaining value is equal to zero.

[<i>Example</i>: The numbering for the items should be represented by the following pattern: 一, 二, 三, …, 八 , 九, 一〇, 一一, 一二, …, 一八, 一九, 二〇, 二一, … <i>end example</i>]</td></tr>
<tr><td>japaneseLegal (Japanese Legal Numbering)</td><td>Specifies that the sequence shall consist of sequential numbers from the Japanese legal counting system.

This system uses a set of characters to represent the numbers 1–9 and then those are combined with additional characters to represent the corresponding power of ten.

The set of characters used by this numbering format for values is U+58F1, U+5F10, U+53C2, U+56DB, U+4F0D, U+516D, U+4E03, U+516B, U+4E5D, U+62FE, U+767E, U+842C, and U+9621, respectively.

To construct a number that is less than hundred thousand, you work from largest groups to smallest following these steps:

1. Create as many groups as possible that contain ten thousand in each group.
• Write down the symbol representing that value (1–9):
<table>
<tr><th>Digit</th><th>Character</th></tr>
<tr><td>1</td><td>壱(U+58F1)</td></tr>
<tr><td>2</td><td>弐(U+5F10)</td></tr>
<tr><td>3</td><td>参(U+53C2)</td></tr>
<tr><td>4</td><td>四(U+56DB)</td></tr>
<tr><td>5</td><td>伍(U+4F0D)</td></tr>
<tr><td>6</td><td>六(U+516D)</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td><table><tr><td>7</td><td>七(U+4E03)</td></tr><tr><td>8</td><td>八(U+516B)</td></tr><tr><td>9</td><td>九(U+4E5D)</td></tr></table>
<ul><li>If no groups are formed, do not write any characters.</li><li>If groups were formed, write down the symbol representing ten thousand (the power of ten represented by that position): 萬</li></ul>
2. Repeat this for groups of one thousand (阡), one hundred (百) and ten (拾) using the corresponding symbol to indicate the groups (so five hundred would be 伍百 and fifty would be 伍拾).
3. Write down the symbol for the remaining number.

If the number is larger than one hundred thousand but less than one hundred million, you perform the cycle for the numbers above one thousand but use two characters to represent each group. So, for example, groups of one million are represented as one hundred ten thousands (拾萬). An additional symbol for counting groups is introduced at one hundred million (because 10 million is one thousand ten thousands).

[<i>Example</i>: The numbering for the items should be represented by the following pattern: 壱, 弐, 参, …, 八 , 九, 壱拾, 壱拾壱, 壱拾弐, … , 壱拾八, 壱拾九, 弐拾, 弐拾壱, … <i>end example</i>]</td></tr>
<tr><td>koreanCounting (Korean Counting System)</td><td>Specifies that the sequence shall consist of sequential numbers from the Korean counting system.

This system uses a set of characters to represent the numbers 1–9 and then those are combined with additional characters to represent the corresponding power of ten.

The set of characters used by this numbering format is U+C77C, U+C774, U+C0BC, U+C0AC, U+C624, U+C721, U+CE60, U+D314, U+AD6C, U+C2ED, U+B9CC, U+CC9C, and U+BC31, respectively.

To construct a number that is less than hundred</td></tr>
</table>

©ISO/IEC 2012 – All rights reserved

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>thousand, you work from largest groups to smallest following these steps:
<ol>
<li>Create as many groups as possible that contain ten thousand in each group.
<ul>
<li>Write down the symbol representing that value (1–9):
<table>
<tr><th>Digit</th><th>Character</th></tr>
<tr><td>1</td><td>일 (U+C77C)</td></tr>
<tr><td>2</td><td>이 (U+C774)</td></tr>
<tr><td>3</td><td>삼 (U+C0BC)</td></tr>
<tr><td>4</td><td>사 (U+C0AC)</td></tr>
<tr><td>5</td><td>오 (U+C624)</td></tr>
<tr><td>6</td><td>육 (U+C721)</td></tr>
<tr><td>7</td><td>칠 (U+CE60)</td></tr>
<tr><td>8</td><td>팔 (U+D314)</td></tr>
<tr><td>9</td><td>구 (U+AD6C)</td></tr>
</table>
</li>
<li>If no groups are formed, do not write any characters.</li>
<li>If groups were formed, write down the symbol representing ten thousand (the power of ten represented by that position): 만</li>
</ul>
</li>
<li>Repeat this for groups of one thousand (천), one hundred (백) and ten (십) using the corresponding symbol to indicate the groups (so five hundred would be 오백 and fifty would be 오십).</li>
<li>Write down the symbol for the remaining number.</li>
</ol>
If the number is larger than one hundred thousand but less than one hundred million, you perform the cycle for the numbers above one thousand but use two characters to represent each group. So, for example, groups of one million are represented as one hundred ten thousands 십일만. An additional symbol for counting groups is introduced at one hundred million (because 10 million is one thousand ten thousands).</td></tr>
</table>

Enumeration Value	Description
	[*Example*: The numbering for the items should be represented by the following pattern: 일, 이, 삼, …, 팔, 구, 십, 십일, … *end example*]
koreanDigital (Korean Digital Counting System)	Specifies that the sequence shall consist of sequential numbers from the Korean digital counting system. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and 영(represents the number zero) to construct the remaining values. The set of characters used by this numbering format for values 0–9 is U+C601, U+C77C, U+C774, U+C0BC, U+C0AC, U+C624, U+C721, U+CE60, U+D314, and U+AD6C, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Divide the value by 10 and write the symbol which represents the remainder. 2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing position. 3. Repeat step 2 until the remaining value is equal to zero. [*Example*: The numbering for the items should be represented by the following pattern: 일, 이, 삼, …, 팔, 구, 일영, 일일, … *end example*]
koreanDigital2 (Korean Digital Counting System Alternate)	Specifies that the sequence shall consist of sequential numbers from the Korean digital counting system. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and 零 (represents the number zero) to construct the remaining values. The set of characters used by this numbering format for values 0–9 is U+96F6, U+4E00, U+4E8C, U+4E09,

©ISO/IEC 2012 – All rights reserved

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>U+56DB, U+4E94, U+516D, U+4E03, U+516B, and U+4E5D, respectively.

For values greater than the size of the set, the number is constructed by following these steps:

1. Divide the value by 10 and write the symbol which represents the remainder.
2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing position.
3. Repeat step 2 until the remaining value is equal to zero.

[<i>Example</i>: The numbering for the items should be represented by the following pattern: 一, 二, 三, …, 八, 九, 一零, 一一, … <i>end example</i>]</td></tr>
<tr><td>koreanLegal (Korean Legal Numbering)</td><td>Specifies that the sequence shall consist of sequential numbers from the Korean legal numbering system.

This system uses a set of characters to represent the numbers 1–9 and then those are combined with additional characters which represent the multiples of ten (less than one hundred).

The set of characters used by this numbering format is U+D558, U+B098, U+B458, U+C14B, U+B137, U+B2E4, U+C12F, U+C5EC, U+C12F, U+C77C, U+ACF1, U+C5EC, U+B35F, U+C544, U+D649, U+C5F4, U+C2A4, U+BB3C, U+C11C, U+B978, U+B9C8, U+D754, U+C270, U+C608, U+C21C, U+C77C, U+D754, U+C5EC, U+B4E0, U+C544, and U+D754, respectively.

They are assigned according to the following table:

<table>
<tr><th>Digit</th><th>Characters</th></tr>
<tr><td>1</td><td>하나 (U+D558, U+B098)</td></tr>
<tr><td>2</td><td>둘 (U+B458)</td></tr>
<tr><td>3</td><td>셋 (U+C14B)</td></tr>
<tr><td>4</td><td>넷 (U+B137)</td></tr>
<tr><td>5</td><td>다섯 (U+B2E4, U+C12F)</td></tr>
<tr><td>6</td><td>여섯 (U+C5EC, U+C12F)</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>7</td><td>일곱 (U+C77C, U+ACF1)</td></tr>
<tr><td>8</td><td>여덟 (U+C5EC, U+B35F)</td></tr>
<tr><td>9</td><td>아홉 (U+C544 , U+D649)</td></tr>
<tr><td>10</td><td>열 (U+C5F4)</td></tr>
<tr><td>20</td><td>스물 (U+C2A4, U+BB3C)</td></tr>
<tr><td>30</td><td>서른 (U+C11C, U+B978)</td></tr>
<tr><td>40</td><td>마흔 (U+B9C8, U+D754)</td></tr>
<tr><td>50</td><td>쉰 (U+C270)</td></tr>
<tr><td>60</td><td>예순 (U+C608, U+C21C)</td></tr>
<tr><td>70</td><td>일흔 (U+C77C, U+D754)</td></tr>
<tr><td>80</td><td>여든 (U+C5EC, U+B4E0)</td></tr>
<tr><td>90</td><td>아흔 (U+C544, U+D754)</td></tr>
</table>

[<i>Example</i>: The numbering for the items should be represented by the following pattern: 하나, 둘, 셋, ..., 아홉, 열, 열하나, 열둘, ..., 스물, 스물하나, ... <i>end example</i>]
</td></tr>
<tr><td>lowerLetter (Lowercase Latin Alphabet)</td><td>
Specifies that the sequence shall consist of one or more occurrences of a single letter of the Latin alphabet in lower case from the set described below.

This system uses a set of characters to represent the numbers 1 to the length of the language of the alphabet and then those same characters are combined to construct the remaining values.

The characters used by for this numbering format is determined by using the language of the lang element (§17.3.2.20). Specifically:
<ul>
<li>When the script in use is derived from the Latin alphabet (a–z), that alphabet is used. [<i>Example</i>: For Norwegian (Nyorsk), the following Unicode characters are used by this numbering format: U+0061–U+007A, U+00E6, U+00F8, U+00E5. <i>end example</i>]</li>
<li>When the language in use is based on any other system, the characters U+0061–U+007A are used.</li>
</ul>
For values greater than the size of the set, the number
</td></tr>
</table>

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	is constructed by following these steps: 1. Repeatedly subtract the size of the set from the value until the result is equal to or less than the size of the set. 2. The result value determines which character to use and the same character is written once and then repeated for each time the size of the set was subtracted from the original value. [*Example*: For English, the numbering for the items should be represented by the following pattern: a, b, c, …, y, z, aa, bb, cc, …, yy, zz, aaa, bbb, ccc, … *end example*]
lowerRoman (Lowercase Roman Numerals)	Specifies that the sequence shall consist of lowercase roman numerals. This system uses a set of characters to represent the numbers 1, 5, 10, 50, 100, 500, and 1000 and then those are combined with each other to construct the remaining values. The set of characters used by this numbering format is U+0069, U+0076, U+0078, U+006C, U+0063, U+0064, U+006D, respectively. To construct a number that is outside the set, you work from largest groups to smallest following these steps: 1. Create as many groups as possible that contain one thousand in each group. - The symbol representing one thousand (the power of ten represented by that position): m is repeated for the number of groups formed. If no groups are formed, do not write any symbol. 2. Repeat this for groups of nine hundred (cm), five-hundred (d), four-hundred (cd), one-hundred (c), ninety (xc), fifty (l), forty (xl), ten (x), nine (ix), five (v), four (iv) and finally one (i) using the corresponding symbol to indicate the groups (so four-hundred fifty would be cdl and forty-five would be xlv). [*Example*: The numbering for the items should be

Enumeration Value	Description
	represented by the following pattern: i, ii, iii, iv, …, xviii, xix, xx, xxi, … *end example*]
none (No Numbering)	Specifies that the sequence shall not display any numbering.
numberInDash (Number With Dashes)	Specifies that the sequence shall consist of the Arabic numbering surrounded by hyphen-minus characters (U+002D). To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and 0 (U+0030, which represents the number zero) to construct the remaining values. The set of characters used by this numbering format is U+002D (-) and, for values 1–9, U+0031–U+0039, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Divide the value by 10 and write the symbol which represents the remainder. 2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing position. 3. Repeat step 2 until the remaining value is equal to zero. 4. Place the final number between two dashes. [*Example*: The numbering for the items should be represented by the following pattern: - 1 -,- 2 -, - 3 -, …, - 8 -,- 9 -,- 10 -, - 11 -, - 12 -, …, - 18 -, - 19 -, - 20 -, - 21 -, … *end example*]
ordinal (Ordinal)	Specifies that the sequence shall consist of ordinals of the run language. This sequence is a set of strings each of which is the textual representation, in the language of the lang element (§17.3.2.20), of a different unique position in that sequence. [*Example*: The numbering for the items in French

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	should be represented by the following pattern: 1er, 2e, 3e, …, 9e, 10e, 11e, … 19e, 20e, 21e, … *end example*]
ordinalText (Ordinal Text)	Specifies that the sequence shall consist of ordinal text of the run language. This sequence is a set of strings each of which is the textual representation, in the language of the lang element (§17.3.2.20), of a different unique position in that sequence. [*Example*: The numbering for the items in German should be represented by the following pattern: Erste, Zweite, Dritte, …, Neunte, Zehnte, Elfte, …, Neunzehnte, Zwanzigste, Einundzwanzigste, … *end example*]
russianLower (Lowercase Russian Alphabet)	Specifies that the sequence shall consist of one or more occurrences of a single letter of the Russian alphabet in lower case, from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–29 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–29 is U+0430–U+0438, U+043A–U+043F, U+0440–U+0449, U+044B, U+044D, U+044E, and U+044F, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Repeatedly subtract the size of the set (29) from the value until the result is equal to or less than the size of the set. 2. The result value determines which character to use and the same character is written once and then repeated for each time the size of the set was subtracted from the original value. [*Example*: The numbering for the items should be represented by the following pattern: а, б, в, …, ю, я, аа, бб, вв, …, юю, яя, ааа, ббб, ввв, … *end example*]

Enumeration Value	Description
russianUpper (Uppercase Russian Alphabet)	Specifies that the sequence shall consist of one or more occurrences of a single letter of the Russian alphabet in upper case, from the set listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–29 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–29 is U+0410–U+0418, U+041A–U+041F, U+0420–U+0429, U+042B, U+042D, U+042E, and U+042F, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Repeatedly subtract the size of the set (29) from the value until the result is equal to or less than the size of the set. 2. The result value determines which character to use and the same character is written once and then repeated for each time the size of the set was subtracted from the original value. [*Example*: The numbering for the items should be represented by the following pattern: А, Б, В, …, Ю, Я, АА, ББ, ВВ, …, ЮЮ, ЯЯ, ААА, БББ, ВВВ, … *end example*]
taiwaneseCounting (Taiwanese Counting System)	Specifies that the sequence shall consist of sequential numbers from the Taiwanese counting system. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and ○ (U+25CB, which represents the number zero) to construct the remaining values. The set of characters used by this numbering format for values 1–10 is U+4E00, U+4E8C,U+4E09, U+56DB, U+4E94, U+516D, U+4E03, U+516B, U+4E5D, and U+5341, respectively. For values greater than the size of the set, the number

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>is constructed by following these steps:

1. Divide the value by 10 and write the symbol which represents the remainder.
   - If the quotient is less than 10 then write 十 to the left of the symbol, which represents the remainder.
2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing positions.
3. Repeat step 2 until the remaining value is equal to zero.

[*Example*: The numbering for the items should be represented by the following pattern: 一, 二, 三, …, 九, 十, 十一, 十二, …, 十九, 二十, 二十一, …, 九十九, 一○○, 一○一, … *end example*]</td></tr>
<tr><td>taiwaneseCountingThousand (Taiwanese Counting Thousand System)</td><td>Specifies that the sequence shall consist of sequential numbers from the Taiwanese counting thousand system.

This system uses a set of characters to represent the numbers 1–10 and then those are combined with additional characters to construct the remaining characters.

The set of characters used by this numbering format is U+4E00, U+4E8C, U+4E09, U+56DB, U+4E94, U+516D, U+4E03, U+516B, U+4E5D, U+842C, U+5343, U+5341, U+767E, and U+96F6.

To construct a number that is beyond the set but less than hundred thousand, you work from largest groups to smallest following these steps:

1. Create as many groups as possible that contain ten thousand in each group.
   - Write down the symbol representing that value (1–9):

<table>
<tr><th>Digit</th><th>Character</th></tr>
<tr><td>1</td><td>一 (U+4E00)</td></tr>
<tr><td>2</td><td>二 (U+4E8C)</td></tr>
<tr><td>3</td><td>三 (U+4E09)</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>4</td><td>四 (U+56DB)</td></tr>
<tr><td>5</td><td>五 (U+4E94)</td></tr>
<tr><td>6</td><td>六 (U+516D)</td></tr>
<tr><td>7</td><td>七 (U+4E03)</td></tr>
<tr><td>8</td><td>八 (U+516B)</td></tr>
<tr><td>9</td><td>九 (U+4E5D)</td></tr>
</table>
<ul>
<li>If no groups are formed, do not write any characters.</li>
<li>If groups were formed, write down the symbol representing ten thousand: 萬</li>
</ul>
<ol start="2">
<li>Repeat step 1 this for groups of one thousand (千) using the corresponding symbol to indicate the groups (so five thousand would be 五千).
<ul><li>If the original value was between 10,000 and 100,000 and If no groups are formed (and the number is not a multiple of ten) write the symbol 零 instead (so ten thousand and five would be 一萬零五)</li></ul></li>
<li>Repeat step 1 this for groups of one hundred (百) using the corresponding symbol to indicate the groups (so five hundred would be 五百).</li>
<li>The value 10 uses the symbol 十. For all other groups of ten, use the corresponding symbol to indicate the groups (so fifty would be 五十).
<ul><li>If the original value was between 100 and 1000 and If no groups are formed (and the number is not a multiple of ten) write the symbol 零 instead (so one hundred and five would be 一百零五)</li></ul></li>
<li>Write down the symbol for the remaining number.</li>
</ol>
<p>If the number is larger than one hundred thousand but less than one hundred million, you perform the cycle for the numbers above one thousand but use two characters to represent each group. So, for example, groups of one million are represented as one hundred ten thousands (十萬). An additional symbol for counting groups is introduced at one hundred million (because 10 million is one thousand ten thousands).</p>
<p>[<i>Example</i>: The numbering for the items should be</p>
</td></tr>
</table>

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	represented by the following pattern: 一, 二, 三, ..., 八, 九, 一十, 十一, 十二, ..., 十九, 二十, 二十一, ..., 九九, 一百, 一百零一, ... *end example*]
taiwaneseDigital (Taiwanese Digital Counting System)	Specifies that the sequence shall consist of sequential numbers from the Taiwanese digital counting system. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and ○ (represents the number zero) to construct the remaining values. The set of characters used by this numbering format for values 0–9 is U+25CB, U+4E00, U+4E8C,U+4E09, U+56DB, U+4E94, U+516D, U+4E03, U+516B, and U+4E5D, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Divide the value by 10 and write the symbol which represents the remainder. 2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing position. 3. Repeat step 2 until the remaining value is equal to zero. [*Example*: The numbering for the items should be represented by the following pattern: 一, 二, ..., 八, 九, 一○,一一, 一二, ..., 一八, 一九, 二○, 二一, ... *end example*]
thaiCounting (Thai Counting System)	Specifies that the sequence shall consist of sequential numbers from the Thai counting system. This sequence is a set of strings each of which is the full name, in Thai, of the next value in that sequence. [*Example*: The numbering for the items should be represented by the following pattern: หนึ่ง, สอง, สาม, สี่, ห้า, หก, เจ็ด, แปด, เก้า, สิบ, ... *end example*]
thaiLetters (Thai Letters)	Specifies that the sequence shall consist of one or more occurrences of a single Thai letter from the set

Enumeration Value	Description
	listed below. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–41 and then repeats those same characters using the logic defined below to construct all other values. The set of characters used by this numbering format for values 1–41 is U+0E01, U+0E02, U+0E04, U+0E07–U+0E23, U+0E25, and U+0E27–U+0E2E, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Repeatedly subtract the size of the set (41) from the value until the result is equal to or less than the size of the set. 2. The result value determines which character to use and the same character is written once and then repeated for each time the size of the set was subtracted from the original value. [*Example*: The numbering for the items should be represented by the following pattern: ก, ข, ค, ..., อ, ฮ, กก, ขข, คค, ..., ออ, ฮฮ, กกก, ขขข, คคค, ... *end example*]
thaiNumbers (Thai Numerals)	Specifies that the sequence shall consist of Thai numerals. To determine the text that is displayed for any value, this sequence specifies a set of characters that represent positions 1–9 and then those same characters are combined with each other and ๐ (represents the number zero) to construct the remaining values. The set of characters used by this numbering format for values 0–9 is U+0E50, U+0E51, U+0E52, U+0E53, U+0E54, U+0E55, U+0E56, U+0E57, U+0E58, and U+0E59, respectively. For values greater than the size of the set, the number is constructed by following these steps: 1. Divide the value by 10 and write the symbol

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	which represents the remainder. 2. Divide the quotient of the previous division by 10 and write the symbol, which represents the remainder, to the left of the existing position. 3. Repeat step 2 until the remaining value is equal to zero. [*Example*: The numbering for the items should be represented by the following pattern: ๑, ๒, ๓, …, ๘, ๙, ๑๐, ๑๑, ๑๒, …, ๑๘, ๑๙, ๒๐, ๒๑, ๒๒, …, ๒๘, ๒๙, … *end example*]
upperLetter (Uppercase Latin Alphabet)	Specifies that the sequence shall consist of one or more occurrences of a single letter of the Latin alphabet in upper case, from the set listed below. This system uses a set of characters to represent the numbers 1 to the length of the language of the alphabet and then those same characters are combined to construct the remaining values. The characters used by this numbering format is determined by using the language of the lang element (§17.3.2.20). Specifically: • When the script in use is derived from the Latin alphabet (A–Z), that alphabet is used. [*Example*: For Norwegian (Nyorsk), the following Unicode characters are used by this numbering format: U+0041–U+005A, U+00C6, U+00D8, U+00C5. *end example*] • When the language in use is based on any other system, the characters U+0041–U+005A are used. For values greater than the size of the set, the number is constructed by following these steps: 1. Repeatedly subtract the size of the set from the value until the result is equal to or less than the size of the set. 2. The result value determines which character to use, and the same character is written once and then repeated for each time the size of the set was subtracted from the original value.

Enumeration Value	Description
	[*Example*: For English, the numbering for the items should be represented by the following pattern: A, B, C, …, Y, Z, AA, BB, CC, …, YY, ZZ, AAA, BBB, CCC, … *end example*]
upperRoman (Uppercase Roman Numerals)	Specifies that the sequence shall consist of uppercase roman numerals. This system uses a set of characters to represent the numbers 1, 5, 10, 50, 100, 500, and 1000 and then those characters are combined with each other to construct the remaining values. The set of characters used by this numbering format is U+0049, U+0056, U+0058, U+004C, U+0043, U+0044, U+004D, respectively. To construct a number that is outside the set, you work from largest groups to smallest following these steps: 1. Create as many groups as possible that contain one thousand in each group. • The symbol representing one thousand (the power of ten represented by that position): M is repeated for the number of groups formed. If no groups are formed, do not write any symbol. 2. Repeat this for groups of nine hundred (CM), five-hundred (D), four-hundred (CD), one-hundred (C), ninety (XC), fifty (L), forty (XL), ten (X), nine (IX), five (V), four (IV) and finally one (I) using the corresponding symbol to indicate the groups (so four-hundred fifty would be CDL and forty-five would be XLV). [*Example*: The numbering for the items should be represented by the following pattern: I, II, III, IV, …, XVIII, XIX, XX, XXI, … *end example*]
vietnameseCounting (Vietnamese Numerals)	Specifies that the sequence shall consist of Vietnamese numerals. This sequence is a set of strings each of which is the full name, in Vietnamese, of the next value in that sequence.

Enumeration Value	Description
	[*Example*: một, hai, ba, bốn, năm, sáu, bảy, tám, chín, mười. *end example*]

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_NumberFormat) is located in §A.1. *end note*]

17.18.60 ST_ObjectDrawAspect (Embedded Object Representations)

This simple type specifies the ways in which embedded objects are displayed in the application.

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
content (Snapshot)	The object's presentation is a picture of the contained document (provided by the object's server application).
icon (Icon)	The object's presentation is an icon.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_ObjectDrawAspect) is located in §A.1. *end note*]

17.18.61 ST_ObjectUpdateMode (Embedded Object Update Modes)

This simple type specifies how an embedded object is updated.

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
always (Server Application Update)	The object is updated whenever the object's server application indicates there is new data available.
onCall (User Update)	The object is updated when the user chooses to update it.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_ObjectUpdateMode) is located in §A.1. *end note*]

17.18.62 ST_PageBorderDisplay (Page Border Display Options)

This simple type specifies the pages in the parent section on which the page border shall be printed.

[*Example*: Consider a section in a document for which the page border must only be printed on the first page. This setting is specified using the following WordprocessingML:

```
<w:pgBorders w:display="firstPage">
  …
</w:pgBorders>
```

The display attribute with a value of firstPage specifies that only the first page must display the page border defined for this section. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
allPages (Display Page Border on All Pages)	Specifies that the page border shall be displayed on all pages in the parent section.
firstPage (Display Page Border on First Page)	Specifies that the page border shall be displayed on only the first page in the parent section.
notFirstPage (Display Page Border on All Pages Except First)	Specifies that the page border shall be displayed on only the first page in the parent section.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PageBorderDisplay) is located in §A.1. *end note*]

17.18.63 ST_PageBorderOffset (Page Border Positioning Base)

This simple type specifies how the relative positioning of the page borders shall be calculated.

If the value of this attribute is text, then the space attribute on each page border shall be interpreted as the distance from the text margins that shall be left before the page border.

[*Example*: Consider the following WordprocessingML fragment:

```
<w:pgBorders w:offsetFrom="page">
  <w:top w:val="dashed" w:space="24" />
  <w:left w:val="dashed" w:space="24" />
  <w:bottom w:val="dashed" w:space="24"/>
  <w:right w:val="dashed" w:space="24"/>
</w:pgBorders>
```

This fragment specifies that the page borders must be indented 24 points from the page extents.

This is distinct from the following fragment with identical space attribute values:

©ISO/IEC 2012 – All rights reserved

```
<w:pgBorders w:offsetFrom="text">
  <w:top w:val="dashed" w:space="24" />
  <w:left w:val="dashed" w:space="24" />
  <w:bottom w:val="dashed" w:space="24"/>
  <w:right w:val="dashed" w:space="24"/>
</w:pgBorders>
```

In this case, the page borders is offset by 24 points, but in this case, that offset is calculated relative to the text margins. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

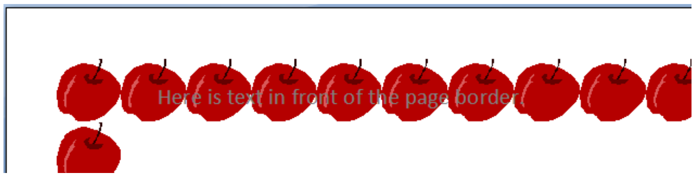
Enumeration Value	Description
page (Page Border Is Positioned Relative to Page Edges)	Specifies that the space attribute on each page border shall be interpreted as the distance from the edge of the page that shall be left before the page border.
text (Page Border Is Positioned Relative to Text Extents)	Specifies that the space attribute on each page border shall be interpreted as the distance from the edge of the text extents (text margins) that shall be left before the page border..

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PageBorderOffset) is located in §A.1. *end note*]

17.18.64 ST_PageBorderZOrder (Page Border Z-Order)

This simple type specifies whether the page border is positioned above or below intersecting texts and objects in this document.

[*Example*: Consider a document in which the page border must be displayed below any intersecting text as follows:



Here is text in front of the page border

This setting is specified by setting the value of an attribute with to back, which specifies that the page border must be displayed behind all intersecting text and objects. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
back (Page Border Behind Text)	Specifies that the page border shall be rendered beneath any text or object which intersects it - effectively placing it at the lowest z-order on the page.
front (Page Border Ahead of Text)	Specifies that the page border shall be rendered above any text or object which intersects it - effectively placing it at the highest z-order on the page.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PageBorderZOrder) is located in §A.1. *end note*]

17.18.65 ST_PageOrientation (Page Orientation)

This simple type specifies the orientation of all pages in the parent section. This information is used to determine the actual paper size to use when printing the file.

[*Example*: Pages 11" wide by 8.5" long in landscape mode use 8.5"x11" paper, because the width and height are reversed for pages in this landscape section with respect to the printed page. *end example*]

[*Example*: Consider the following WordprocessingML:

```
<w:pgSz w:w="15840" w:h="12240" w:orient="landscape" />
```

Although the page width is 11", and page height is 8.5", according to the w and h attributes, because the orient attribute is set to `landscape`, pages in this section are printed on 8.5x11" paper in landscape mode. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
landscape (Landscape Mode)	Specifies that pages in this section shall be printed in landscape mode, which prints the page contents with a 90 degree rotation with respect to the normal page orientation.
portrait (Portrait Mode)	Specifies that pages in this section shall be printed in portrait mode.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PageOrientation) is located in §A.1. *end note*]

17.18.66 ST_Pitch (Font Pitch Value)

This simple type specifies the possible values for the font pitch of a font.

[*Example*: Consider the following information stored for a single font:

```
<w:font w:name="Courier New">
  <w:pitch w:val="fixed" />
  …
</w:font>
```

The pitch element specifies via its val attribute value of `fixed` that this is a fixed width font. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
default (Default)	Specifies that no information is available about the pitch of a font.
fixed (Fixed Width)	Specifies that this is a fixed width font.
variable (Proportional Width)	Specifies that this is a proportional width font.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Pitch) is located in §A.1. *end note*]

17.18.67 ST_PixelsMeasure (Measurement in Pixels)

This simple type specifies that its contents contain a positive whole number, whose contents consist of a measurement in pixels.

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of 96 whose simple type is ST_PixelsMeasure. This attribute value specifies a size of 96 pixels (one inch on a 96 pixels per inch display). *end example*]

This simple type's contents are a restriction of the ST_UnsignedDecimalNumber datatype (§22.9.2.16).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PixelsMeasure) is located in §A.1. *end note*]

17.18.68 ST_PointMeasure (Measurement in Points)

This simple type specifies that its contents contain a positive whole number, whose contents consist of a measurement in points (equivalent to 1/72nd of an inch).

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of 24 whose simple type is ST_PointMeasure. This attribute value specifies a size in points (24 points = 1/3 of an inch). *end example*]

This simple type's contents are a restriction of the ST_UnsignedDecimalNumber datatype (§22.9.2.16).

[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_PointMeasure) is located in §A.1. *end note*]

17.18.69 ST_Proof (Proofing State Values)

This simple type specifies the values which can be used to indicate the status of a given hosting application's grammar and spell checking when a given WordprocessingML document was last saved.

[*Example*: Consider a WordprocessingML document that is saved by a hosting application whose grammar checking engine had completed checking the grammar in the given WordprocessingML document, but whose spell checking engine had not completed checking the spelling in the given WordprocessingML document . In this instance, the following WordprocessingML must be written in the document settings:

```
<w:proofState w:spelling="dirty" w:grammar="clean" />
```

The proofState element's attributes spelling and grammar have the ST_Proof simple type enumeration values `dirty` and `clean` respectively, specifying that the hosting application's spell checking engine had not completed checking the spelling of the document, and that the hosting application's grammar checking engine had completed checking the grammar of the document, when the WordprocessingML document was last saved. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
clean (Check Completed)	Specifies that the given proofing engine completed checking the document when it was last saved.
dirty (Check Not Completed)	Specifies that the given proofing engine did not complete checking the document when it was last saved.

[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_Proof) is located in §A.1. *end note*]

17.18.70 ST_ProofErr (Proofing Error Type)

This simple type specifies the possible values for the types of proofing error markers which can appear in the contents of a WordprocessingML document to indicate the last known state of any spell- and grammar-checking performed on the contents of this document.

[*Example*: Consider the following paragraph consisting of two misspelled words, where the second word has been explicitly flagged as not being a spelling error. This paragraph would consist of the following WordprocessingML markup:

```
<w:p>
  <w:proofErr w:type="spellStart"/>
  <w:r>
    <w:t>erqwt</w:t>
  </w:r>
  <w:proofErr w:type="spellEnd"/>
  <w:r>
    <w:t xml:space="preserve"> werewr</w:t>
  </w:r>
</w:p>
```

The proofErr elements with a val attribute value of `spellStart` and `spellEnd`, respectively delimit the start and end the content in this paragraph which is stored as a spelling error. Since the second word is not included in that range, it is not stored as a spelling error. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
gramEnd (End of Region Marked as Grammatical Error)	Specifies that this proofing error marker shall indicate the start of a region to be marked as a grammatical error in the document.
gramStart (Start of Region Marked as Grammatical Error)	Specifies that this proofing error marker shall indicate the end of a region to be marked as a grammatical error in the document.
spellEnd (End of Region Marked as Spelling Error)	Specifies that this proofing error marker shall indicate the end of a region to be marked as a spelling error in the document.
spellStart (Start of Region Marked as Spelling Error)	Specifies that this proofing error marker shall indicate the start of a region to be marked as a spelling error in the document.

[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_ProofErr) is located in §A.1. *end note*]

17.18.71 ST_PTabAlignment (Absolute Position Tab Alignment)

This simple type specifies the alignment of an absolutely positioned tab character in a document. This alignment value determines the position on the line to which this absolute tab shall advance, as well as the alignment of the text entered after the alignment tab character position.

[*Example*: Consider a positional tab stop in a WordprocessingML document who must move to the left edge of the text margins and whose subsequent text should be left aligned. This positional tab stop would be defined as follows:

```
<w:ptab w:alignment="left" w:relativeTo="margin" … />
```

The alignment attribute has a value of `left`, which specifies that this custom tab stop must align on the left edge of the line relative to the text margin. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
center (Center)	Specifies that the positional tab should be center aligned on the line relative to the specified base (the text margins with or without indents), and that the text at that location shall be center aligned.
left (Left)	Specifies that the positional tab should be left aligned on the line relative to the specified base (the text margins with or without indents), and that the text at that location shall be left aligned.
right (Right)	Specifies that the positional tab should be right aligned on the line relative to the specified base (the text margins with or without indents), and that the text at that location shall be right aligned.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PTabAlignment) is located in §A.1. *end note*]

17.18.72 ST_PTabLeader (Absolute Position Tab Leader Character)

This simple type specifies the characters which can be used to fill in the space created by a positional tab. This character shall be repeated as required to completely fill the tab spacing generated by the positional tab character.

[*Example*: Consider a positional tab stop which should be preceded by a sequence of underscore characters, as follows:

_______________Text at the positional tab stop

This tab stop would have a leader attribute value of `underscore`, indicating that the tab stop must be preceded by underscore characters as needed to fill the tab spacing. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

©ISO/IEC 2012 – All rights reserved

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
dot (Dot Leader Character)	Specifies that the leader character for this positional tab stop shall be a dot. [*Example*: .... Text at absolute tab. *end example*]
hyphen (Hyphen Leader Character)	Specifies that the leader character for this positional tab stop shall be a hyphen. [*Example*: --------------------- Text at absolute tab. *end example*]
middleDot (Centered Dot Leader Character)	Specifies that the leader character for this positional tab stop shall be a centered dot. [*Example*: ··························· Text at absolute tab. *end example*]
none (No Leader Character)	Specifies that there shall be no leader character for this positional tab. [*Example*: Text at absolute tab. *end example*]
underscore (Underscore Leader Character)	Specifies that the leader character for this positional tab stop shall be an underscore. [*Example*: ______________ Text at absolute tab. *end example*]

©ISO/IEC 2012 – All rights reserved

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PTabLeader) is located in §A.1. *end note*]

17.18.73 ST_PTabRelativeTo (Absolute Position Tab Positioning Base)

Specifies the possible extents which can be used to calculate the absolute positioning of this positional tab character.

[*Example*: Consider a positional tab stop in a WordprocessingML document that should have a resulting position that is centered on the text margins, ignoring both any custom tab stops and any text indents on the paragraph. This positional tab stop would be defined as follows:

```
<w:ptab w:relativeTo="margin" … />
```

The relativeTo attribute specifies that this absolute position tab stop must be relative to the `margin`. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
indent (Relative To Indents)	Specifies that the absolute positioning of the tab shall be relative to the indents.
margin (Relative To Text Margins)	Specifies that the absolute positioning of the tab shall be relative to the margins.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PTabRelativeTo) is located in §A.1. *end note*]

17.18.74 ST_RestartNumber (Footnote/Endnote Numbering Restart Locations)

This simple type specifies the possible values for when the automatic numbering of footnotes or endnotes shall be restarted.

[*Example*: Consider a WordprocessingML document where the numbering for its endnotes must be restarted after each section must be restarted after each page. This setting is represented by the following WordprocessingML:

```
<w:footnotePr>
  …
  <w:numRestart w:val="eachSect" />
  …
</w:footnotePr>
```

The val attribute value of `eachSect` specifies that numbering must be restarted after each section. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
continuous (Continue Numbering From Previous Section)	Specifies that the numbering of footnotes or endnotes shall continue from the previous section in the document.
eachPage (Restart Numbering On Each Page)	Specifies that the numbering of footnotes or endnotes shall be restarted to its starting value for each unique page in the document.
eachSect (Restart Numbering For Each Section)	Specifies that the numbering of footnotes or endnotes shall be restarted to its starting value for each unique section in the document.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_RestartNumber) is located in §A.1. *end note*]

17.18.75 ST_RubyAlign (Phonetic Guide Text Alignment)

This simple type specifies the possible alignment settings which can be used to determine the placement of phonetic guide text with respect to the base text when this phonetic guide is displayed.

[*Example*: Consider a run of phonetic guide text which must have the ruby text positioned to the far left of the base text. This constraint is specified using the following WordprocessingML:

```
<w:rubyPr>
  …
  <w:rubyAlign w:val="left"/>
  …
</w:rubyPr>
```

The rubyAlign property is `left` for the phonetic guide, so the ruby text is displayed on the left side of the base text. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
center (Center)	Specifies that the phonetic guide text shall be centered with respect to the base text in this document. [*Example*: guide text this is a test center

Enumeration Value	Description
	end example]
distributeLetter (Distribute All Characters)	Specifies that the phonetic guide text shall be distributed with respect to the base text in this document. This type of justification shall equally affect the inter-word spacing on each line as well as the inter-character spacing between each word when justifying its contents - that is, an equal amount of additional character pitch shall be added to all characters on the line. [*Example*: guide text a test distribute letter *end example*]
distributeSpace (Distribute all Characters w/ Additional Space On Either Side)	Specifies that the phonetic guide text shall be distributed with respect to the base text in this document, with additional space added to the guide text to ensure it is indented with respect to the base text. This type of justification shall equally affect the inter-word spacing on each line as well as the inter-character spacing between each word when justifying its contents - that is, an equal amount of additional character pitch shall be added to all characters on the line. As well, an additional space is added before and after the guide text to ensure it is indented with respect to the base text. [*Example*: guide text a test distribute space *end example*]
left (Left Aligned)	Specifies that the phonetic guide text shall be left aligned with respect to the base text in this document. [*Example*: guide text this is a test left

Enumeration Value	Description
	end example]
right (Right Aligned)	Specifies that the phonetic guide text shall be right aligned with respect to the base text in this document. [*Example*: guide text this is a test right *end example*]
rightVertical (Vertically Aligned to Right of Base Text)	Specifies that the phonetic guide text shall be right aligned with respect to the base text in this document, and shall always be displayed vertically and to the right of the base text, regardless of the alignment of the base text. [*Example*: this is a test right vertical guide text *end example*]

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_RubyAlign) is located in §A.1. *end note*]

17.18.76 ST_SdtDateMappingType (Date Storage Format Types)

This simple type specifies then possible types of translations which can be performed on the displayed date in a date picker structured document tag when the current contents are saved into the associated custom XML data via the dataBinding element (§17.5.2.6).

[*Example*: Consider the following date picker structured document tag:

```
<w:sdt>
  <w:sdtPr>
    <w:date w:fullDate="01-01-2006T06:30:00Z">
      <w:storeMappedDateAs w:val="text"/>
      …
    </w:date>
  </w:sdtPr>
  <w:sdtContent>
    <w:r>
      <w:t>January 1</w:t>
    </w:r>
  </w:sdtContent>
</w:sdt>
```

The value of the storeMappedDateAs element's attribute value is `text`, therefore the current run contents must be sent to the mapped XML element without any translation (in this case, the value must be `January 1`). *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
date (XML Schema Date Format)	Specifies that the date specified in the parent date picker structured document tag shall be converted to the xsd:date format when stored in a mapped XML element.
dateTime (XML Schema DateTime Format)	Specifies that the date specified in the parent date picker structured document tag shall be converted to the xsd:dateTime format when stored in a mapped XML element.
text (Same As Display)	Specifies that no translation shall be performed on the displayed date when stored in a mapped XML element - the mapped contents shall be the same as the displayed contents.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SdtDateMappingType) is located in §A.1. *end note*]

17.18.77 ST_SectionMark (Section Type)

Specifies the kind of the current section.

[*Example*: Consider a section that must start on the next page in the document. The WordprocessingML specifying this would look like:

```
<w:sectPr>
  …
  <w:type w:val="nextPage"/>
</w:sectPr>
```

The nextPage value specifies that this section starts on the next page. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

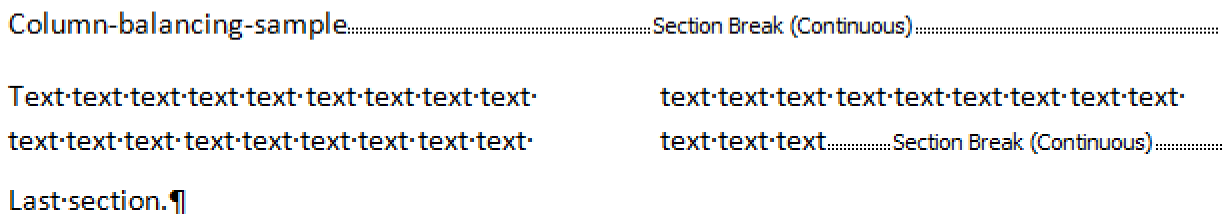
This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
continuous (Continuous Section Break)	Specifies a continuous section break, which begin the new section on the following paragraph. This means that continuous section breaks might not specify certain page-level section properties, since they are inherited from the following section. These breaks, however, can specify other section properties, such as line numbering and footnote/endnote settings. If a footnote reference (§17.11.14) occurs on the same page as a section break of this kind, the new section shall begin on the following page.
evenPage (Even Page Section Break)	Specifies an even page section break, which begins the new section on the next even-numbered page, leaving the next odd page blank if necessary.
nextColumn (Column Section Break)	Specifies a column section break, which begins the new section on the following column on the page.
nextPage (Next Page Section Break)	Specifies a next page section break, which begins the new section on the following page.
oddPage (Odd Page Section Break)	Specifies an odd page section break, which begins the new section on the next odd-numbered page, leaving the next even page blank if necessary.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SectionMark) is located in §A.1. *end note*]

[*Note*: A continuous section break balances the content of the previous section. *end note*]

[*Example*: Consider the following sample document:



Column-balancing-sample Section Break (Continuous)

Text·text·text·text·text·text·text·text·text·
text·text·text·text·text·text·text·text·text·

text·text·text·text·text·text·text·text·text·
text·text·text Section Break (Continuous)

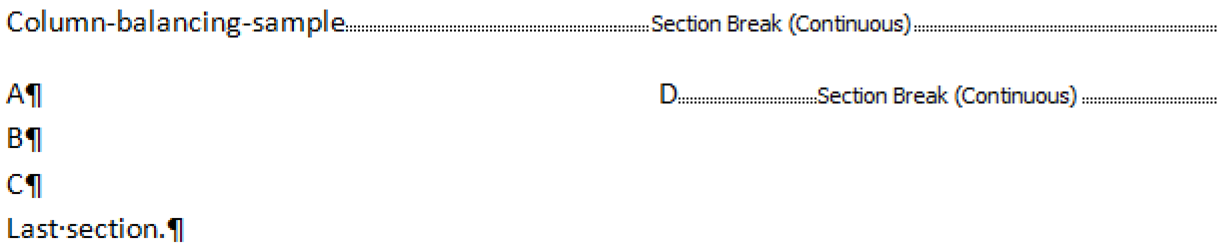
Last·section.¶

The second section (“Text text text …”) has two columns and the third section (“Last section”) has the type “continuous”. As a result, the content of the second section is balanced before the third section is laid out. *end example*]

The process of “balancing” is defined by starting the next section at the minimum section height such that all content constraints are met.

[*Example*:

Consider the following sample document:



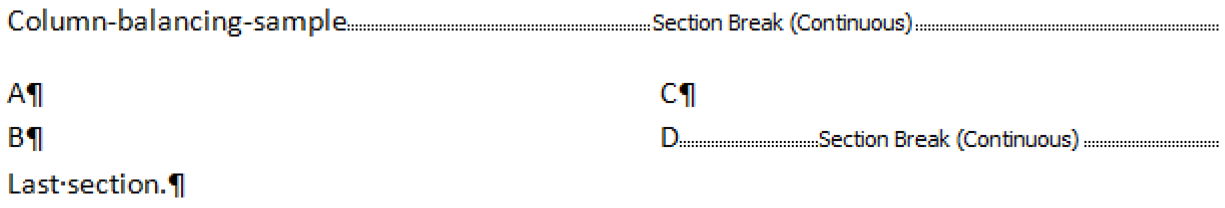
Column-balancing-sample Section Break (Continuous)

A¶
B¶
C¶

D Section Break (Continuous)

Last·section.¶

The above balancing is incorrect, as the height of the second section can be reduced such that the following minimal balancing is achieved:



Column-balancing-sample Section Break (Continuous)

A¶
B¶

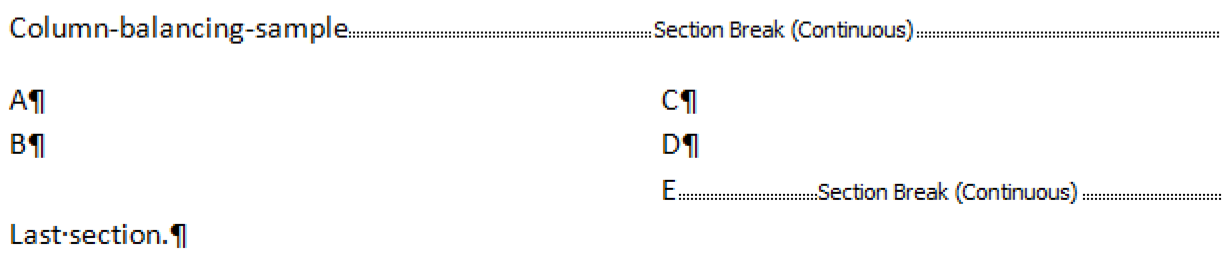
C¶
D Section Break (Continuous)

Last·section.¶

end example]

[*Example*:

The document



Column-balancing-sample Section Break (Continuous)

A¶

B¶

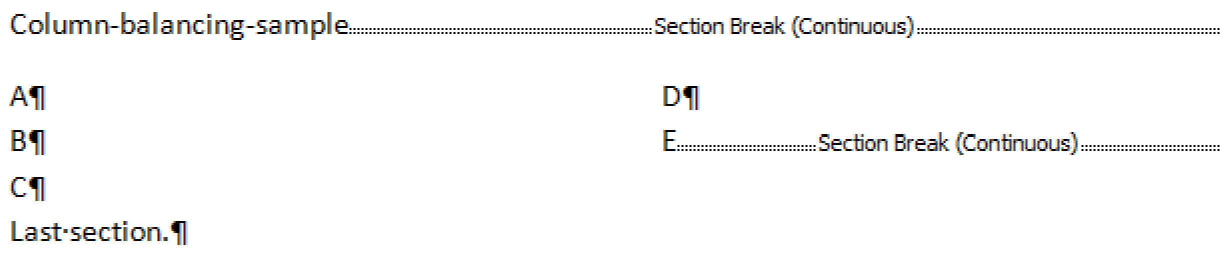
C¶

D¶

E Section Break (Continuous)

Last·section.¶

has incorrect column balancing, as the content is not laid out with respect to the implied constraints—although the section height is minimal. The correct minimal balancing looks like this:



Column-balancing-sample Section Break (Continuous)

A¶

B¶

C¶

Last·section.¶

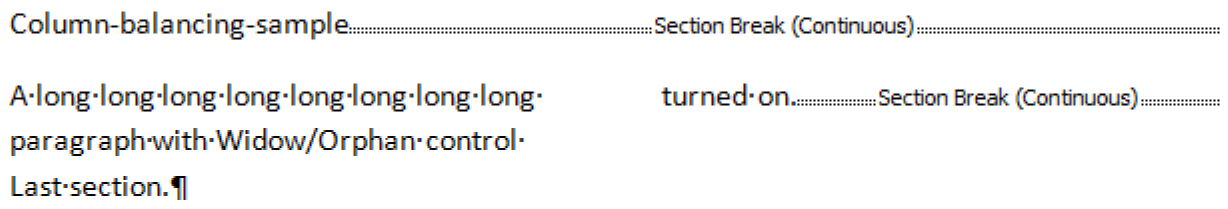
D¶

E Section Break (Continuous)

end example]

[*Example*:

The following section balancing



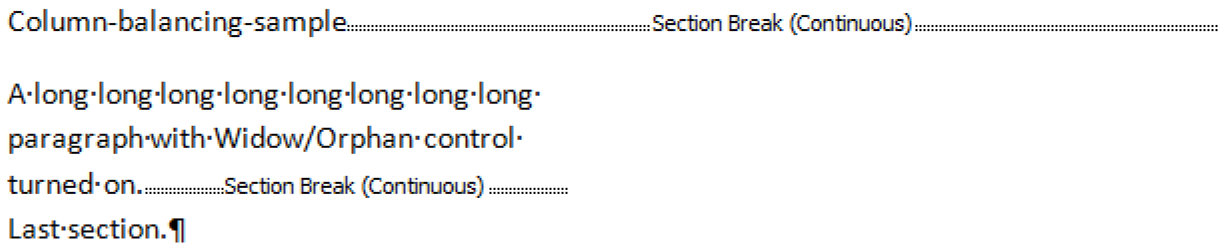
Column-balancing-sample Section Break (Continuous)

A·long·long·long·long·long·long·long·long· paragraph·with·Widow/Orphan·control·

Last·section.¶

turned·on. Section Break (Continuous)

is incorrect, as the “Widow/Orphan” constraints are not met. The correct balancing looks like this:



Column-balancing-sample Section Break (Continuous)

A·long·long·long·long·long·long·long·long· paragraph·with·Widow/Orphan·control· turned·on. Section Break (Continuous)

Last·section.¶

end example]

17.18.78 ST_Shd (Shading Patterns)

This simple type specifies the pattern that shall be used to lay the pattern color over the background color for shading.

This pattern consists of a mask that is applied over the background shading color to get the locations where the pattern color should be shown. Each of these possible masks is shown in the enumeration values below. In each example, an 8 pixel by 8 pixel mask is displayed where black has been used as the fill color (the parent element's fill attribute), and white has been used as the pattern color (the parent element's color attribute). When the shading is applied, the mask is tiled as necessary to match the size of the shaded area.

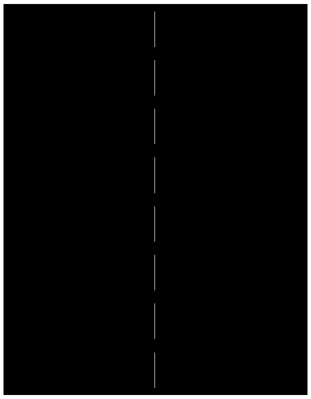
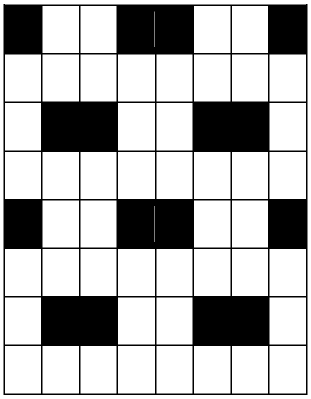
[*Example*: Consider a shaded paragraph which uses a 10 percent foreground fill, resulting in the following WordprocessingML:

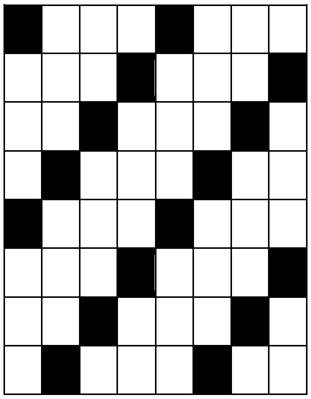
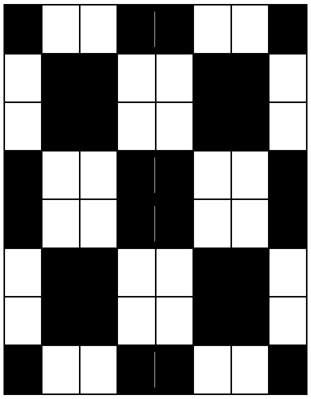
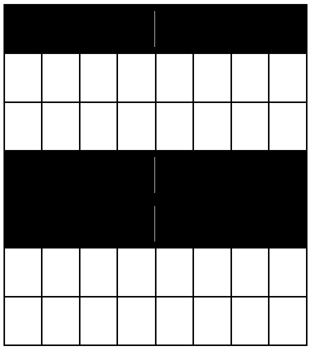
```
<w:shd w:val="pct10" …/>
```

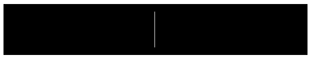
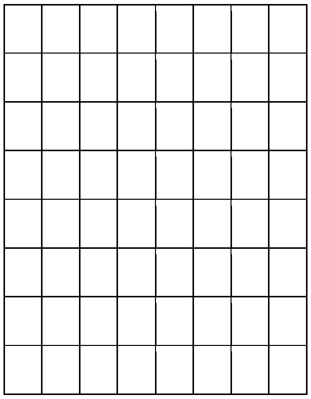
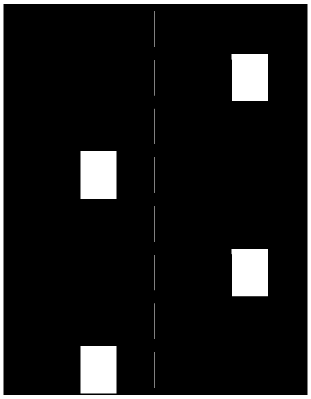
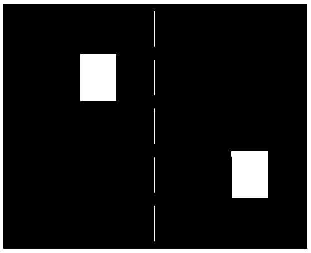
This shading val is `pct10`, indicating that the border style is a 10 percent foreground fill mask. *end example*]

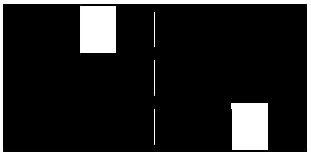
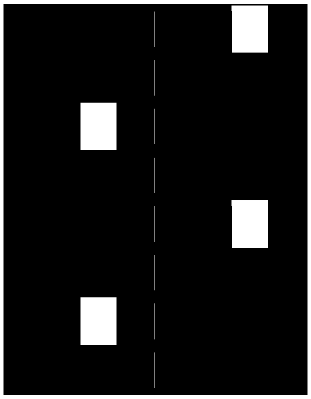
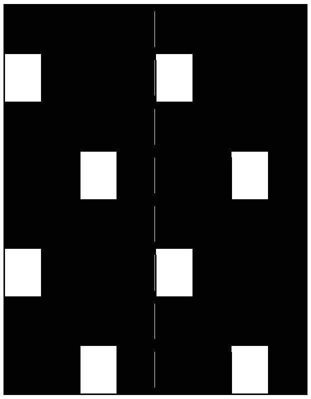
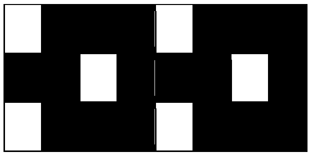
This simple type's contents are a restriction of the W3C XML Schema string datatype.

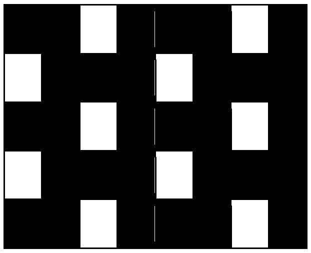
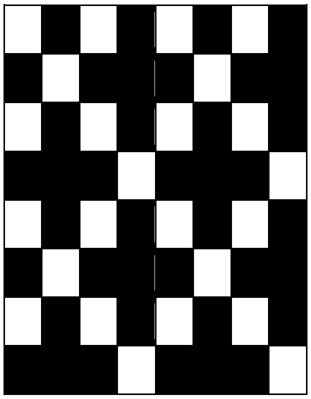
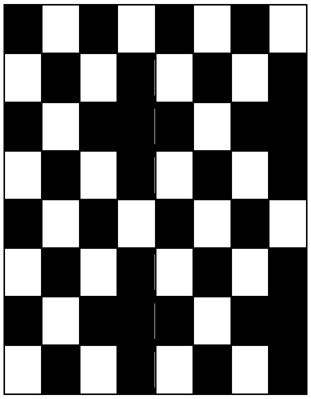
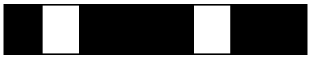
This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
clear (No Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
diagCross (Diagonal Cross Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

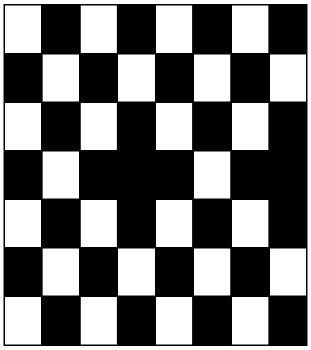
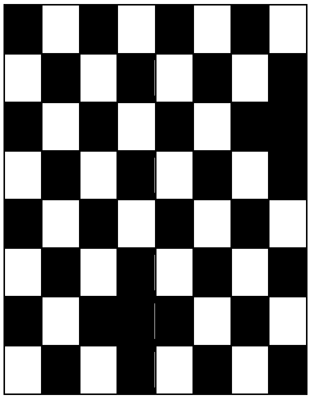
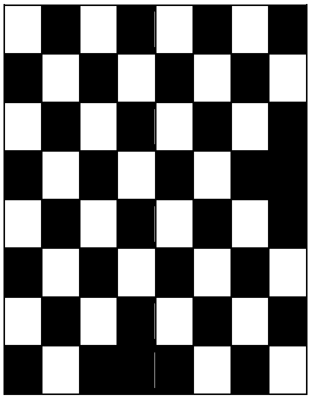
Enumeration Value	Description
diagStripe (Diagonal Stripe Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
horzCross (Horizontal Cross Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
horzStripe (Horizontal Stripe Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

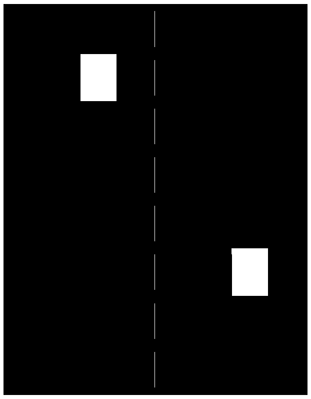
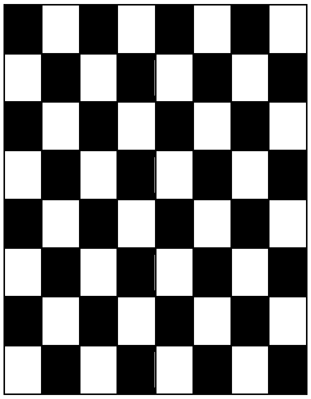
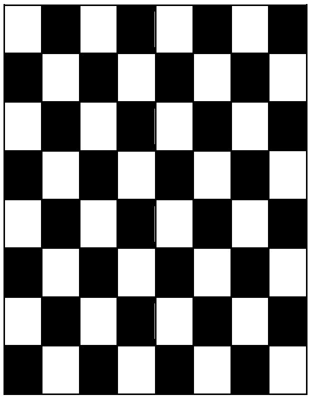
Enumeration Value	Description
	
nil (No Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct10 (10% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct12 (12.5% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

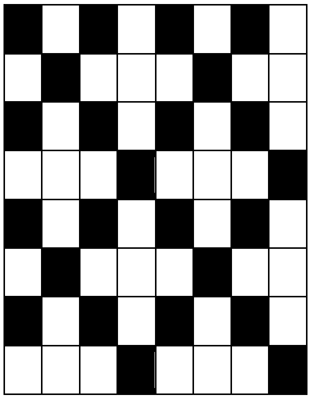
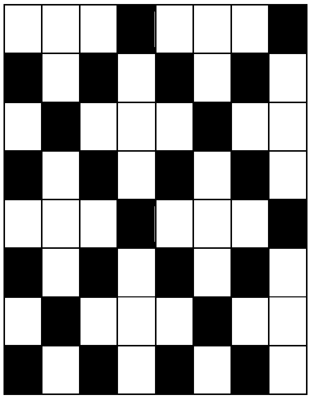
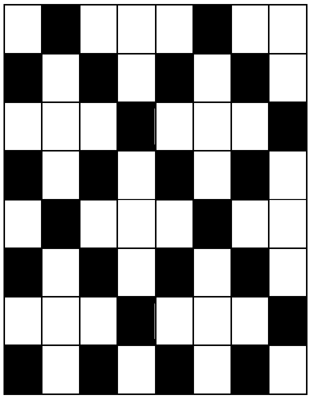
Enumeration Value	Description
	
pct15 (15% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct20 (20% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct25 (25% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

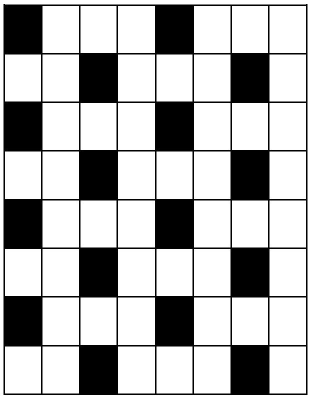
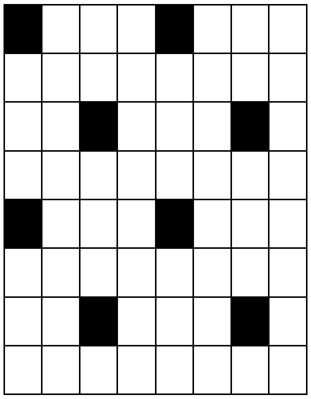
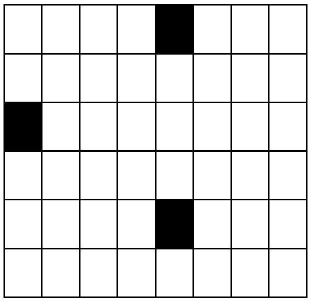
Enumeration Value	Description
	
pct30 (30% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct35 (35% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct37 (37.5% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

©ISO/IEC 2012 – All rights reserved

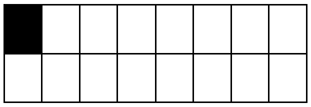
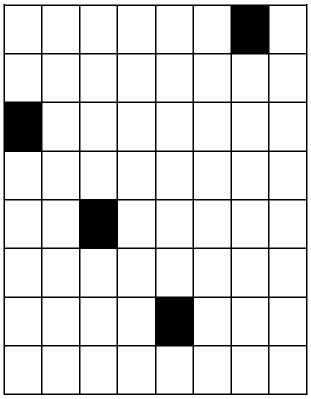
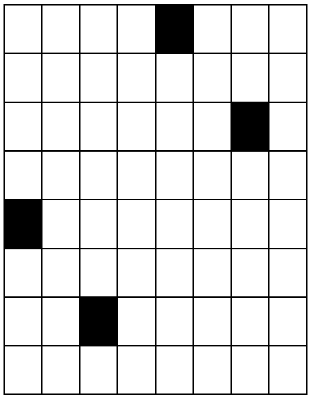
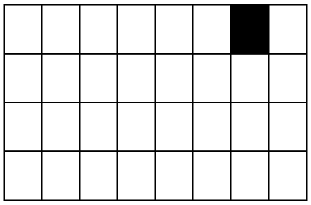
Enumeration Value	Description
	
pct40 (40% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct45 (45% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct5 (5% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern:

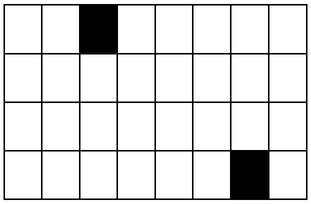
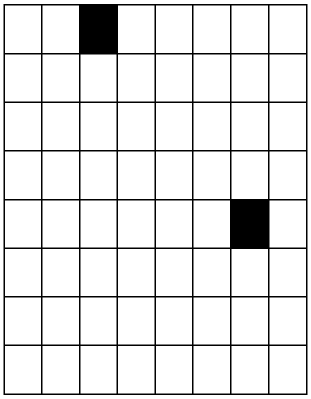
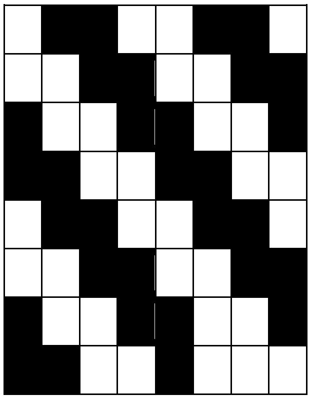
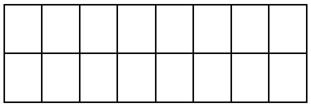
Enumeration Value	Description
	
pct50 (50% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct55 (55% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

Enumeration Value	Description
pct60 (60% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct62 (62.5% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct65 (65% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

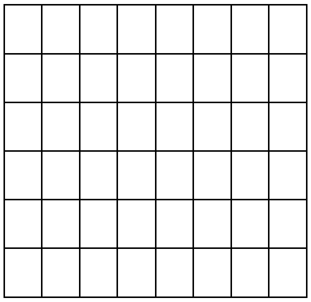
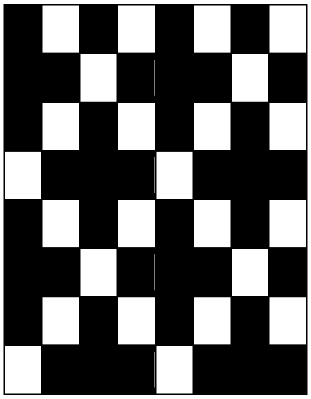
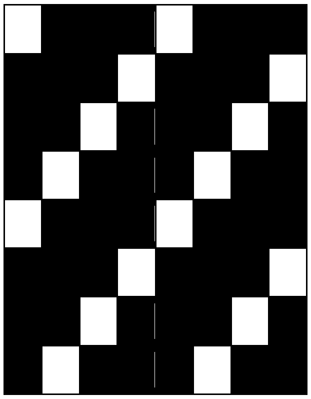
Enumeration Value	Description
pct70 (70% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct75 (75% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct80 (80% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

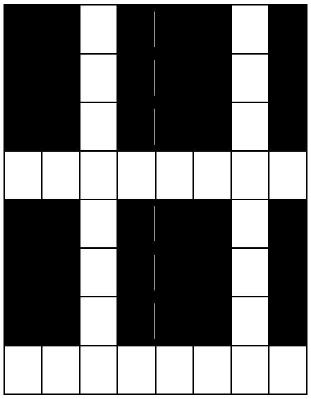
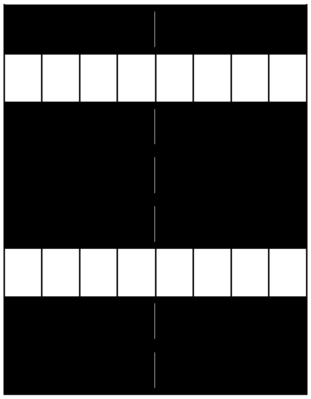
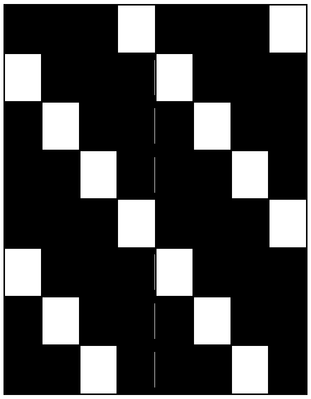
©ISO/IEC 2012 – All rights reserved

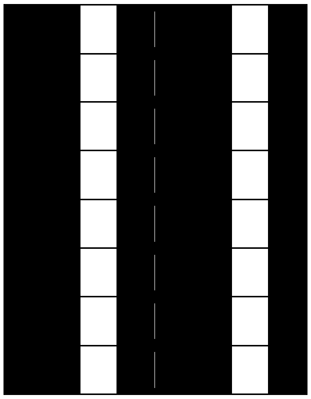
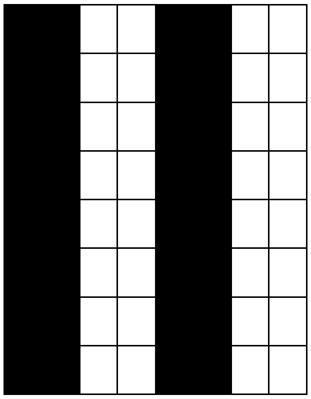
Enumeration Value	Description
	
pct85 (85% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct87 (87.5% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
pct90 (90% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

Enumeration Value	Description
	
pct95 (95% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
reverseDiagStripe (Reverse Diagonal Stripe Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
solid (100% Fill Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	
thinDiagCross (Thin Diagonal Cross Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
thinDiagStripe (Thin Diagonal Stripe Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
thinHorzCross (Thin Horizontal Cross Pattern)	Specifies that the following mask shall be applied as the shading pattern:

Enumeration Value	Description
	
thinHorzStripe (Thin Horizontal Stripe Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
thinReverseDiagStripe (Thin Reverse Diagonal Stripe Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

Enumeration Value	Description
thinVertStripe (Thin Vertical Stripe Pattern)	Specifies that the following mask shall be applied as the shading pattern: 
vertStripe (Vertical Stripe Pattern)	Specifies that the following mask shall be applied as the shading pattern: 

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Shd) is located in §A.1. *end note*]

17.18.79 ST_ShortHexNumber (Four Digit Hexadecimal Value)

This simple type specifies a number value specified as a two octet (four digit) hexadecimal number, whose contents are interpreted based on the context of the parent XML element.

[*Example*: Consider the following value for a node of type ST_ShortHexNumber: 2F6C.

This value is permitted, as it contains two hexadecimal octets, each an encoding of an octet of the actual decimal number value. *end example*]

This simple type's contents are a restriction of the W3C XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a length of exactly 4 hexadecimal digit(s).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_ShortHexNumber) is located in §A.1. *end note*]

17.18.80 ST_SignedHpsMeasure (Signed Measurement in Half-Points)

This simple type specifies that its contents contain either:

- A positive or negative whole number, whose contents consist of a measurement in half-points (equivalent to 1/144th of an inch).
- A positive or negative number followed immediately by a unit identifier.

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of -72 whose type is ST_SignedHpsMeasure. This attribute value specifies a size of negative one-half of an inch or -36 points (-72 halves of a point = -36 points = -0.5 inches). *end example*]

[*Example*: Consider an attribute value of -12.7mm whose type is ST_SignedHpsMeasure. This attribute value specifies a size of -0.0127 meter or negative one-half of an inch or -36 points. *end example*]

This simple type is a union of the following types:

- The ST_UniversalMeasure simple type (§22.9.2.15).
- The W3C XML Schema integer datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SignedHpsMeasure) is located in §A.1. *end note*]

17.18.81 ST_SignedTwipsMeasure (Signed Measurement in Twentieths of a Point)

This simple type specifies that its contents contain either:

- A positive or negative whole number, whose contents consist of a measurement in twentieths of a point (equivalent to 1/1440th of an inch), or
- A positive decimal number immediately following by a unit identifier.

The contents of this measurement are interpreted based on the context of the parent XML element.

[*Example*: Consider an attribute value of 720 whose simple type is ST_SignedTwipsMeasure. This attribute value specifies a size of one-half of an inch or 36 points (720 twentieths of a point = 36 points = 0.5 inches). *end example*]

[*Example*: Consider an attribute value of -12.7mm whose type is ST_SignedTwipsMeasure. This attribute value specifies a size of -0.0127 meter or negative one-half of an inch or -36 points. *end example*]

This simple type is a union of the following types:

- The ST_UniversalMeasure simple type (§22.9.2.15).
- The W3C XML Schema integer datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SignedTwipsMeasure) is located in §A.1. *end note*]

17.18.82 ST_StyleSort (Style Sort Settings)

This simple type specifies the ways in which the list of document styles can be ordered when they are displayed in a user interface.

[*Example*: Consider the WordprocessingML below:

```
<w:stylePaneSortMethod w:val="basedOn" />
```

The stylePaneSortMethod element has a val attribute value of basedOn, which specifies that styles are sorted by the value of their basedOn element (§**17.7.4.3**). *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
basedOn (Sort by Based On Style)	Specifies that styles which are visible should be sorted by the style on which they are based using the basedOn element (§**17.7.4.3**).
default (Sort by Default Method)	Specifies that styles which are visible should be sorted by the default sorting of the host application.
font (Sort by Font)	Specifies that styles which are visible should be sorted by the font which they apply.
name (Sort by Style Name)	Specifies that styles which are visible should be sorted by their names.
priority (Sort by Style Priority)	Specifies that styles which are visible should be sorted by their UI priority using the uiPriority element (§**17.7.4.19**).
type (Sort by Style Type)	Specifies that styles which are visible should be sorted by their style types (i.e. character, linked, paragraph).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_StyleSort) is located in §A.1. *end note*]

17.18.83 ST_StyleType (Style Types)

This simple type specifies the possible values for the types of style definitions defined within a WordprocessingML document. WordprocessingML supports six types of style definitions:

- Paragraph styles
- Character styles
- Table styles
- Numbering styles
- Linked styles (paragraph + character)
- Default paragraph + character properties

Each of the first four style types corresponds to a different value below, and therefore defines the style type of the current style. [*Note*: The last two style types are unique in that they are not simply a style type: a linked style is a pairing of a character and paragraph style via the link element (§17.7.4.6); and the document default properties are defined via the docDefaults element (§17.7.5.1). *end note*]

[*Example*: Consider a style defined as follows:

```
<w:style w:type="paragraph" … >
  <w:name w:val="My Paragraph Style"/>
  <w:rPr>
    <w:b/>
  </w:rPr>
</w:style>
```

The type attribute is of simple type ST_StyleType, and its value of paragraph specifies that this style definition creates a paragraph style. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
character (Character Style)	Specifies that the parent style definition is a character style.
numbering (Numbering Style)	Specifies that the parent style definition is a numbering style.
paragraph (Paragraph Style)	Specifies that the parent style definition is a paragraph style.
table (Table Style)	Specifies that the parent style definition is a table style.

©ISO/IEC 2012 – All rights reserved

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_StyleType) is located in §A.1. *end note*]

17.18.84 ST_TabJc (Custom Tab Stop Type)

This simple type specifies the available types of custom tab stop, which determines the behavior of the tab stop and the alignment which shall be applied to text entered at the current custom tab stop.

[*Example*: Consider a custom tab stops at 1.5" in a WordprocessingML document. This tab stop would be contained within a tab element defining the tab stop as follows:

```
<w:tab w:val="start" w:pos="2160" />
```

The val attribute specifies that this custom tab stop must align all text entered at its location to its left. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
bar (Bar Tab)	Specifies that the current tab is a bar tab. A *bar tab* is a tab which does not result in a custom tab stop in the parent paragraph (this tab stop location shall be skipped when positioning custom tab characters), but instead shall be used to draw a vertical line (or bar) at this location in the parent paragraph.
center (Centered Tab)	Specifies that the current tab stop shall result in a location in the document where all following text is centered (i.e. all text runs following this tab stop and preceding the next tab stop shall be centered around the tab stop location).
clear (No Tab Stop)	Specifies that the current tab stop is cleared and shall be removed and ignored when processing the contents of this document.
decimal (Decimal Tab)	Specifies that the current tab stop shall result in a location in the document where all following text is aligned around the first decimal character in the following text runs. All text runs before the first decimal character shall be before the tab stop, all text runs after it shall be after the tab stop location.
end (Trailing Tab)	Specifies that the current tab stop shall result in a location in the document where all following text is aligned to its trailing edge (i.e. all text runs following this tab stop and preceding the next tab stop shall be

Enumeration Value	Description
	aligned against the trailing edge with respect to the tab stop location). [*Example*: In an RTL paragraph, the trailing edge is the left edge, so text aligns to that edge, extending to the right. *end example*]
num (List Tab)	Specifies that the current tab is a list tab, which is the tab stop between the numbering and the paragraph contents in a numbered paragraph. [*Note*: This justification style is used for backwards compatibility with earlier word processors, and should be avoided in favor of hanging paragraph indentation. *end note*]
start (Leading Tab)	Specifies that the current tab stop shall result in a location in the document where all following text is aligned to its leading edge (i.e. all text runs following this tab stop and preceding the next tab stop shall be aligned against the leading edge with respect to the tab stop location).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TabJc) is located in §A.1. *end note*]

17.18.85 ST_TabTlc (Custom Tab Stop Leader Character)

This simple type specifies the characters which can be used to fill in the space created by a tab which ends at this custom tab stop. The chosen character shall be repeated as required to completely fill the tab spacing generated by the tab character.

[*Example*: Consider a tab stop which should be preceded by a sequence of underscore characters, as follows:

_______________Text at the tab stop

This tab stop would have a leader attribute value of underscore, indicating that the tab stop must be preceded by underscore characters as needed to fill the tab spacing. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
dot (Dotted leader line)	Specifies that the leader character for this custom tab stop shall be a dot. [*Example*:

Enumeration Value	Description
	..................Text at tab stop. *end example*]
heavy (Heavy solid leader line)	Specifies that the leader character for this custom tab stop shall be a heavy solid line, or an underscore. [*Note*: This setting is used for backwards compatibility with earlier word processors, and should be avoided in favor of other leader characters. It can be displayed using underscores if desired. *end note*] [*Example*: __________Text at tab stop. *end example*]
hyphen (Dashed tab stop leader line)	Specifies that the leader character for this custom tab stop shall be a hyphen. [*Example*: --------------- Text at tab stop. *end example*]
middleDot (Middle dot leader line)	Specifies that the leader character for this custom tab stop shall be a centered dot. [*Example*: ··················Text at tab stop. *end example*]
none (No tab stop leader)	Specifies that there shall be no leader character for this custom tab. [*Example*: Text at tab stop. *end example*]
underscore (Solid leader line)	Specifies that the leader character for this custom tab stop shall be an underscore. [*Example*:

Enumeration Value	Description
	__________Text at tab stop. *end example*]

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TabTlc) is located in §A.1. *end note*]

17.18.86 ST_TargetScreenSz (Target Screen Sizes for Generated Web Pages)

This simple type specifies possible ideal minimum target screen sizes (width by height, specified in pixels) for which web pages generated can be optimized when saving this document as a web page.

[*Example*: Consider a WordprocessingML document which contains the following content within the web settings part:

```
<w:webSettings>
  <w:targetScreenSz w:val="1600x1200" />
</w:webSettings>
```

The targetScreenSz element's val attribute has a value of `1600x1200`, which specifies that a target screen size of 1600 by 1200 pixels must be assumed when saving this document as a web page. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
1024x768 (Optimize for 1024x768)	Specifies that web pages produced from this document should be optimized for a screen size of 1024x768.
1152x882 (Optimize for 1152x882)	Specifies that web pages produced from this document should be optimized for a screen size of 1152x882.
1152x900 (Optimize for 1152x900)	Specifies that web pages produced from this document should be optimized for a screen size of 1152x900.
1280x1024 (Optimize for 1280x1024)	Specifies that web pages produced from this document should be optimized for a screen size of 1280x1024.
1600x1200 (Optimize for 1600x1200)	Specifies that web pages produced from this document should be optimized for a screen size of 1600x1200.
1800x1440 (Optimize for 1800x1440)	Specifies that web pages produced from this

Enumeration Value	Description
	document should be optimized for a screen size of 1800x1440.
1920x1200 (Optimize for 1920x1200)	Specifies that web pages produced from this document should be optimized for a screen size of 1920x1200.
544x376 (Optimize for 544x376)	Specifies that web pages produced from this document should be optimized for a screen size of 544x376.
640x480 (Optimize for 640x480)	Specifies that web pages produced from this document should be optimized for a screen size of 640x480.
720x512 (Optimize for 720x512)	Specifies that web pages produced from this document should be optimized for a screen size of 720x512.
800x600 (Optimize for 800x600)	Specifies that web pages produced from this document should be optimized for a screen size of 800x600.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TargetScreenSz) is located in §A.1. *end note*]

17.18.87 ST_TblLayoutType (Table Layout Type)

This simple type defines the possible types of layout algorithms which can be used to lay out a table within a WordprocessingML document.

These algorithms are defined in the following paragraphs (noting, of course, that implementations are free to implement more efficient versions of each).

Fixed Width Table Layout - This method of table layout uses the preferred widths on the table items to generate the final sizing of the table, but does not change that size regardless of the contents of each table cell, hence the table is fixed width.

[*Guidance*: Although an application can choose to use a different process, this layout could be performed as follows:

- The table grid is used to create the set of shared columns in the table and their initial widths as defined in the tblGrid element (§17.4.48)
- The table's total width is defined based on the tblW property (§17.4.63) – if it is set to `auto` or `nil`, then the width is not yet determined and is specified using the row and cell information.
- The first table row is read and the initial number of grid units before the row starts is skipped. The width of the skipped grid columns is set using the wBefore property (§17.4.86).

- The first cell is placed on the grid, and the width of the specified grid column span set by gridSpan (§17.4.17) is set based on the tcW property (§17.4.71).
- Each additional cell is placed on the grid.
- If at any stage, the preferred width requested for the cells exceeds the preferred width of the table, then each grid column is proportionally reduced in size to fit the table width.
- If the grid is exceeded (e.g. tblGrid specifies three grid columns, but the second cell has a gridSpan of three), the grid is dynamically increased with a default width for the new grid column.
- For each subsequent row, cells are placed on the grid, and each grid column is adjusted to be the maximum value of the requested widths (if the widths do not agree) by adding width to the last cell that ends with that grid column. Again, if at any point, the space requested for the cells exceeds the width of the table, then each grid column is proportionally reduced in size to fit the table width.

end guidance]

The resulting table shall be displayed regardless of its contents to the size requested.

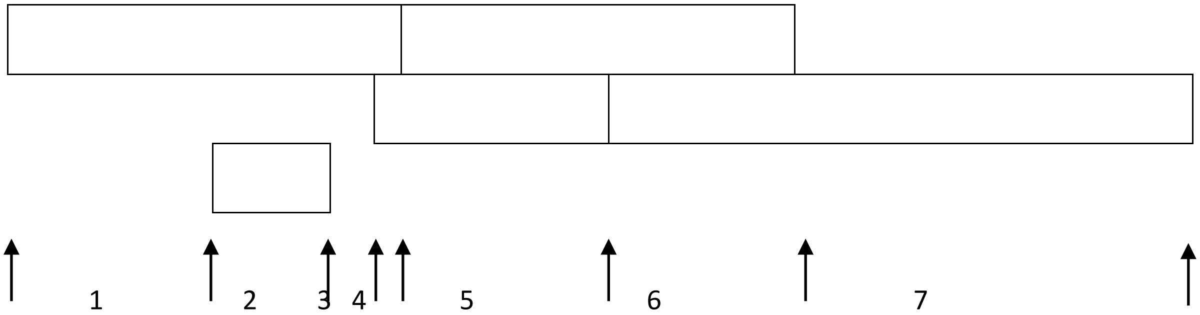
AutoFit Table Layout - This method of table layout uses the preferred widths on the table items to generate the final sizing of the table, but then uses the contents of each cell to determine final column widths.

[*Guidance*: This layout can be performed in any manner available to an application, but one algorithm as follows can be used:

- Perform the steps above to lay out the fixed width version of the table.
- Calculate the minimum content width - the width of the cell's contents including all possible line breaking locations (or the cell's width, if the width of the content is smaller), and the maximum content width -the width of the cell's contents (assuming no line breaking not generated by explicit line breaks).
- The minimum and maximum content width of all cells that span a single grid column is the minimum and maximum content width of that column.
- For cells which span multiple grid columns, enlarge all cells which it spans as needed to meet that cell's minimum width.
- If any cell in a grid column has a preferred width, the first such width overrides the maximum width of the column's contents.
- Place the text in the cells in the table, respecting the minimum content width of each cell's content. If a cell's minimum content width exceeds the cell's current width, preferences are overridden as follows:
- First, override the column widths by making all other grid columns proportionally smaller until each it at its minimum width. This cell can then grow to any width between its own minimum and maximum width.
- Next, override the preferred table width until the table reaches the page width.
- Finally, force a line break in each cell's contents as needed

end guidance]

[*Example*: Consider the following fixed width table, which makes extensive use of resized and merged cells on what is actually just a seven-column grid. (The arrows point to each (invisible) vertical line of the grid and the numbers refer to the grid columns):



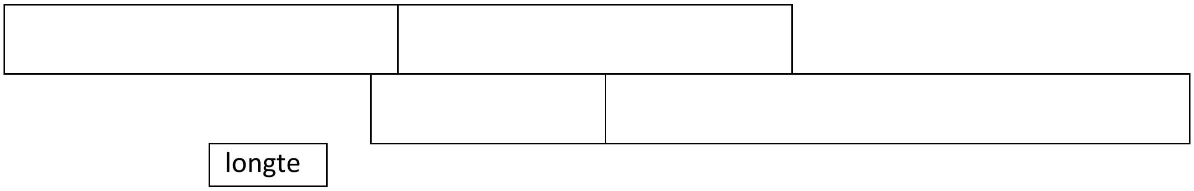
Although the table is visually complex, the standard rules apply: the first cell in the table is simply a cell which spans four grid units horizontally, as specified in the gridSpan element, and whose preferred with is 2952 twentieths of a point, specified in the tcW element:

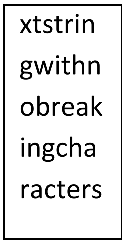
```
<w:tc>
  <w:tcPr>
    <w:tcW w:w="2952" w:type="dxa"/>
    <w:gridSpan w:val="4"/>
  </w:tcPr>
  <w:p/>
</w:tc>
```

Similarly, all cells indented from the stand and end of the grid specify that indent using the gridBefore and gridAfter elements. For example, the XML for the second row in the table shows that that row starts three grid units into the table:

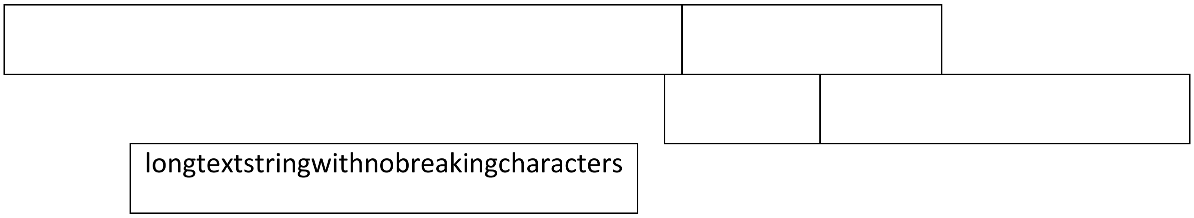
```
<w:tr>
  <w:trPr>
    <w:gridBefore w:val="3"/>
    <w:wBefore w:w="2748" w:type="dxa"/>
  </w:trPr>
  …
</w:tr>
```

If we take this fixed width table and introduce a long string into the single cell in row three, we see that the presence of this text does not affect cell widths:





If we now turn on the AutoFit property, we see that the algorithm for this AutoFit table causes grid column two to increase in size, proportionally decreasing the other grid columns' size to accommodate the long non-breaking string in the last cell:



Each of the other grid columns was reduced, but since all columns are not at their minimum size, the table width is not increased even though the table is not yet at the page width. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

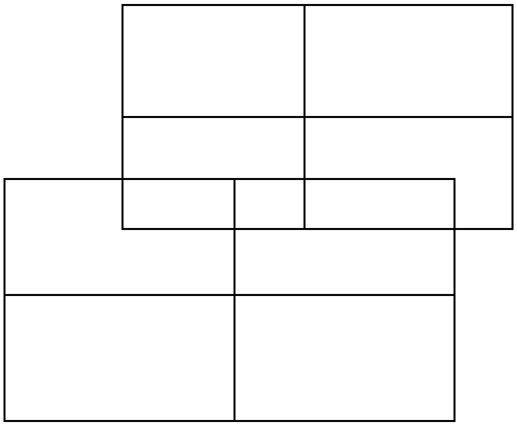
Enumeration Value	Description
autofit (AutoFit Table Layout)	Specifies that this table shall use an AutoFit table layout algorithm.
fixed (Fixed Width Table Layout)	Specifies that this table shall use the fixed width table layout algorithm described above.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TblLayoutType) is located in §A.1. *end note*]

17.18.88 ST_TblOverlap (Table Overlap Setting)

This simple type contains the possible settings for a floating table which shall be used to determine if the table can overlap with other floating tables when displayed in the document.

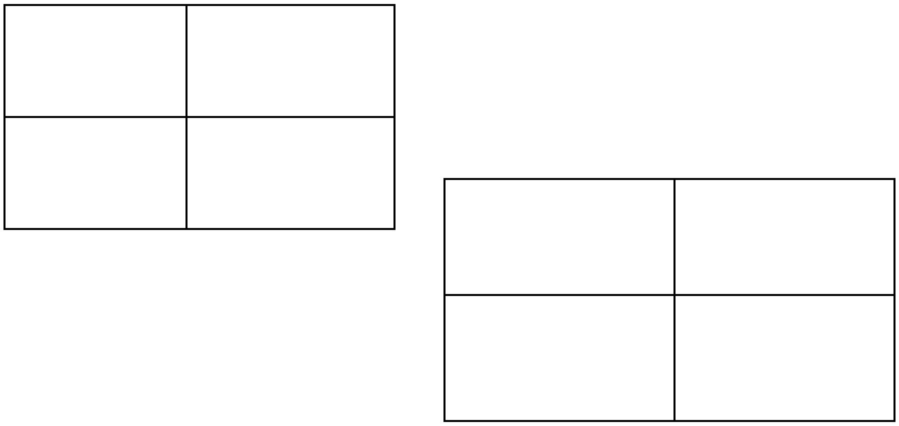
[*Example*: Consider two floating tables in a WordprocessingML document which overlap when displayed, as follows:



If either of these tables specifies that it must not allow overlapping, using the following WordprocessingML:

```
<w:tblPr>
  <w:tblOverlap w:val="never"/>
</w:tblPr>
```

The resulting tables must not overlap, and shall be adjusted at display time to prevent any overlapping, for example:



The value of `never` specifies that the specified table cannot overlap with other floating tables in the document. *end example*]

This simple type's contents are a restriction of the W3C XML Schema `string` datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
never (Floating Table Cannot Overlap)	Specifies that the parent table, if floating, shall never be displayed in a state where it would be overlapping another floating table in the document. If two floating tables intersect and this option is set on either of them, then one or both tables shall be adjusted as needed to ensure that the table whose value is `never` is not overlapped when displayed.
overlap (Floating Table Can Overlap)	Specifies that the parent table, if floating, can be displayed in a state where it would be overlapping another floating table in the document.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TblOverlap) is located in §A.1. *end note*]

17.18.89 ST_TblStyleOverrideType (Conditional Table Style Formatting Types)

This simple type specifies possible values for the sections of the table to which the current conditional formatting properties shall be applied when this table style is used.

[*Example*: Consider a table style which contains conditional formatting, defined as follows:

```
<w:style w:type="table" …>
  …
  <w:tblStylePr w:type="lastRow">
    …
  </w:tblStylePr>
</w:style>
```

The type attribute value of `lastRow` specifies that this set of conditional formatting properties must be applied to the last row of the table only. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
band1Horz (Banded Row Conditional Formatting)	Specifies that the table formatting applies to odd numbered groupings of rows.
band1Vert (Banded Column Conditional Formatting)	Specifies that the table formatting applies to odd numbered groupings of columns.
band2Horz (Even Row Stripe Conditional Formatting)	Specifies that the table formatting applies to even numbered groupings of rows.

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
band2Vert (Even Column Stripe Conditional Formatting)	Specifies that the table formatting applies to even numbered groupings of columns.
firstCol (First Column Conditional Formatting)	Specifies that the table formatting applies to the first column.
firstRow (First Row Conditional Formatting)	Specifies that the table formatting applies to the first row. Any subsequent row which has the tblHeader element present (§17.4.49) shall also use this conditional format.
lastCol (Last table column formatting)	Specifies that the table formatting applies to the last column.
lastRow (Last table row formatting)	Specifies that the table formatting applies to the last row.
neCell (Top right table cell formatting)	Specifies that the table formatting applies to the top right cell.
nwCell (Top left table cell formatting)	Specifies that the table formatting applies to the top left cell.
seCell (Bottom right table cell formatting)	Specifies that the table formatting applies to the bottom right cell.
swCell (Bottom left table cell formatting)	Specifies that the table formatting applies to the bottom left cell.
wholeTable (Whole table formatting)	Specifies that the conditional formatting applies to the whole table.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TblStyleOverrideType) is located in §A.1. *end note*]

17.18.90 ST_TblWidth (Table Width Units)

This simple type specifies the possible values for the units of the width property being defined by a specific table width property. These properties are used to define various properties of a table, including: cell spacing, preferred width, and table margins.

[*Example*: Consider a table with a table cell bottom cell spacing with a type of dxa, as follows:

```
<w:bottom … w:type="dxa" />
```

This type must therefore be used to interpret the width specified in the w attribute as a value in twentieths of a point. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

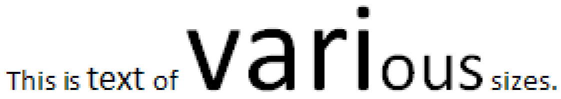
Enumeration Value	Description
auto (Automatically Determined Width)	Specifies that the value for the measurement of the current table width property in the parent table shall be automatically determined by the table layout algorithm when the table is displayed (this width can be adjusted as appropriate). If this value is inappropriate for the current measurement (i.e. this measurement is not affected by that algorithm), then this width type and the associated value can be ignored.
dxa (Width in Twentieths of a Point)	Specifies that the value for the measurement of the current table width property in the parent table shall be interpreted as twentieths of a point (1/1440 of an inch).
nil (No Width)	Specifies that the current width is zero, regardless of any width value specified on the parent element.
pct (Width in Percent of Table Width)	Specifies that the value for the measurement of the current table width property in the parent table shall be interpreted as whole percentage point when a percent sign (U+0025) is present. These percentages shall be calculated relative to the extents specified by the parent XML element. If this value is inappropriate for the current measurement (i.e. this measurement is not part of the width of the table), then this width type and the associated value can be ignored.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TblWidth) is located in §A.1. *end note*]

17.18.91 ST_TextAlignment (Vertical Text Alignment Types)

This simple type specifies the type of vertical alignment which shall be used to align the characters on each line in the parent object.

[*Example*: Consider a paragraph of text of different font sizes, as follows:

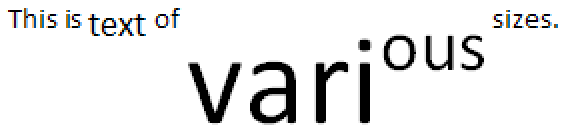


This is text of various sizes.

If the text on this paragraph must be aligned based on the top point of the maximum character height, that requirement would be specified as follows in the WordprocessingML:

```
<w:pPr>
  <w:textAlignment w:val="top" />
</w:pPr>
```

The resulting text would be top aligned, as follows:



This is text of various sizes.

The characters are all aligned to the maximum character extent on the line. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
auto (Automatically Determine Alignment)	Specifies that all text in the parent object shall be aligned automatically when displayed.
baseline (Align Text at Baseline)	Specifies that all text in the parent object shall be aligned to the baseline of each character when displayed.
bottom (Align Text at Bottom)	Specifies that all text in the parent object shall be aligned to the bottom of each character when displayed.
center (Align Text at Center)	Specifies that all text in the parent object shall be aligned to the center of each character when displayed.
top (Align Text at Top)	Specifies that all text in the parent object shall be aligned to the top of each character when displayed.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TextAlignment) is located in §A.1. *end note*]

17.18.92 ST_TextboxTightWrap (Lines To Tight Wrap Within Text Box)

This simple type specifies the lines in the parent paragraph which shall allow the text to be tight wrapped to the paragraph (and not the containing text box) extents when displaying the document.

[*Example*: Consider a paragraph in a text box which meets the criteria specified above which must allow wrapping to the text extents on its first line only. That requirement would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:textboxTightWrap w:val="firstLineOnly" />
</w:pPr>
```

The resulting paragraph would allow text to tightly wrap to the contents of its first line only. All other lines would wrap to the text box's extents. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
allLines (Tight Wrap All Lines)	Specifies that all lines in the paragraph shall allow surrounding text to be tight wrapped to their extents and not the containing text box's extents.
firstAndLastLine (Tight Wrap First and Last Lines)	Specifies that only the first and last lines in the paragraph shall allow surrounding text to be tight wrapped to their extents and not the containing text box's extents.
firstLineOnly (Tight Wrap First Line)	Specifies that only the first line in the paragraph shall allow surrounding text to be tight wrapped to their extents and not the containing text box's extents.
lastLineOnly (Tight Wrap Last Line)	Specifies that only the last line in the paragraph shall allow surrounding text to be tight wrapped to their extents and not the containing text box's extents.
none (Do Not Tight Wrap)	Specifies that no lines in the paragraph shall allow surrounding text to be tight wrapped to their extents and not the containing text box's extents.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TextboxTightWrap) is located in §A.1. *end note*]

17.18.93 ST_TextDirection (Text Flow Direction)

This simple type specifies the direction of the text flow for the parent object.

[*Example*: Consider an object in which text must be oriented vertically, flowing from left to right horizontally on the page. This is achieved by using an `lr` value in an element of type ST_TextDirection. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
lr (Lines Flow From Left to Right)	Specifies that text in the parent object shall be oriented vertically, flowing from left to right

©ISO/IEC 2012 – All rights reserved

Enumeration Value	Description
	horizontally on the page. This means that vertical lines are filled before the text expands horizontally.
lrV (Lines Flow From Left to Right Rotated)	Specifies that text in the parent object shall be oriented vertically, flowing from left to right horizontally on the page. This means that vertical lines are filled before the text expands horizontally. This flow is also rotated clockwise such that text which is not in an East Asian script is rotated 90 degrees when displayed on a page.
rl (Lines Flow From Right to Left)	Specifies that text in the parent object shall be oriented vertically, flowing from right to left horizontally on the page, as if the text were rotated 90 degrees. This means that vertical lines are filled before the text expands horizontally.
rlV (Lines Flow From Right to Left Rotated)	Specifies that text in the parent object shall be oriented vertically, flowing from right to left horizontally on the page. This means that vertical lines are filled before the text expands horizontally. This flow is also rotated such that text which is not in an East Asian script is rotated 90 degrees clockwise when displayed on a page.
tb (Lines Flow From Top to Bottom)	Specifies that text in the parent object shall be oriented horizontally, flowing from top to bottom vertically on the page. The text direction can be set to right-to-left using the bidi element (§17.3.1.6) within individual paragraphs. This means that horizontal lines are filled before the text expands vertically.
tbV (Lines Flow From Top to Bottom Rotated)	Specifies that text in the parent object shall be oriented horizontally, flowing from top to bottom vertically on the page.

Enumeration Value	Description
	This means that horizontal lines are filled before the text expands vertically. This flow is also rotated such that any characters from an East Asian script shall be rotated 270 degrees clockwise when displayed on a page.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TextDirection) is located in §A.1. *end note*]

17.18.94 ST_TextEffect (Animated Text Effects)

This simple type specifies the possible types of animated text effect which can be applied to a text run when it is displayed..

[*Example*: Consider a run of text which must have an animated text effect consisting of multiple colored flashing lights. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:effect w:val="lights"/>
</w:rPr>
```

This run explicitly declares a type of text effect, using the val property, of `lights`, so the contents of this run has the animated lights text effect. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
antsBlack (Black Dashed Line Animation)	Specifies that this text shall be surrounded by an animated black dashed line border.
antsRed (Marching Red Ants)	Specifies that this text shall be surrounded by an animated red dashed line border.
blinkBackground (Blinking Background Animation)	Specifies that this text shall be surrounded by a background color which alternates between black and white.
lights (Colored Lights Animation)	Specifies that this text shall be surrounded by a border consisting of a series of colored lights, which constantly change colors in sequence.
none (No Animation)	Specifies that this text shall have no animated text effect.
shimmer (Shimmer Animation)	Specifies that this text shall be animated by alternating between normal and blurry states.

Enumeration Value	Description
sparkle (Sparkling Lights Animation)	Specifies that this text shall have a background consisting of a random pattern of colored lights, which constantly change colors in sequence.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TextEffect) is located in §A.1. *end note*]

17.18.95 ST_TextScale (Text Expansion/Compression Percentage)

This simple type specifies that the percentage by which the contents of a run shall be expanded or compressed with respect to its normal (100%) character width, with a minimum width of 1% and maximum width of 600%.

[*Example*: Consider a run of text which must be compressed by half when displaying each character within the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:w w:val="50%"/>
</w:rPr>
```

This run explicitly declares that the w value is 50%, so the contents of this run appear at 50% of their normal character width by compressing the width of each character. *end example*]

This simple type is a union of the following types:

- The ST_TextScalePercent simple type (§A.1).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TextScale) is located in §A.1. *end note*]

17.18.96 ST_Theme (Theme Font)

This simple type specifies a theme font type which can be referenced as a theme font within the parent run properties. This theme font is a reference to one of the predefined theme fonts, located in the document's Theme part,which allows for font information to be set centrally in the document.

[*Example*: Consider a run of text which must be displayed using the majorASCII theme font. This requirement would be specified as follows in the resulting WordprocessingML:

```
<w:rPr>
  <w:rFonts w:asciiTheme="majorAscii" />
</w:rPr>
```

The ascii attribute specifies that the run must use the majorAscii theme font as defined in the document's themes part for all text in the appropriate range. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
majorAscii (Major ASCII Theme Font)	Specifies that the current font is a reference to the major theme font for the range of characters from U+0000–U+007F.
majorBidi (Major Complex Script Theme Font)	Specifies that the current font is a reference to the major theme font for the Complex Script range.
majorEastAsia (Major East Asian Theme Font)	Specifies that the current font is a reference to the major theme font for the East Asian range.
majorHAnsi (Major High ANSI Theme Font)	Specifies that the current font is a reference to the major theme font for the High ANSI range.
minorAscii (Minor ASCII Theme Font)	Specifies that the current font is a reference to the minor theme font for the range of characters from U+0000–U+007F.
minorBidi (Minor Complex Script Theme Font)	Specifies that the current font is a reference to the minor theme font for the Complex Script range.
minorEastAsia (Minor East Asian Theme Font)	Specifies that the current font is a reference to the minor theme font for the East Asian range.
minorHAnsi (Minor High ANSI Theme Font)	Specifies that the current font is a reference to the minor theme font for the High ANSI range.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Theme) is located in §A.1. *end note*]

17.18.97 ST_ThemeColor (Theme Color)

This simple type specifies a theme color to be applied to the current object. The specified theme color is a reference to one of the predefined theme colors, located in the document's Theme part, which allows color information to be set centrally in the document.

[*Example*: Consider a set of borders configured to use the accent2 theme color, resulting in the following WordprocessingML markup:

```
<w:top … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />
<w:bottom … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />
<w:left … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />
<w:right … w:color="FFA8A0" w:themeColor="accent2" w:themeTint="99" />
```

The borders have a themeColor attribute of type ST_ThemeColor that when specified, imports the accent2 theme color specified for this document. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
accent1 (Accent 1 Theme Color)	Specifies that the color to be used shall be the theme color specified by the accent1 attribute of the clrSchemeMapping element (§17.15.1.20).
accent2 (Accent 2 Theme Color)	Specifies that the color to be used shall be the theme color specified by the accent2 attribute of the clrSchemeMapping element (§17.15.1.20).
accent3 (Accent 3 Theme Color)	Specifies that the color to be used shall be the theme color specified by the accent3 attribute of the clrSchemeMapping element (§17.15.1.20).
accent4 (Accent 4 Theme Color)	Specifies that the color to be used shall be the theme color specified by the accent4 attribute of the clrSchemeMapping element (§17.15.1.20).
accent5 (Accent 5 Theme Color)	Specifies that the color to be used shall be the theme color specified by the accent5 attribute of the clrSchemeMapping element (§17.15.1.20).
accent6 (Accent 6 Theme Color)	Specifies that the color to be used shall be the theme color specified by the accent6 attribute of the clrSchemeMapping element (§17.15.1.20).
background1 (Background 1 Theme Color)	Specifies that the color to be used shall be the theme color specified by the bg1 attribute of the clrSchemeMapping element (§17.15.1.20).
background2 (Background 2 Theme Color)	Specifies that the color to be used shall be the theme color specified by the bg2 attribute of the clrSchemeMapping element (§17.15.1.20).
dark1 (Dark 1 Theme Color)	Specifies that the color to be used shall be the theme color specified by the t1 attribute of the clrSchemeMapping element (§17.15.1.20).
dark2 (Dark 2 Theme Color)	Specifies that the color to be used shall be the theme color specified by the t2 attribute of the clrSchemeMapping element (§17.15.1.20).
followedHyperlink (Followed Hyperlink Theme Color)	Specifies that the color to be used shall be the theme color specified by the followedHyperlink attribute of the clrSchemeMapping element (§17.15.1.20).
hyperlink (Hyperlink Theme Color)	Specifies that the color to be used shall be the theme color specified by the hyperlink attribute of the clrSchemeMapping element (§17.15.1.20).
light1 (Light 1 Theme Color)	Specifies that the color to be used shall be the theme color specified by the bg1 attribute of the clrSchemeMapping element (§17.15.1.20).

Enumeration Value	Description
light2 (Light 2 Theme Color)	Specifies that the color to be used shall be the theme color specified by the bg2 attribute of the clrSchemeMapping element (§17.15.1.20).
none (No Theme Color)	Specifies that no theme color shall be applied to the current object.
text1 (Text 1 Theme Color)	Specifies that the color to be used shall be the theme color specified by the t1 attribute of the clrSchemeMapping element (§17.15.1.20).
text2 (Text 2 Theme Color)	Specifies that the color to be used shall be the theme color specified by the t2 attribute of the clrSchemeMapping element (§17.15.1.20).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_ThemeColor) is located in §A.1. *end note*]

17.18.98 ST_UcharHexNumber (Two Digit Hexadecimal Value)

This simple type specifies a number value specified as a one octet (two digit) hexadecimal number, whose contents are interpreted based on the context of the parent XML element.

[*Example*: Consider the following value for a node of type ST_UcharHexNumber: BE.

This value is permitted, as it contains two hexadecimal digits, as an encoding of an octet of the actual decimal number value. *end example*]

This simple type's contents are a restriction of the W3C XML Schema hexBinary datatype.

This simple type also specifies the following restrictions:

- This simple type's contents have a length of exactly 2 hexadecimal digit(s).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_UcharHexNumber) is located in §A.1. *end note*]

17.18.99 ST_Underline (Underline Patterns)

This simple type specifies the types of patterns which can be used to create the underline applied beneath the text in a run.

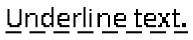
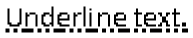
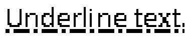
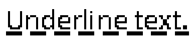
[*Example*: Consider a run of text which must have a double underline explicitly turned on for the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:u w:val="double"/>
</w:rPr>
```

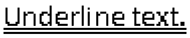
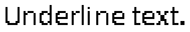
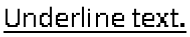
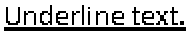
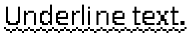
The val of the underline on this run is `double`, so the style of the underline on this run must be a double line. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
dash (Dashed Underline)	Specifies an underline consisting of a dashed line beneath all characters in this run. [*Example*:  Underline text. *end example*]
dashDotDotHeavy (Thick Dash-Dot-Dot Underline)	Specifies an underline consisting of a series of thick dash, dot, dot characters beneath all characters in this run. [*Example*:  Underline text. *end example*]
dashDotHeavy (Thick Dash-Dot Underline)	Specifies an underline consisting of a series of thick dash, dot characters beneath all characters in this run. [*Example*:  Underline text. *end example*]
dashedHeavy (Thick Dashed Underline)	Specifies an underline consisting of a series of thick dashes beneath all characters in this run. [*Example*:  Underline text. *end example*]
dashLong (Long Dashed Underline)	Specifies an underline consisting of long dashed characters beneath all characters in this run. [*Example*:

Enumeration Value	Description
	Underline text. *end example*]
dashLongHeavy (Thick Long Dashed Underline)	Specifies an underline consisting of thick long dashed characters beneath all characters in this run. [*Example*: Underline text. *end example*]
dotDash (Dash-Dot Underline)	Specifies an underline consisting of a series of dash, dot characters beneath all characters in this run. [*Example*: Underline text. *end example*]
dotDotDash (Dash-Dot-Dot Underline)	Specifies an underline consisting of a series of dash, dot, dot characters beneath all characters in this run. [*Example*: Underline text. *end example*]
dotted (Dotted Underline)	Specifies an underline consisting of a series of dot characters beneath all characters in this run. [*Example*: Underline text. *end example*]
dottedHeavy (Thick Dotted Underline)	Specifies an underline consisting of a series of thick dot characters beneath all characters in this run. [*Example*: Underline text.

Enumeration Value	Description
	end example]
double (Double Underline)	Specifies an underline consisting of two lines beneath all characters in this run. [*Example*: Underline text. *end example*]
none (No Underline)	Specifies no underline beneath this run. [*Example*: Underline text. *end example*]
single (Single Underline)	Specifies an underline consisting of a single line beneath all characters in this run. [*Example*: Underline text. *end example*]
thick (Thick Underline)	Specifies an underline consisting of a single thick line beneath all characters in this run. [*Example*: Underline text. *end example*]
wave (Wave Underline)	Specifies an underline consisting of a single wavy line beneath all characters in this run. [*Example*: Underline text. *end example*]
wavyDouble (Double Wave Underline)	Specifies an underline consisting of a pair of wavy lines beneath all characters in this run.

Enumeration Value	Description
	[*Example*: Underline text. *end example*]
wavyHeavy (Heavy Wave Underline)	Specifies an underline consisting of a single thick wavy line beneath all characters in this run. [*Example*: Underline text. *end example*]
words (Underline Non-Space Characters Only)	Specifies an underline consisting of a single line beneath all non-space characters in the run. There shall be no underline beneath any space character (breaking or non-breaking). [*Example*: Underline text. *end example*]

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Underline) is located in §A.1. *end note*]

17.18.100 ST_VAnchor (Vertical Anchor Location)

This simple type specifies the vertical position to which the parent object has been anchored in the document. This anchor position shall be used as the base location to determine the final vertical position of the object in the document.

[*Example*: Consider a text frame which should be positioned one inch to the right of its column in a left-to-right document. This text frame would be specified using the following WordprocessingML:

```
<w:pPr>
  <w:framePr … w:y="1440" w:vAnchor="page" />
</w:pPr>
```

These frame vertical anchor properties specify that they are relative to the anchor paragraph's page. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
margin (Relative To Margin)	Specifies that the parent object shall be vertically anchored to the text margins. This shall be used to specify that any vertical positioning values shall be calculated with respect to the location of the text margin.
page (Relative To Page)	Specifies that the parent object shall be vertically anchored to the page edge. This shall be used to specify that any vertical positioning values shall be calculated with respect to the location of the edge of the page.
text (Relative To Vertical Text Extents)	Specifies that the parent object shall be vertically anchored to the text extents. This shall be used to specify that any vertical positioning values shall be calculated with respect to the location of the top edge of the text in the anchor paragraph.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_VAnchor) is located in §A.1. *end note*]

17.18.101 ST_VerticalJc (Vertical Alignment Type)

This simple type specifies the vertical alignment for text between the top and bottom margins of the parent container (page or table cell).

[*Example*: Consider a region where the text must be vertically centered in the parent element. This would require a val value of `center`, in order to specify that all justification vertically must be centered relative to the parent. For a section, this setting would be specified as follows:

```
<w:vAlign w:val="center" />
```

The val attribute of `center` specifies that the content is centered relative to its container . *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
both (Vertical Justification)	Specifies that the text shall be vertically justified

Enumeration Value	Description
	between the top and bottom margins of the parent object, by adding additional line spacing to each paragraph as required. This setting is only applied for the content of the section which is displayed on full pages. If the content does not use the full page (e.g. another section begins on the same page, or the document ends mid-page), then the value shall be ignored when rendering that page (returning to the default value of top) This value is only permitted for page justification settings, and shall be ignored when specified on a table cell (returning to the default value of top).
bottom (Align Bottom)	Specifies that the text shall be vertically aligned to the bottom margin of the parent object, by moving all text to the bottom text extent within the parent object as required.
center (Align Center)	Specifies that the text shall be vertically aligned to the center of the parent object..
top (Align Top)	Specifies that the text shall be vertically aligned to the top margin of the parent object, by moving all text to the top text extent within the parent object as required.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_VerticalJc) is located in §A.1. *end note*]

17.18.102 ST_View (Document View Values)

This simple type defines the possible views which can be used to determine how WordprocessingML documents can be rendered when displayed by an application.

[*Example*: Consider a WordprocessingML document that must be displayed on the screen in the same form as it is printed. This requirement would be specified using the following WordprocessingML in the document settings part:

```
<w:view w:val="print" />
```

The view element's val attribute is equal to print specifying that the given WordprocessingML document must be rendered as it is printed. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

©ISO/IEC 2012 – All rights reserved

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
masterPages (Master Document View)	Specifies that a given WordprocessingML document shall be rendered in a view optimized for outlining or creating long documents. [*Note*: This setting can be interpreted as functionally equivalent to the `outline` setting, as it only remains separate to support legacy applications. *end note*]
none (Default View)	Specifies that a given WordprocessingML document shall be rendered in the default view of the application.
normal (Draft View)	Specifies that a given WordprocessingML document shall be rendered in a view optimized for outlining or creating long documents.
outline (Outline View)	Specifies that a given WordprocessingML document shall be rendered in a view optimized for outlining or creating long documents.
print (Print Layout View)	Specifies that this document shall be opened in a view that displays the document as it prints.
web (Web Page View)	Specifies that a given WordprocessingML document shall be rendered in a view mimicking the way this document would be displayed in a web page.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_View) is located in §A.1. *end note*]

17.18.103 ST_WmlColorSchemeIndex (Theme Color Reference)

This simple type specifies the possible set of theme color stored in the document's Theme part which can be referenced by document content. This reference is used to map the use of the theme colors in the ST_ThemeColor enumeration to the theme colors in the theme part.

[*Example*: Consider a WordprocessingML document that must have references to the theme color `accent1` mapped to the theme color lt1 as defined in the document's theme part. This requirement would be specified using the following WordprocessingML in the document settings:

```
<w:clrSchemeMapping w:accent1="light1" />
```

The accent1 attribute has a value of `light1`, specifying that uses of the theme color value `accent1` must be mapped to the theme color lt1. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
accent1 (Accent 1 Theme Color Reference)	Specifies a reference to the accent1 theme color in the document's Theme part.
accent2 (Accent 2 Theme Color Reference)	Specifies a reference to the accent2 theme color in the document's Theme part.
accent3 (Accent 3 Theme Color Reference)	Specifies a reference to the accent3 theme color in the document's Theme part.
accent4 (Accent4 Theme Color Reference)	Specifies a reference to the accent4 theme color in the document's Theme part.
accent5 (Accent5 Theme Color Reference)	Specifies a reference to the accent5 theme color in the document's Theme part.
accent6 (Accent 6 Theme Color Reference)	Specifies a reference to the accent6 theme color in the document's Theme part.
dark1 (Dark 1 Theme Color Reference)	Specifies a reference to the dk1 theme color in the document's Theme part.
dark2 (Dark 2 Theme Color Reference)	Specifies a reference to the dk2 theme color in the document's Theme part.
followedHyperlink (Followed Hyperlink Theme Color Reference)	Specifies a reference to the folHlink theme color in the document's Theme part.
hyperlink (Hyperlink Theme Color Reference)	Specifies a reference to the hlink theme color in the document's Theme part.
light1 (Light 1 Theme Color Reference)	Specifies a reference to the lt1 theme color in the document's Theme part.
light2 (Light 2 Theme Color Reference)	Specifies a reference to the lt2 theme color in the document's Theme part.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_WmlColorSchemeIndex) is located in §A.1. *end note*]

17.18.104 ST_Wrap (Text Wrapping around Text Frame Type)

This simple type specifies the type of text wrapping which shall be allowed around a text frame within a document.

[*Example*: Consider the following WordprocessingML fragment specifying a text frame:

```
<w:p>
  <w:pPr>
    <w:framePr w:w="2419" w:h="2189" w:hRule="atLeast" w:hSpace="187"
w:wrap="around" w:vAnchor="text" w:hAnchor="page" w:x="1643" w:y="73" />
  </w:pPr>
```

```
    <w:r>
      <w:t>Text Frame Content.</w:t>
    </w:r>
  </w:p>
```

This wrap attribute on this text frame specifies that when the frame is rendered on the page, any non-text frame paragraphs which would normally flow onto the same lines must be allowed to wrap around it. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
around (Allow Text Wrapping Around Frame)	Specifies that text shall be allowed to wrap around the remaining space on each line around this text frame in the document.
auto (Default Text Wrapping Around Frame)	Specifies that text shall have the default application-defined behavior of the application displaying the WordprocessingML document with regard to the text wrapping displayed around the frame.
none (No Text Wrapping Around Frame)	Specifies that text shall not be allowed to wrap around the remaining space on each lines around this text frame. Any text content shall therefore be placed on the next line following this text frame which does not intersect with the frame's extents.
notBeside (No Text Wrapping Beside Frame)	Specifies that text shall not be allowed to wrap around the remaining space on each lines around this text frame. Any text content shall therefore be placed on the next line following this text frame which does not intersect with the frame's extents.
through (Through Text Wrapping Around Frame)	Specifies that text shall be allowed to wrap around the remaining space on each line around this text frame in the document.
tight (Tight Text Wrapping Around Frame)	Specifies that text shall be allowed to tightly wrap around the remaining space on each line around this text frame in the document.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Wrap) is located in §A.1. *end note*]

17.18.105 ST_Zoom (Magnification Preset Values)

This simple type specifies the type of magnification settings which can be applied to a given document on open.

[*Example*: Consider a WordprocessingML document that should be visible without any horizontal scrolling when it is displayed. This requirement would be specified using the following WordprocessingML:

```
<w:zoom w:val="bestFit" w:percent="90%" />
```

The val attribute is equal to the value `bestFit` specifying that an application must dynamically calculate the magnification needed such that the given document must be visible on the horizontal plane of the document with no horizontal scrolling required to see any part of the WordprocessingML document's pages. *end example*]

This simple type's contents are a restriction of the W3C XML Schema string datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
bestFit (Display Page Width)	Specifies that the magnification setting shall be adjusted to ensure the width of the current page matches the available window width.
fullPage (Display One Full Page)	Specifies that the magnification setting shall be adjusted to ensure that one full page can be seen at a time.
none (No Preset Magnification)	Specifies that no preset magnification is present, and the last known cached setting shall be used.
textFit (Display Text Width)	Specifies that the magnification setting shall be adjusted to ensure the width of the text extents on the current page matches the available window width.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Zoom) is located in §A.1. *end note*]

17.18.106 ST_TextScalePercent (Text Expansion/Compression Percentage)

This simple type specifies that the percentage by which the contents of a run shall be expanded or compressed with respect to its normal (100%) character width, with a minimum width of 1% and maximum width of 600%.

[*Example*: Consider a run of text that must be compressed by half when displaying each character within the contents of the run. This constraint is specified using the following WordprocessingML:

```
<w:rPr>
  <w:w w:val="50%"/>
</w:rPr>
```

This run explicitly declares that the w value is 50%, so the contents of this run appear at 50% of their normal character width by compressing the width of each character. *end example*]

This simple type also specifies the following restrictions:

This simple type's contents shall match the following regular expression pattern:
0*(600|([0-5]?[0-9]?[0-9]))%.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TextScalePercent) is located in §A.1. *end note*]

17.18.107 ST_MeasurementOrPercent (Measurement or Percentage Value)

This simple type specifies the possible values for a table measurement, which can be percentage-based or absolute. See the union's member types for details.

This simple type is a union of the following types:

The ST_DecimalNumberOrPercent simple type (§17.18.11).
The ST_UniversalMeasure simple type (§22.9.2.15).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_MeasurementOrPercent) is located in §A.1. *end note*]