18. SpreadsheetML 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 SpreadsheetML document, as defined by §12 of this Part of ISO/IEC 29500.

18.1 Table of Contents

This subclause is informative.

End of informative text.

18.2 Workbook

A workbook is composed of workbook-level properties and a collection of 1 or more sheets. The sheets are the central structure within a workbook, and can contain cells, which, in turn, can contain the text, numbers, dates, formulas, and other constructs of a workbook. The workbook part and corresponding properties comprise data used to set application and workbook-level operational state. The workbook also serves to bind all the sheets and child elements into an organized single file. The workbook XML attributes and elements include information about what application last saved the file, where and how the windows of the workbook were positioned, and an enumeration of the worksheets in the workbook.

It is important for the sake of simplicity to minimize the *required* set of workbook XML attributes and elements that shall be present to compose a valid SpreadsheetML workbook. Therefore this is the XML for the smallest possible (blank) workbook:

```
<workbook>
```

```
  <sheets>
    <sheet name="Sheet1" sheetId="1" r:id="rId1"/>
  </sheets>
</workbook>
```

Note that this workbook has a single sheet, named Sheet1. An Id for the sheet is required, and a relationship Id pointing to the location of the sheet definition is also required.

18.2.1 bookViews (Workbook Views)

This element specifies the collection of workbook views of the enclosing workbook. Each view can specify a window position, filter options, and other configurations. There is no limit on the number of workbook views that can be defined for a workbook.

[*Example*:

```
<bookViews>
  <workbookView showHorizontalScroll="0" showVerticalScroll="0"
    showSheetTabs="0" xWindow="120" yWindow="45" windowWidth="15135"
    windowHeight="8130" activeTab="2" autoFilterDateGrouping="0"/>
</bookViews>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_BookViews) is located in §A.2. *end note*]

18.2.2 calcPr (Calculation Properties)

This element defines the collection of properties the application uses to record calculation status and details. Calculation is the process of computing formulas and then displaying the results as values in the cells that contain the formulas.

[*Example*:

```
<calcPr calcId="122211" calcMode="auto" refMode="R1C1" iterate="1"
  fullPrecision="0"/>
```

end example]

Attributes	Description
calcCompleted (Calc Completed)	Specifies a boolean value that determines whether workbook data was recalculated before the workbook was saved. A value of `1` or `true` indicates recalculation was completed before save. A value of `0` or `false` indicates that recalculation was not completed before save.

Attributes	Description
	The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
calcId (Calculation Id)	Specifies the version of the calculation engine used to calculate values in the workbook. When you open a workbook created in the current version, the application recalculates only the formulas that depend on cells that have changed. When you open a workbook that was created in an earlier version of the application, all the formulas in the workbook— those that depend on cells that have changed and those that do not— are recalculated. This ensures that the workbook is fully optimized for the current application version. The value for `calcID` depends on the application. SpreadsheetML defaults form [version][build], where [version] refers to the version of the application, and [build] refers to the build of the application when the calculation engine changed. [*Example*: `<calcPr calcId="122211"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
calcMode (Calculation Mode)	Specifies when the application should calculate formulas in the workbook. The default value for this attribute is "auto." The possible values for this attribute are defined by the ST_CalcMode simple type (§18.18.4).
calcOnSave (Calculate On Save)	Specifies a boolean value that indicates whether the application recalculates values when the workbook is saved. A value of `1` or `true` indicates recalculation is performed when the workbook is saved. A value of `0` or `false` indicates recalculation is not performed when the workbook is saved. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
concurrentCalc (Concurrent	Specifies a boolean value that indicates whether concurrent calculation processes are enabled for this workbook.

Attributes	Description
Calculations)	A value of on, 1, or true indicates concurrent calculations are enabled in this workbook. A value of 0 or false indicates concurrent calculations are not enabled. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
concurrentManualCount (Concurrent Thread Manual Count)	Specifies the count of concurrent calculation processes manually set by the user. If omitted, the count is set automatically by the application. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
forceFullCalc (Force Full Calculation)	Specifies a boolean value that indicates whether the application performs a full recalculation when one was not indicated by other calculation properties. This attribute allows the application to expose mechanisms in the user interface that give users the ability to trigger when full recalculations take place. A value of 1 or true indicates the application performs a full recalculation of the workbook. A value of 0 or false indicates the application does not perform a full recalculation of the workbook. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
fullCalcOnLoad (Full Calculation On Load)	Specifies a boolean value that indicates whether the application shall perform a full recalculation when the workbook is opened. After load and successful calculation, the application should set this value to false. The application should set this value to true when cell formulas or values are modified by another process while the application has the workbook opened. A value of 1 or true indicates the application performs a full recalculation of workbook values when the workbook is opened. A value of 0 or false indicates the application does not perform a full recalculation when the workbook is opened. [*Note*: If manual calcMode is true, then a full recalculation does not performed on load, even when this attribute is set. *end note*] The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
	datatype.
fullPrecision (Full Precision Calculation)	Specifies a boolean that indicates the precision the application uses when performing calculations in the workbook. Full precision means that the application uses the entire value(s) stored in cells referenced by the formula to perform the calculation. [*Example*: If two cells each contain the value 10.005 and the cells are formatted to display values in currency format, the value $10.01 is displayed in each cell. If you add the two cells together, the result is $20.01 because the application adds the stored values 10.005 and 10.005, not the displayed values. You can change the precision of calculations so that the application uses the displayed value instead of the stored value when it recalculates formulas. For the above example, if fullPrecision is `false`, then the result must be $20.02, because each cell shows $10.01, so those are the values to be added. Furthermore, when fullPrecision is `false`, the calculated value as displayed must be saved to file. *end example*] A value of `1` or `true` indicates the application uses the stored values of the referenced cells when performing calculations. A value of `0` or `false` indicates the application uses the display values of the referenced cells when performing calculations. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
iterate (Calculation Iteration)	Specifies a boolean value that indicates whether the application should attempt to calculate formulas that contain circular references. A circular reference is a formula that refers to the cell— either directly or indirectly— that contains the formula. If a formula refers back to one of its own cells, you shall determine how many times the formula should recalculate. A value of on, 1, or true indicates the application should attempt to calculate circular references. The calculation engine performs iterative `iterateCount` calculations to before stopping. A value of `0` or `false` indicates that the application should not attempt to calculate formulas with circular references. The calculation engine stops on the first iteration when it encounters a circular references. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
iterateCount (Iteration Count)	Specifies the number of iterations the calculation engine attempts when calculating a workbook with circular references, when `iterate` is true.

Attributes	Description
	The default value for this attribute is 100. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
iterateDelta (Iterative Calculation Delta)	Specifies a double that contains the maximum change for iterative calculations. The application stops calculating after iterateCount iterations or after all values in the circular reference change by less than iterateDelta between iterations, whichever comes first. The default value for this attribute is "0.001" The possible values for this attribute are defined by the W3C XML Schema double datatype.
refMode (Reference Mode)	Specifies the reference style for this workbook. Instead of using letters for columns and numbers for rows ("A1"), this option enables using numbers for both rows and columns. Cells are then referred to in this format: R1C1. The default value for this attribute is "A1." The possible values for this attribute are defined by the ST_RefMode simple type (§18.18.64).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CalcPr) is located in §A.2. *end note*]

18.2.3 customWorkbookView (Custom Workbook View)

This element specifies a single custom workbook view. A custom workbook view consists of a set of display and print settings that you can name and apply to a workbook. You can create more than one custom workbook view of the same workbook. Custom Workbook Views are not required in order to construct a valid SpreadsheetML document, and are not necessary if the document is never displayed by a spreadsheet application, or if the spreadsheet application has a fixed display for workbooks. However, if a spreadsheet application chooses to implement configurable display modes, the customWorkbookView element should be used to persist the settings for those display modes. The settings fall into the following categories, and individual settings are detailed in the table following:

Window settings - these include size and positioning of the spreadsheet window as well as which window features should be displayed (scrollbars, sheet tabs, etc.).

Workbook content display and print settings - specifies whether features in the workbook should be included or ignored by a spreadsheet application when a custom workbook view is displayed or printed. [*Example*: Whether comments in the workbook should be displayed and how objects such as images should be displayed can be controlled by a Custom Workbook View. *end example*]

Persistence settings - these include settings that describe how a spreadsheet application should update persisted SpreadsheetML content if multiple spreadsheet applications are accessing a common instance of the SpreadsheetML document at the same time.

When a Custom Workbook View is present, there should also be corresponding customSheetView (§18.3.1.25) elements for each sheet (§18.2.19) in the workbook. The guid attribute of these customSheetView elements associates the customSheetView with the appropriate customWorkbookView. Attributes on the customWorkbookView element should be used to determine which settings within the Custom Sheet View should be respected by a spreadsheet application if the Custom Workbook View is displayed.

Attributes	Description
activeSheetId (Active Sheet in Book View)	Specifies the sheetId of a sheet in the workbook that identifies to a consuming application the default sheet to display. Corresponds to a sheetId of a sheet in the sheets collection. This attribute is required. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
autoUpdate (Auto Update)	Specifies a boolean value that is an instruction that if the workbook is loaded by a spreadsheet application, that spreadsheet application should automatically update changes at the interval specified by the `mergeInterval` attribute. This is only applicable for shared workbooks (§18.11). A value of `1` or `true` is an instruction to the spreadsheet application to update changes at the interval specified in the `mergeInterval` attribute. A value of `0` or `false` is an instruction to the spreadsheet applicationto update changes whenever the spreadsheet application generates SpreadsheetML representing the workbook. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
changesSavedWin (Changes Saved Win)	Specifies a boolean value that instructs a spreadsheet application to overwrite the persisted version of the document with the updated version being persisted. This is only applicable for shared workbooks in automatic update mode. A value of `1` or `true` instructs a spreadsheet application to overwrite changes in the persisted version of a shared workbook when conflicts in data are found. A value of `0` or `false` instructs a spreadsheet application to not overwrite changes in the persisted version of a shared workbook when conflicts are found. The default value for this attribute is false.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
guid (Custom View GUID)	Specifies a globally unique identifier (GUID) for this custom view The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).
includeHiddenRowCol (Include Hidden Rows & Columns)	Specifies a boolean value that indicates whether to include hidden rows, columns, and filter settings in this custom view. A value of `1` or `true` indicates that hidden rows, columns, and filter settings are included in this custom view. A value of `0` or `false` indicates that hidden rows, columns, and filter settings are not included. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
includePrintSettings (Include Print Settings)	Specifies a boolean value that indicates whether to include print settings in this custom view. A value of `1` or `true` indicates that print settings are included in this custom view. A value of `0` or `false` indicates print settings are not included in this custom view. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
maximized (Maximized)	Specifies a boolean value that indicates whether the workbook window is maximized. A value of `1` or `true` indicates the workbook window is maximized. A value of `0` or `false` indicates the workbook window is not maximized. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
mergeInterval (Merge Interval)	Automatic update interval (in minutes). Only applicable for shared workbooks in automatic update mode. The possible values for this attribute are defined by the W3C XML Schema unsignedInt

Attributes	Description
	datatype.
minimized (Minimized)	Specifies a boolean value that indicates whether the workbook window is minimized. A value of 1 or true indicates the workbook window is minimized. A value of 0 or false indicates the workbook window is not minimized. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
name (Custom View Name)	Specifies the name of the custom view. This attribute is required. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
onlySync (Only Synch)	Specifies a boolean value that indicates, during automatic update, the current user's changes are not saved. The workbook is only updated with other users' changes. Only applicable for shared workbooks in automatic update mode. A value of 1 or true indicates the current user's changes is not saved during automatic update. A value of 0 or false indicates the current user's is saved during automatic update. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
personalView (Personal View)	Specifies a boolean value that indicates that this custom view is a personal view for a shared workbook user. Only applicable for shared workbooks. Personal views allow each user of a shared workbook to store their individual print and filter settings. A value of 1 or true indicates this custom view is a personal view for a shared workbook user. A value of 0 or false indicates this view is not a personal view. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showComments	Specifies how comments are displayed in this custom view

Attributes	Description
(Show Comments)	The possible values for this attribute are defined by the ST_Comments simple type (§18.18.14).
showFormulaBar (Show Formula Bar)	Specifies a boolean value that indicates whether to display the formula bar in the application user interface. A value of `1` or `true` indicates the formula bar is shown in the user interface. A value of `0` or `false` indicates the formula bar is not shown in the user interface. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showHorizontalScroll (Show Horizontal Scroll)	Specifies a boolean value that indicates whether to display the horizontal scroll bar in the user interface. A value of `1` or `true` indicates that the horizontal scrollbar is shown. A value of `0` or `false` indicates that the horizontal scrollbar is not shown. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showObjects (Show Objects)	Specifies how objects are displayed in this custom view. The default value for this attribute is "all." The possible values for this attribute are defined by the ST_Objects simple type (§18.18.48).
showSheetTabs (Show Sheet Tabs)	Specifies a boolean value that indicates whether to display the sheet tabs in the user interface. A value of `1` or `true` indicates that sheet tabs shall be shown. A value of `0` or `false` indicates that sheet tabs shall not be shown. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showStatusbar (Show Status Bar)	Specifies a boolean value that indicates whether to display the status bar in the user interface.

Attributes	Description
	A value of 1 or true indicates that the status bar is shown. A value of 0 or false indicates the status bar is not shown. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showVerticalScroll (Show Vertical Scroll)	Specifies a boolean value that indicates whether to display the vertical scroll bar. A value of 1 or true indicates the vertical scrollbar shall be shown. A value of 0 or false indicates the vertical scrollbar shall not be shown. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
tabRatio (Sheet Tab Ratio)	Specifies the ratio between the workbook tabs bar and the horizontal scroll bar. tabRatio is assumed to be out of 1000 of the horizontal window width. The default value for this attribute is 600. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
windowHeight (Window Height)	Specifies the height of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
windowWidth (Window Width)	Specifies the width of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
xWindow (Top Left Corner (X Coordinate))	Specifies the X coordinate for the upper left corner of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the W3C XML Schema int datatype.
yWindow (Top Left Corner (Y Coordinate))	Specifies the Y coordinate for the upper left corner of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomWorkbookView) is located in §A.2. *end note*]

18.2.4 customWorkbookViews (Custom Workbook Views)

This element defines the collection of custom workbook views that are defined for this workbook. A customWorkbookView is similar in concept to a workbookView (§18.2.30) in that its attributes contain settings related to the way that the workbook should be displayed on a screen by a spreadsheet application.

[*Note*: Whilst a workbookView is intended to store the way the workbook window should be displayed by a spreadsheet application, customWorkbookView elements are intended to allow the user to switch between a selection of customWorkbookView items defining window and content display options. *end note*]

A customWorkbookView contains a greater number of settings (e.g., the presence of a formula bar; visibility of hidden data; whether or not to show comments) and is named.

[*Example*: A workbook which is used by two different departments might contain two customWorkbookView elements – one where the comments and hidden data are not shown, and one where they are. Users might switch between the customWorkbookView items according to the department in which they work. *end example*]

There is no limit on the number of custom views that can be contained within a SpreadsheetML instance.

[*Example*:

```
<customWorkbookViews>
  <customWorkbookView name="CustomView"
    guid="{CE6681F1-E999-414D-8446-68A031534B57}" maximized="1" xWindow="1"
    yWindow="1" windowWidth="1024" windowHeight="547" activeSheetId="1"/>
</customWorkbookViews>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomWorkbookViews) is located in §A.2. *end note*]

18.2.5 definedName (Defined Name)

This element defines the defined names that are defined within this workbook. Defined names are descriptive text that is used to represents a cell, range of cells, formula, or constant value. Use easy-to-understand names, such as Products, to refer to hard to understand ranges, such as Sales!C20:C30.

A defined name in a formula can make it easier to understand the purpose of the formula. [*Example*: The formula =SUM(FirstQuarterSales) might be easier to identify than =SUM(C20:C30). *end example*]

Names are available to any sheet. [*Example*: Ifthe name ProjectedSales refers to the range A20:A30 on the first worksheet in a workbook, you can use the name ProjectedSales on any other sheet in the same workbook to refer to range A20:A30 on the first worksheet. *end example*]

Names can also be used to represent formulas or values that do not change (constants). [*Example*: The name SalesTax can be used to represent the sales tax amount (such as 6.2 percent) applied to sales transactions. *end example*]

You can also link to a defined name in another workbook, or define a name that refers to cells in another workbook. [*Example*: The formula =SUM(Sales.xls!ProjectedSales) refers to the named range ProjectedSales in the workbook named Sales. *end example*]

A compliant producer or consumer considers a defined name in the range A1-XFD1048576 to be an error.

All other names outside this range can be defined as names and overrides a cell reference if an ambiguity exists.

[*Example*: For clarification: LOG10 is always a cell reference, LOG10() is always formula, LOGO1000 can be a defined name that overrides a cell reference. *end example*]

Attributes	Description
comment (Comment)	Specifies the comment the user provided when the name was created. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
customMenu (Custom Menu Text)	Specifies custom menu text for the defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
description (Description)	Specifies description text for the defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
function (Function)	Specifies a boolean value that indicates that the defined name refers to a user-defined function. This attribute is used when there is an add-in or other code project associated with the file. A value of `1` or `true` indicates the name refers to a function. A value of `0` or `false` indicates the name does not refer to a function. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
functionGroupId	Specifies the function group index if the defined name refers to a function. The function

Attributes	Description
(Function Group Id)	group defines the general category for the function. This attribute is used when there is an add-in or other code project associated with the file. The following functionGroupIds are defined in SpreadsheetML for applications that support the association of an add-in or code project for their workbook: • 1 Financial • 2 Date and Time • 3 Math and Trig • 4 Statistical • 5 Lookup and Reference • 6 Database • 7 Text • 8 Logical • 9 Information • 10 Commands • 11 Customizing • 12 Macro Control • 13 DDE / External • 14 User Defined • 15 Engineering • 16 Cube The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
help (Help)	Specifies the help topic to display for this defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
hidden (Hidden Name)	Specifies a boolean value that indicates whether the defined name is hidden in the user interface. A value of `1` or `true` indicates the name is hidden. A value of `0` or `false` indicates the name is not hidden. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
localSheetId (Local Name Sheet Id)	Specifies the sheet index in this workbook where data from an external reference is displayed. [*Example*: In the following example, the defined name refers to a range whose data source is an external database called “Northwind_Database”: `<definedName name="Northwind_Database"`

Attributes	Description
	`localSheetId="2">Sheet5!$A$1:$T$47</definedName>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (Defined Name)	Specifies the name that appears in the user interface for the defined name. This attribute is required. The following built-in names are defined in this SpreadsheetML specification: • • **Print** • _xlnm .Print_Area: this defined name specifies the workbook's print area. • _xlnm .Print_Titles: this defined name specifies the row(s) or column(s) to repeat at the top of each printed page. • • **Filter & Advanced Filter** • _xlnm .Criteria: this defined name refers to a range containing the criteria values to be used in applying an advanced filter to a range of data. • _xlnm ._FilterDatabase: can be one of the following a. this defined name refers to a range to which an advanced filter has been applied. This represents the source data range, unfiltered. b. This defined name refers to a range to which an AutoFilter has been applied. • _xlnm .Extract: this defined name refers to the range containing the filtered output values resulting from applying an advanced filter criteria to a source range. • • **Miscellaneous** • _xlnm .Consolidate_Area: the defined name refers to a consolidation area. • _xlnm .Database: the range specified in the defined name is from a database data source. • _xlnm .Sheet_Title: the defined name refers to a sheet title. • Built-in names reserved by SpreadsheetML begin with "_xlnm.". End users shall not use this string for custom names in the user interface. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
publishToServer (Publish To Server)	Specifies a boolean value that indicates whether the defined name is included in the version of the workbook that is published to or rendered on a Web or application server. A value of `1` or `true` indicates the name shall be published. A value of `0` or `false` indicates the name shall not be published. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
	datatype.
shortcutKey (Shortcut Key)	Specifies the keyboard shortcut for the defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
statusBar (Status Bar)	Specifies text that is displayed on the application status bar when the user places focus on the defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
vbProcedure (Procedure)	Specifies a boolean value that indicates whether the defined name is related to an external function, command, or other executable code. A value of `1` or `true` indicates the name is related to an external function, command, or other executable code, and the loading application can optionally decide whether to load and/or execute the commands. A value of `0` or `false` indicates the name does not refer to an external function, command, or other executable code. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
workbookParameter (Workbook Parameter (Server))	Specifies a boolean value that indicates that the name is used as a workbook parameter on a version of the workbook that is published to or rendered on a Web or application server. A value of `1` or `true` indicates the name is used as a workbook parameter on the application server. A value of `0` or `false` indicates the name is not used as a workbook parameter on the application server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
xlm (External Function)	Specifies a boolean value that indicates whether the defined name is related to an external function, command, or other executable code. A value of `1` or `true` indicates the name is related to an external function, command, or other executable code, and the loading application can optionally decide whether to load and/or execute the commands. A value of `0` or `false` indicates the name does not refer to an external function,

Attributes	Description
	command, or other executable code. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
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. 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_DefinedName) is located in §A.2. *end note*]

18.2.6 definedNames (Defined Names)

This element defines the collection of defined names for this workbook. Defined names are descriptive names to represent cells, ranges of cells, formulas, or constant values. Defined names can be used to represent a range on any worksheet.

[*Example*:

```
<definedNames>
  <definedName name="NamedFormula"
    comment="Comment text for defined name.">SUM(Sheet3!$B$2:$B$9)</definedName>
  <definedName name="NamedRange">Sheet3!$A$1:$C$12</definedName>
  <definedName name="NamedRangeFromExternalReference" localSheetId="2"
    hidden="1">Sheet5!$A$1:$T$47</definedName>
</definedNames>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_DefinedNames) is located in §A.2. *end note*]

18.2.7 ext (Extension)

Each extension within an extension list shall be contained within an ext element. Extensions shall be versioned by namespace, using the uri attribute, and shall be allowed to appear in any order within the extension list. Any number of extensions shall be allowed within an extension list.

When extension lists are processed, a consumer might understand some extensions, and might not understand other extensions. The preservation model for extensions is that unprocessed extensions shall always be preserved (when consuming) and written out (when producing) in whole, as long as the underlying schema extended by the extension list remains. [*Example*: If a spreadsheetML sheet contains several extensions within an extension list, and through runtime processing that sheet is removed from the workbook, then the extensions associated with that sheet must not be written out when producing the resulting markup document. *end example*].

Markup namespaces within extensions shall not be required to be listed in the Ignorable Compatibility-Rule attribute, nor shall these namespaces be required to be listed in the PreserveElements and PreserveAttributes Compatibility-Rule attributes. [*Note*: See Part 3 for additional discussion on Application-Defined Extension Elements and processing rules. *end note*]

[*Example*:

In this example, there are two extensions written. The first extension describes a new structure that might have been defined by a fictitious second version of ISO/IEC 29500. The second extension describes a structure that might be introduced by a private party, developed independently from ISO/IEC 29500.

```
<extLst>
  <ext uri='http://purl.oclc.org/ooxml/spreadsheetml/versionTwoExtension'>
    <v2:newContent
xmln:v2='http://purl.oclc.org/ooxml/spreadsheetml/versionTwoExtension'>
      …
    </v2:newContent>
  </ext>
  <ext uri='http://www.extension.com/versionOneExtension'>
    <v2:moreContent xmlsn:v2='http://www.extension.com/versionOneExtension'>
      …
    </v2:moreContent>
  </ext>
</extList>
```

end example]

Each extension has an uri attribute, which serves as an identifier to indicate information about the extension. [*Note*: For example, the uri might state the version of a markup specification to which the content conforms, or it might state the version of a producing application that wrote the content. *end note*] Upon encountering extensions, a processing consumer shall determine whether it knows how to process extensions using the value of the uri. If the consumer knows how to process such an extension, the markup contained within that extension is processed. Otherwise, the extension content shall be preserved so long as the underlying structure being extended by the extLst has not been removed.

Attributes	Description
uri (URI)	A token to identify version and application information for this particular extension. The possible values for this attribute are defined by the W3C XML Schema token datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Extension) is located in §A.2. *end note*]

18.2.8 externalReference (External Reference)

This element defines an external reference that stores data for workbook elements.

Attributes	Description
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies a unique identifier that is used to identify a relationship to another part in the file. Relationship identifiers link the element definition with the part where data for the element is stored. 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_ExternalReference) is located in §A.2. *end note*]

18.2.9 externalReferences (External References)

This element defines the collection of external references for this workbook.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExternalReferences) is located in §A.2. *end note*]

18.2.10 extLst (Future Feature Data Storage Area)

This element provides a convention for extending spreadsheetML in defined locations within the markup specification. The locations shall be denoted with the extLst element, and are called extension lists. Extension list locations within the markup document are specified in the markup specification and can be used to store extensions to the markup specification, whether those are future version extensions of the markup specification or are private extensions implemented independently from the markup specification. Markup within an extension might not be understood by a markup consumer.

extLst elements contain ext elements, called *extensions*. See §18.2.7 for more discussion of extensions.

[*Note*: Allowing markup specification extensions and private markup extensions within an extension list does not violate interoperability because the rules articulated within §18.2.7 and Part 3, §12 describe how markup producers and consumers must generate and consume markup documents containing application defined extension elements, including how to avoid and when to generate error conditions. *end note*]

[*Note*: This element is not intended to reintroduce transitional schema into the strict conformance class. *end note*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExtensionList) is located in §A.2. *end note*]

18.2.11 fileRecoveryPr (File Recovery Properties)

This element defines properties that track the state of the workbook file, such as whether the file was saved during a crash, or whether it should be opened in auto-recover mode.

Attributes	Description
autoRecover (Auto Recover)	Specifies a boolean value that indicates whether the file is mark for auto-recovery. Applications typically mark files for auto-recover following a crash. A value of `1` or `true` indicates the file is marked for auto-recover. A value of `0` or `false` indicates the file is not marked for auto-recover. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
crashSave (Crash Save)	Specifies a boolean value that indicates whether the application last saved the workbook file after a crash. A value of `1` or `true` indicates the workbook was last saved after a crash. A value of `0` or `false` indicates was not last saved as part of a crash. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dataExtractLoad (Data Extract Load)	Specifies a boolean value that indicates whether the application last opened the workbook for data recovery. A value of `1` or `true` indicates the workbook was last opened for data recovery. A value of `0` or `false` indicates was not last opened for data recovery.

Attributes	Description
	The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
repairLoad (Repair Load)	Specifies a boolean value that indicates whether the application last opened the workbook in safe or repair mode. A value of `1` or `true` indicates the workbook was last opened in safe or repair mode. A value of `0` or `false` indicates the workbook was last opened without problems. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_FileRecoveryPr) is located in §A.2. *end note*]

18.2.12 fileSharing (File Sharing)

This element tracks file sharing settings for the workbook. When a password is to be hashed and stored in this element, it shall be hashed 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.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
<tr><td>MD4</td><td>Specifies that the MD4 algorithm, as defined by RFC 1320, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>MD5</td><td>Specifies that the MD5 algorithm, as defined by RFC 1321, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</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.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</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>
<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:

<code>< … algorithmName="SHA-1"
hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></code>

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_Xstring simple type (§22.9.2.19).</td></tr>
<tr><td>hashValue (Password Hash Value)</td><td>Specifies the hash value for the password required for editing this workbook. 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</td></tr>
</table>

Attributes	Description
	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 cryptAlgorithmSid attribute value of 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.
readOnlyRecommended (Read Only Recommended)	Specifies a boolean value that indicates on open, whether the application alerts the user that the file be marked as read-only. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
saltValue (Salt Value for Password Verifier)	Specifies the salt that 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 that 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	Specifies the number of times the hashing function shall be iteratively run (runs using

Attributes	Description
(Iterations to Run Hashing Algorithm)	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 hashValue attribute. *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
userName (User Name)	Specifies the username of the person with write reservation for this workbook. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FileSharing) is located in §A.2. *end note*]

18.2.13 fileVersion (File Version)

This element defines properties that track which version of the application accessed the data and source code contained in the file.

Attributes	Description
appName (Application Name)	Specifies the application name. When saving, applications can write their appName value and optionally write `lastEdited` and `lowestEdited` attributes to track the version of the application that performed those actions. When opening the workbook, applications can examine the value of `appName` and decide how to interpret the `lastEdited`, `lowestEdited`, and `rupBuild` attributes. The possible values for this attribute are defined by the W3C XML Schema string datatype.
codeName (Code	Specifies the GUID that identifies the code project that is associated with the workbook.

Attributes	Description
Name)	[*Note*: the primary use of this attribute is to track the version of the compiled code.] The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).
lastEdited (Last Edited Version)	Specifies the version of the application that last saved the workbook. This attribute is application-dependent. The possible values for this attribute are defined by the W3C XML Schema string datatype.
lowestEdited (Lowest Edited Version)	Specifies the earliest version of the application that saved the workbook. This value is reset any time an application that can read all data in the file saves the file. This attribute is application-dependent. The possible values for this attribute are defined by the W3C XML Schema string datatype.
rupBuild (Build Version)	Specifies the incremental public release of the application. [*Example*: Betas, service packs, and versions. *end example*] The possible values for this attribute are defined by the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_FileVersion) is located in §A.2. *end note*]

18.2.14 functionGroup (Function Group)

This element represents a single function group.

Attributes	Description
name (Name)	Specifies the name of the function group. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FunctionGroup) is located in §A.2. *end note*]

18.2.15 functionGroups (Function Groups)

This element defines the collection of function groups for the workbook.

Attributes	Description
builtInGroupCount	Specifies the count of built-in function groups that the application provides in this

Attributes	Description
(Built-in Function Group Count)	workbook. The default value for this attribute is 16. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_FunctionGroups) is located in §A.2. *end note*]

18.2.16 oleSize (Embedded Object Size)

This element defines the embedded object server for this workbook.

Attributes	Description
ref (Reference)	Specifies the reference for the embedded object. The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).

[*Note*: The W3C XML Schema definition of this element's content model (CT_OleSize) is located in §A.2. *end note*]

18.2.17 pivotCache (PivotCache)

This element represents a cache of data for pivot tables and formulas in the workbook.

Attributes	Description
cacheId (PivotCache Id)	Specifies the unique identifier for the pivot cache for this workbook in the pivot cache part. This attribute is required. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the identifier to a pivot cache definition part where cached data is stored. This attribute is required. 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_PivotCache) is located in §A.2. *end note*]

18.2.18 pivotCaches (PivotCaches)

This element enumerates pivot cache definition parts used by pivot tables and formulas in this workbook.

[*Example*:

```
<pivotCaches>
  <pivotCache cacheId="4" r:id="rId8"/>
</pivotCaches>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotCaches) is located in §A.2. *end note*]

18.2.19 sheet (Sheet Information)

This element defines a sheet in this workbook. Sheet data is stored in a separate part.

Attributes	Description
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the identifier of the sheet part where the definition for this sheet is stored. This attribute is required. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
name (Sheet Name)	Specifies the name of the sheet. This name shall be unique. This attribute is required. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
sheetId (Sheet Tab Id)	Specifies the internal identifier for the sheet. This identifier shall be unique. This attribute is required. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
state (Visible State)	Specifies the visible state of this sheet. The default value for this attribute is "visible."

Attributes	Description
	The possible values for this attribute are defined by the ST_SheetState simple type (§18.18.68).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Sheet) is located in §A.2. *end note*]

18.2.20 sheets (Sheets)

This element represents the collection of sheets in the workbook. There are different types of sheets you can create in SpreadsheetML. The most common sheet type is a worksheet; also called a spreadsheet. A worksheet is the primary document that you use in SpreadsheetML to store and work with data. A worksheet consists of cells that are organized into columns and rows.

Some workbooks might have a modular design where there is one sheet for data and another worksheet for each specific analysis performed on that data. In a complex modular system, you might have dozens of sheets, each dedicated to a specific task.

[*Example*:

```
<sheets>
  <sheet name="Sheet1" sheetId="1" r:id="rId1"/>
  <sheet name="Sheet2" sheetId="2" r:id="rId2"/>
  <sheet name="Sheet5" sheetId="3" r:id="rId3"/>
  <sheet name="Chart1" sheetId="4" r:id="rId4"/>
</sheets>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Sheets) is located in §A.2. *end note*]

18.2.21 smartTagPr (Smart Tag Properties)

This element defines a collection of smart tag properties that determine smart tag behavior in the workbook.

[*Example*:

```
<smartTagPr embed="1" show="noIndicator"/>
```

end example]

Attributes	Description
embed (Embed SmartTags)	Specifies a boolean value that indicates whether the application saves smart tags with the workbook. Smart tag information is saved both in the workbook part and the sheet parts. A value of 1 or `true` indicates the application saves smart tags with the workbook.

Attributes	Description
	A value of `0` or `false` indicates the application does not save smart tags with the workbook. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
show (Show Smart Tags)	Specifies how the application displays smart tags in the user interface. The default value for this attribute is "all." The possible values for this attribute are defined by the ST_SmartTagShow simple type (§18.18.71).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SmartTagPr) is located in §A.2. *end note*]

18.2.22 smartTagType (Smart Tag Type)

This element represents a smart tag in the workbook.

Attributes	Description
name (Name)	Specifies the element name used for a smart tag that is used by the application. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
namespaceUri (SmartTag Namespace URI)	Specifies the namespace Uniform Resource Identifier (URI) for a smart tag used by the application. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
url (Smart Tag URL)	Specifies the URL for a smart tag provided by the smart tag provider in the application. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SmartTagType) is located in §A.2. *end note*]

18.2.23 smartTagTypes (Smart Tag Types)

This element defines the collection of smart tag types in the workbook. Smart tags represent data that is recognized and labeled as a particular type.

[*Example*: For example, a person's name or a date can be recognized and labeled with a smart tag.

```
<smartTagTypes>
  <smartTagType namespaceUri="urn:schemas-openxmlformats-org:office:smarttags"
    name="date"/>
</smartTagTypes>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SmartTagTypes) is located in §A.2. *end note*]

18.2.24 webPublishing (Web Publishing Properties)

This element defines properties that relate to publishing this workbook to the Web.

Attributes	Description
allowPng (Allow PNG)	Specifies a boolean value that indicates whether the application saves images in the PNG (Portable Network Graphics) graphic format. A value of 1 or true indicates the application supports PNG . A value of 0 or false indicates the application does not support PNG. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
characterSet (Character Set)	Name of the character set the application uses when a Web page is saved. The values alllowed within this attribute are names and aliases listed in the IANA CHARACTER SETS listing found at http://www.iana.org/assignments/character-sets. If this attribute is not present then the codePage attribute can be used. The possible values for this attribute are defined by the W3C XML Schema string datatype.
css (Use CSS)	Specifies a boolean value that indicates whether the application uses Cascading Style Sheet (CSS) for font formatting on Web pages. A value of 1 or true indicates the application uses CSS for font formats in Web pages. A value of 0 or false indicates the application does not use CSS for font formats.

Attributes	Description
	The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dpi (DPI)	Specifies the DPI (defined as the number of pixels per inch) that are used to display images in Web pages. The specified DPI affects the size of graphics relative to the size of text on the screen. The default value for this attribute is 96. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
longFileNames (Enable Long File Names)	Specifies a boolean value that indicates whether the application allows file names longer than 8 octets with a three octet extension for Web pages. File names are not case-sensitive. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
targetScreenSize (Target Screen Size)	Specifies the screen size on which Web pages are displayed. The specified screen size might affect the size and layout of images on web pages. The default value of this attribute is "800x600." The possible values for this attribute are defined by the ST_TargetScreenSize simple type (§18.18.79).
thicket (Thicket)	Specifies a boolean value that indicates that the application stores supporting files such as bullets, background textures, and graphics in a separate folder from the Web page The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
vml (VML in Browsers)	Specifies a boolean value that indicates whether the application uses VML (Vector Markup Language) to display graphics in Web browsers The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_WebPublishing) is located in §A.2. *end note*]

18.2.25 webPublishObject (Web Publishing Object)

This element defines a single Web publishing object for the workbook. This element tracks basic information about an object in the workbook, such as a named range, that is published to the Web.

Attributes	Description
autoRepublish (Auto Republish)	Specifies a boolean value that indicates whether the object specified in `sourceObject` is automatically published every time the workbook is saved. A value of `1` or `true` indicates the application will publish the `sourceObject` when the workbook is saved. A value of `0` or `false` indicates the application will not publish the `sourceObject` when the workbook is saved. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
destinationFile (Destination File)	Specifies the destination file name to which the `sourceObject` is published. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
divId (Div Id)	Specifies the destination bookmark (div id) for the published object. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
id (Id)	Specifies the number, in "nnnnn" format, used in generated div id, in style id's, token filenames, and other variables. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sourceObject (Source Object)	Specifies the named range to be published. If omitted, the entire workbook is published. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
title (Title)	Specifies the title of the published item. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_WebPublishObject) is located in §A.2. *end note*]

18.2.26 webPublishObjects (Web Publish Objects)

This element defines the collection of Web publishing objects in the workbook.

Attributes	Description
count (Count)	Specifies the number of items in the collection.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_WebPublishObjects) is located in §A.2. *end note*]

18.2.27 workbook (Workbook)

The workbook element is the top level element. It contains elements and attributes that encompass the data content of the workbook. The workbook's child elements each have their own subclause references, and these are shown in the child elements table below. A partial list of the workbook's structures that these elements represent are:

- Sheets: represents the collection of worksheets in the workbook. The sheets are the central structure within a workbook, and contain the text, numbers, dates, formulas, and other elements of a workbook.
- Views: SpreadsheetML defines a collection of Workbook views that define basic window dimensions and position of the workbook if it is ever displayed by a spreadsheet application. It also defines a collection of Custom Workbook Views that allows SpreadsheetML to describe one or more views of the data within a workbook.
- Properties: the workbook has several property collection that store basic workbook settings, such as the date system to use, file protection settings, calculation settings, and smart tag behaviors.
- Names: words or strings of characters that represent cells, ranges of cells, formulas, or constant values.

[*Example*:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<workbook xmlns="http://purl.oclc.org/ooxml/spreadsheetml/main"
mlns:r="http://purl.oclc.org/ooxml/officeDocument/relationships">
  <fileVersion lastEdited="4" lowestEdited="4" rupBuild="4017"/>
  <workbookPr vbName="ThisWorkbook" defaultThemeVersion="123820"/>
  <bookViews>
    <workbookView xWindow="120" yWindow="45" windowWidth="15135"
      windowHeight="7650" activeTab="4"/>
  </bookViews>
  <sheets>
    <sheet name="Sheet1" sheetId="1" r:id="rId1"/>
    <sheet name="Sheet2" sheetId="2" r:id="rId2"/>
    <sheet name="Sheet5" sheetId="3" r:id="rId3"/>
    <sheet name="Chart1" sheetId="4" r:id="rId4"/>
  </sheets>
```



```
    <definedNames>
      <definedName name="MyDefinedName">Sheet3!$A$1:$C$12</definedName>
    </definedNames>
    <calcPr calcId="122211" calcMode="autoNoTable" refMode="R1C1" iterate="1"
      fullPrecision="0"/>
    <customWorkbookViews>
      <customWorkbookView name="CustomView1"
        guid="{CE6681F1-E999-414D-8446-68A031534B57}" maximized="1" xWindow="1"
        yWindow="1" windowWidth="1024" windowHeight="547" activeSheetId="1"/>
    </customWorkbookViews>
    <pivotCaches>
      <pivotCache cacheId="0" r:id="rId8"/>
    </pivotCaches>
    <smartTagPr embed="1" show="noIndicator"/>
    <smartTagTypes>
      <smartTagType namespaceUri="urn:schemas-openxmlformats-org:office:smarttags"
        name="date"/>
    </smartTagTypes>
    <webPublishing codePage="1252"/>
  </workbook>
```

end example]

Attributes	Description
conformance (Document Conformance Class)	Specifies the conformance class (§2.1) to which the SpreadsheetML document conforms. If this attribute is omitted, its default value is `transitional`. [*Example*: Consider the following SpreadsheetML Workbook part markup: `<workbook conformance="strict">` … `</workbook>` This document has a conformance attribute value of `strict`, therefore it conforms to the SML 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_Workbook) is located in §A.2. *end note*]

18.2.28 workbookPr (Workbook Properties)

This element defines a collection of workbook properties.

[*Example*:

```
<workbookPr showObjects="none" saveExternalLinkValues="0"
  defaultThemeVersion="123820"/>
```

end example]

Attributes	Description
allowRefreshQuery (Allow Refresh Query)	Specifies a boolean value that indicates whether the application will refresh query tables in this workbook. A value of `1` or `true` indicates the application will refresh query tables when the workbook is loaded. A value of `0` or `false` indicates the application will not refresh query tables. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoCompressPictures (Auto Compress Pictures)	Specifies a boolean value that indicates the application automatically compressed pictures in the workbook. A value of `1` or `true` indicates the application automatically compresses pictures of the workbook. When a picture is compressed, the application: • Reduces resolution (to 96 dots per inch (dpi) for Web and 200 dpi for print), and unnecessary information is discarded. • Discards extra information. [*Example*: When a picture has been cropped or resized, the "hidden" parts of the picture are stored in the file. *end example*] • Compress the picture, if possible. A value of `0` or `false` indicates the application does not compress pictures in this workbook. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
backupFile (Create Backup File)	Specifies a boolean value that indicates whether the application creates a backup of the workbook on save. A value of `1` or `true` indicates the application creates a backup of the workbook on save.

Attributes	Description
	A value of 0 or false indicates the application does not create a backup. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
checkCompatibility (Check Compatibility On Save)	Specifies a boolean value that indicates whether the application checks for compatibility when saving this workbook to older file formats. A value of 1 or true indicates the application performs a compatibility check when saving to legacy binary formats. A value of 0 or false indicates the application does not perform a compatibility check when saving to legacy binary formats. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
codeName (Code Name)	Specifies the codename of the application that created this workbook. Use this attribute to track file content in incremental releases of the application. The possible values for this attribute are defined by the W3C XML Schema string datatype.
date1904 (Date 1904)	Value that indicates whether to use a 1900 or 1904 date system when converting serial date-times in the workbook to dates. A value of 1 or true indicates the workbook uses the 1904 date system. A value of 0 or false indicates the workbook uses the 1900 date system. (See §18.17.4.1 for the definition of the date systems.) The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
defaultThemeVersion (Default Theme Version)	Specifies the default version of themes to apply in the workbook. The value for defaultThemeVersion depends on the application. SpreadsheetML defaults to the form [version][build], where [version] refers to the version of the application, and [build] refers to the build of the application when the themes in the user interface changed. The possible values for this attribute are defined by the W3C XML Schema unsignedInt

Attributes	Description
	datatype.
filterPrivacy (Filter Privacy)	Specifies a boolean value that indicates whether the application has inspected the workbook for personally identifying information (PII). If this flag is set, the application warns the user any time the user performs an action that will insert PII into the document. [*Example*: Inserting a comment might insert the user's name. *end example*] A value of `1` or `true` indicates the application will warn the user when they insert PII into the workbook. A value of `0` or `false` indicates the application will not warn the user when they insert PII into the workbook; the workbook has not been inspected for PII. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
hidePivotFieldList (Hide Pivot Field List)	Specifies a boolean value that indicates whether a list of fields is shown for pivot tables in the application user interface. A value of `1` or `true` indicates a list of fields is show for pivot tables. A value of `0` or `false` indicates a list of fields is not shown for pivot tables. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
promptedSolutions (Prompted Solutions)	Specifies a boolean value that indicates whether the user has received an alert to load Smart Document components. A value of `1` or `true` indicates the user received an alert to load SmartDoc. A value of `0` or `false` indicates the user did not receive an alert. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
publishItems (Publish Items)	Specifies a boolean value that indicates whether the publish the workbook or workbook items to the application server. A value of `1` or `true` indicates that workbook items are published. A value of `0` or `false` indicates that the workbook is published.

Attributes	Description
	The default value for this attribute is `false`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
refreshAllConnections (Refresh all Connections on Open)	Specifies a boolean value that indicates whether the workbok shall refresh all the connections to data sources during load. The default value for this attribute is `false`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
saveExternalLinkValues (Save External Link Values)	Specifies a boolean value that indicates whether the application will cache values retrieved from other workbooks via an externally linking formula. Data is cached at save. A value of `1` or `true` indicates data from externally linked formulas is cached. A supporting part is written out containing a cached cell table from the external workbook. A value of `0` or `false` indicates data from externally linked formulas is not cached. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showBorderUnselectedTables (Show Border Unselected Table)	Specifies a boolean value that indicates whether a border is drawn around unselected tables in the workbook. A value of `1` or `true` indicates borders are drawn around unselected tables. A value of `0` or `false` indicates borders are not drawn around unselected tables. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showInkAnnotation (Show Ink Annotations)	Specifies a boolean value that indicates whether the book shows ink annotations. A value of `1` or `true` indicates that ink annotations are shown in the workbook. A value of `0` or `false` indicates that ink annotations are not shown in the workbook. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
showObjects (Show Objects)	Specifies how the application shows embedded objects in the workbook. This attribute it optional. The default value for this attribute is "all." The possible values for this attribute are defined by the ST_Objects simple type (§18.18.48).
showPivotChartFilter (Show Pivot Chart Filter)	Specifies a boolean value that indicates whether filtering options are shown for pivot charts in the workbook. A value of `1` or `true` indicates filtering options shall be shown for pivot charts. A value of `0` or `false` indicates filtering options shall not be shown. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
updateLinks (Update Links Behavior)	Specifies how the application updates external links when the workbook is opened. The default value for this attribute is userSet. The possible values for this attribute are defined by the ST_UpdateLinks simple type (§18.18.87).

[*Note*: The W3C XML Schema definition of this element's content model (CT_WorkbookPr) is located in §A.2. *end note*]

18.2.29 workbookProtection (Workbook Protection)

This element specifies options for protecting data in the workbook. Applications might use workbook protection to prevent anyone from accidentally changing, moving, or deleting important data. This protection can be ignored by applications which choose not to support this optional protection mechanism.

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.

[*Note*: Worksheet or workbook element protection should not be confused with file security. It is not meant to make your workbook safe from unintentional modification, and cannot protect it from malicious modification. *end note*]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>lockRevision (Lock Revisions)</td><td>Specifies a boolean value that indicates whether the workbook is locked for revisions.

The possible values for this attribute are defined by the W3C XML Schema boolean datatype.</td></tr>
<tr><td>lockStructure (Lock Structure)</td><td>Specifies a boolean value that indicates whether structure of workbook is locked.

A value of 1 or true indicates the structure of the workbook is locked. Worksheets in the workbook can't be moved, deleted, hidden, unhidden, or renamed, and new worksheets can't be inserted.

A value of 0 or false indicates the structure of the workbook is not locked.

The default value for this attribute is false.

The possible values for this attribute are defined by the W3C XML Schema boolean datatype.</td></tr>
<tr><td>lockWindows (Lock Windows)</td><td>Specifies a boolean value that indicates whether the windows that comprise the workbook are locked.

A value of 1 or true indicates the workbook windows are locked. Windows are the same size and position each time the workbook is opened.

A value of 0 or false indicates the workbook windows are not locked.

The default value for this attribute is false.

The possible values for this attribute are defined by the W3C XML Schema boolean datatype.</td></tr>
<tr><td>revisionsAlgorithmName (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 for the revisionsHashValue attribute.

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.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
<tr><td>MD4</td><td>Specifies that the MD4 algorithm, as defined by RFC 1320, shall be used.

[Note: It is recommended that applications should avoid using</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
<tr><td><code>MD5</code></td><td>Specifies that the MD5 algorithm, as defined by RFC 1321, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</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.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</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>

[Example: Consider an Office Open XML document with the following information stored in one of its protection elements:

<code>< … revisionsAlgorithmName="SHA-1"
revisionsHashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></code>

The revisionsAlgorithmName attribute value of “<code>SHA-1</code>” specifies that the SHA-1 hashing algorithm shall be used to generate a hash from the user-defined password. end example]

The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).</td></tr>
<tr><td>revisionsHashValue (Password Hash Value)</td><td>Specifies the hash value for the password stored for unlocking revisions in this workbook. 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</td></tr>
</table>

Attributes	Description
	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 a SpreadsheetML document with the following information stored in one of its protection elements: `<… revisionsAlgorithmName="SHA-1"` `  revisionsHashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The revisionsHashValue 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 revisionsAlgorithmName 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.
revisionsSaltValue (Salt Value for Password Verifier)	Specifies the salt that was prepended to the user-supplied password before it was hashed using the hashing algorithm defined by the preceding attribute values to generate the revisionsHashValue attribute, and that 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: `<… revisionsSaltValue="ZUdHa+D8F/OAKP3I7ssUnQ=="` `  revisionsHashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The revisionsSaltValue 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.
revisionsSpinCount (Iterations to Run	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

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>Hashing Algorithm)</td><td>of the iteration as the input for the next iteration) when attempting to compare a user-supplied password with the value stored in the revisionsHashValue 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:

<code><… revisionsSpinCount="100000"
revisionHashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></code>

The revisionsSpinCount attribute value of <code>100000</code> specifies that the hashing function must be run one hundred thousand times to generate a hash value for comparison with the revisionsHashValue attribute. end example]

The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.</td></tr>
<tr><td>workbookAlgorithmName (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 for the workbookHashValue attribute.

The following values are reserved:
<table>
<tr><th>Value</th><th>Algorithm</th></tr>
<tr><td><code>MD2</code></td><td>Specifies that the MD2 algorithm, as defined by RFC 1319, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
<tr><td><code>MD4</code></td><td>Specifies that the MD4 algorithm, as defined by RFC 1320, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
<tr><td><code>MD5</code></td><td>Specifies that the MD5 algorithm, as defined by RFC 1321, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>RIPEMD-128</td><td>Specifies that the RIPEMD-128 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</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>
<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:

<code>< … workbookAlgorithmName="SHA-1"</code>
<code>workbookHashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></code>

The workbookAlgorithmName 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_Xstring simple type (§22.9.2.19).</td></tr>
<tr><td>workbookHashValue (Password Hash Value)</td><td>Specifies the hash value for the password stored for unlocking this workbook. 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 a SpreadsheetML document with the following information stored in one of its protection elements:

<code><… workbookAlgorithmName="SHA-1"</code></td></tr>
</table>

Attributes	Description
	`workbookHashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The workbookHashValue 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 workbookAlgorithmName 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.
workbookSaltValue (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 workbookHashValue 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: `<… workbookSaltValue="ZUdHa+D8F/OAKP3I7ssUnQ=="` `workbookHashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The workbookSaltValue 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.
workbookSpinCount (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 workbookHashValue 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: `<… workbookSpinCount="100000"` `revisionHashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" />` The workbookSpinCount 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 workbookHashValue attribute. *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_WorkbookProtection) is located in §A.2. *end note*]

18.2.30 workbookView (Workbook View)

This element specifies a single Workbook view.

Units for window widths and other dimensions are expressed in twips. Twip measurements are portable between different display resolutions. The formula is (screen pixels) * (20 * 72) / (logical device dpi), where the logical device dpi can be different for x and y coordinates.

Attributes	Description
activeTab (Active Sheet Index)	Specifies an unsignedInt that contains the index to the active sheet in this book view. The default value for this attribute is 0. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
autoFilterDateGrouping (AutoFilter Date Grouping)	Specifies a boolean value that indicates whether to group dates when presenting the user with filtering options in the user interface. A value of `1` or `true` indicates that dates are grouped. A value of `0` or `false` indicates that dates are not grouped. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
firstSheet (First Sheet)	Specifies the index to the first sheet in this book view.

Attributes	Description
	The default value for this attribute is 0. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
minimized (Minimized)	Specifies a boolean value that indicates whether the workbook window is minimized. A value of 1 or true indicates the workbook window is minimized. A value of 0 or false indicates the workbook window is not minimized. The default value for this attribute is false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showHorizontalScroll (Show Horizontal Scroll)	Specifies a boolean value that indicates whether to display the horizontal scroll bar in the user interface. A value of 1 or true indicates that the horizontal scrollbar shall be shown. A value of 0 or false indicates that the horizontal scrollbar shall not be shown. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showSheetTabs (Show Sheet Tabs)	Specifies a boolean value that indicates whether to display the sheet tabs in the user interface. A value of 1 or true indicates that sheet tabs shall be shown. A value of 0 or false indicates that sheet tabs shall not be shown. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showVerticalScroll (Show Vertical Scroll)	Specifies a boolean value that indicates whether to display the vertical scroll bar. A value of 1 or true indicates the vertical scrollbar shall be shown. A value of 0 or false indicates the vertical scrollbar shall not be shown. The default value for this attribute is true. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
	datatype.
tabRatio (Sheet Tab Ratio)	Specifies ratio between the workbook tabs bar and the horizontal scroll bar. The default value for this attribute is 600. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
visibility (Visibility)	Specifies visible state of the workbook window. The default value for this attribute is "visible." The possible values for this attribute are defined by the ST_Visibility simple type (§18.18.89).
windowHeight (Window Height)	Specifies the height of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
windowWidth (Window Width)	Specifies the width of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
xWindow (Upper Left Corner (X Coordinate))	Specifies the X coordinate for the upper left corner of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the W3C XML Schema int datatype.
yWindow (Upper Left Corner (Y Coordinate))	Specifies the Y coordinate for the upper left corner of the workbook window. The unit of measurement for this value is twips. The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_BookView) is located in §A.2. *end note*]

18.3 Worksheets

Sheets are the central structures within a workbook, and are where the user does most of their spreadsheet work. The most common type of sheet is the worksheet, which is represented as a grid of cells. Worksheet cells can contain text, numbers, dates, and formulas. Cells can be formatted as well. Workbooks usually contain more than one sheet. To aid in the analysis of data and making informed decisions, spreadsheet applications often implement features and objects which help calculate, sort, filter, organize, and graphically display information.

Since these features are often connected very tightly with the spreadsheet grid, these are also included in the sheet definition on disk.

Other types of sheets include chart sheets and dialog sheets.

Note that sheet information is organized into three main sections:

- Top-level sheet properties (everything before sheetData)
- The cell table (sheetData)
- Supporting sheet features (everything after sheetData)

18.3.1 Worksheets

The following elements define a sheet and its contents:

18.3.1.1 anchor (Object Cell Anchor)

This element specifies the position of an embedded object or embedded control.

[*Example*: The following example demonstrates an embedded object whose top-left corner is at the top-left point of the cell in the first column and first row and whose bottom-right corner is offset horizontally into the cell at the fifth column and eleventh row.

```
<oleObject … >
  <objectPr … >
    <anchor sizeWithCells="true">
      <from>
        <col>0</col>
        <colOff>0</colOff>
        <row>0</row>
        <rowOff>0</rowOff>
      </from>
      <to>
        <col>4</col>
        <colOff>182880</colOff>
        <row>10</row>
        <rowOff>0</rowOff>
      </to>
    <anchor>
  </objectPr>
</oleObject>
```

end example]

Attributes	Description
moveWithCells	Specifies that the object moves with its underlying cells.

Attributes	Description
(Move With Cells)	[*Example*: `<anchor moveWithCells="true" … >` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sizeWithCells (Size With Cells)	Specifies that the object resizes with its underlying cells. [*Example*: `<anchor sizeWithCells="true" … >` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
z-order (Z-Order)	Specifies the Z-order index of the object. Higher numbers indicate higher Z-order. Objects with higher Z-order are rendered on top of objects with lower Z-order when they intersect or overlap. [*Example*: `<anchor z-order="10 " … >` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ObjectAnchor) is located in §A.2. *end note*]

18.3.1.2 autoFilter (AutoFilter Settings)

AutoFilter temporarily hides rows based on a filter criteria, which is applied column by column to a table of data in the worksheet. This collection expresses AutoFilter settings.

[*Example*: This example expresses a filter indicating to 'show only values greater than 0.5'. The filter is being applied to the range B3:E8, and the criteria is being applied to values in the column whose colId is 1 (zero based column numbering, from left to right). Therefore any rows must be hidden if the value in that particular column is less than or equal to 0.5.


```
<autoFilter ref="B3:E8">
  <filterColumn colId="1">
    <customFilters>
      <customFilter operator="greaterThan" val="0.5"/>
    </customFilters>
  </filterColumn>
</autoFilter>
```

end example]

Attributes	Description
ref (Cell or Range Reference)	Reference to the cell range to which the AutoFilter is applied. The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).

[*Note*: The W3C XML Schema definition of this element's content model (CT_AutoFilter) is located in §A.2. *end note*]

18.3.1.3 brk (Break)

Individual row or column breaks

Attributes	Description
id (Id)	Zero-based row or column Id of the page break. Breaks occur above the specified row and left of the specified column. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
man (Manual Page Break)	Manual Break flag. 1 means the break is a manually inserted break. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
max (Maximum)	Zero-based index of end row or column of the break. For row breaks, specifies column index; for column breaks, specifies row index. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
min (Minimum)	Zero-based index of start row or column of the break. For row breaks, specifies column index; for column breaks, specifies row index. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

Attributes	Description
pt (Pivot-Created Page Break)	Flag indicating that a PivotTable created this break. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Break) is located in §A.2. *end note*]

18.3.1.4 c (Cell)

This collection represents a cell in the worksheet. Information about the cell's location (reference), value, data type, formatting, and formula is expressed here.

[*Example*: This example shows the information stored for a cell whose address in the grid is C6, whose style index is '6', and whose value metadata index is '15'. The cell contains a formula as well as a calculated result of that formula.

```
<c r="C6" s="1" vm="15">
  <f>CUBEVALUE("xlextdat9 Adventure Works",C$5,$A6)</f>
  <v>2838512.355</v>
</c>
```

end example]

While a cell can have a formula element f and a value element v, when the cell's type t is inlineStr then only the element is is allowed as a child element.

[*Example*:

Here is an example of expressing a string in the cell rather than using the shared string table.

```
<row r="1" spans="1:1">
  <c r="A1" t="inlineStr">
    <is><t>This is inline string example</t></is>
  </c>
</row>
```

end example]

Attributes	Description
cm (Cell Metadata Index)	The zero-based index of the cell metadata record associated with this cell. Metadata information is found in the Metadata Part. Cell metadata is extra information stored at the cell level, and is attached to the cell (travels through moves, copy / paste, clear, etc). Cell metadata is not accessible via formula reference. The possible values for this attribute are defined by the W3C XML Schema unsignedInt

Attributes	Description
	datatype.
ph (Show Phonetic)	A Boolean value indicating if the spreadsheet application should show phonetic information. Phonetic information is displayed in the same cell across the top of the cell and serves as a 'hint' which indicates how the text should be pronounced. This should only be used for East Asian languages. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
r (Reference)	An A1 style reference to the location of this cell The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
s (Style Index)	The index of this cell's style. Style records are stored in the Styles Part. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
t (Cell Data Type)	An enumeration representing the cell's data type. The possible values for this attribute are defined by the ST_CellType simple type (§18.18.11).
vm (Value Metadata Index)	The zero-based index of the value metadata record associated with this cell's value. Metadata records are stored in the Metadata Part. Value metadata is extra information stored at the cell level, but associated with the value rather than the cell itself. Value metadata is accessible via formula reference. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Cell) is located in §A.2. *end note*]

18.3.1.5 cellSmartTag (Cell Smart Tag)

Single smart tag associated with a cell. There can be more than one cellSmartTag for a cell.

Attributes	Description
deleted (Deleted)	Boolean flag indicating that the application shouldn't display a particular smart tag in the cell. [*Example*: When the user has chosen to explicitly remove the Smart Tag by interacting with the application's user interface. *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
type (Smart Tag Type Index)	Book-level zero-based index of the smart tag type. This index references a `<smartTagType>` element in the `<smartTagTypes>` collection in the workbook start

Attributes	Description
	part. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
xmlBased (XML Based)	Boolean flag indicating the Smart Tag recognition is triggered because the cell is associated with an XML map (schema-based semantic recognition), as contrasted with the more usual cell-content-based recognition of smart tags. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellSmartTag) is located in §A.2. *end note*]

18.3.1.6 cellSmartTagPr (Smart Tag Properties)

Represents a single property of a smart tag in a cell; contains a key-value pair.

Attributes	Description
key (Key Name)	Key name of a single property of a smart tag in a cell. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
val (Value)	String value of a single property of a smart tag in a cell. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellSmartTagPr) is located in §A.2. *end note*]

18.3.1.7 cellSmartTags (Cell Smart Tags)

The element is used to label the cell with a smart tag. A cell can be determined to have semantic meaning and the cell containing this data can be labeled with a smart tag. The actions which can be taken depend on the semantic meaning of the data and the actions that the application decides to associate with that type of smart tag.

[*Example*: If you recently sent mail to "Chad Rothschiller", and you type the name into a cell on the worksheet, the name is recognized and given a smart tag with actions you can take including Send Mail, Schedule a Meeting, Open Contact, or Add to Contacts.

end example]

An application can decide that the smart tag indicators appear in the cell in the worksheet.

This collection represents a collection of smart tags on a cell.

[*Example*: This example expresses a smart tag associated with cell A1. The @type is used to associate this smart tag with a workbook-level smart tag type defined in the workbook start part.

```
<cellSmartTags r="A1">
  <cellSmartTag type="0"/>
</cellSmartTags>
```

end example]

Attributes	Description
r (Reference)	Reference to the cell that contains this set of smart tags. The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellSmartTags) is located in §A.2. *end note*]

18.3.1.8 cellWatch (Cell Watch Item)

The watch window is a single UI location where the application user can keep track of certain cell formulas & values which they have chosen to be in the set of watched cells. This element expresses the cell address of a cell being watched. It is always a reference to a single cell.

Attributes	Description
r (Reference)	Cell reference of the cell being watched. The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellWatch) is located in §A.2. *end note*]

18.3.1.9 cellWatches (Cell Watch Items)

Collection of cells on this worksheet being watched in the 'watch window'.

[*Example*: In this example, cells B3 and B4 are being watched.


```
<cellWatches>
  <cellWatch r="B3"/>
  <cellWatch r="B4"/>
</cellWatches>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellWatches) is located in §A.2. *end note*]

18.3.1.10 cfRule (Conditional Formatting Rule)

This collection represents a description of a conditional formatting rule.

[*Example*:

This example shows a conditional formatting rule highlighting cells whose values are greater than 0.5. Note that in this case the content of `<formula>` is a static value, but can also be a formula expression.

```
<conditionalFormatting sqref="E3:E9">
  <cfRule type="cellIs" dxfId="0" priority="1" operator="greaterThan">
    <formula>0.5</formula>
  </cfRule>
</conditionalFormatting>
```

end example]

Only rules with a type attribute value of `expression` support formula syntax.

Attributes	Description
aboveAverage (Above Or Below Average)	Indicates whether the rule is an "above average" rule. 1 indicates 'above average'. This attribute is ignored if type is not equal to `aboveAverage`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
bottom (Bottom N)	Indicates whether a "top/bottom n" rule is a "bottom n" rule. 1 indicates 'bottom'. This attribute is ignored if type is not equal to `top10`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dxfId (Differential Formatting Id)	This is an index to a dxf element in the Styles Part indicating which cell formatting to apply when the conditional formatting rule criteria is met. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).

Attributes	Description
equalAverage (Equal Average)	Flag indicating whether the 'aboveAverage' and 'belowAverage' criteria is inclusive of the average itself, or exclusive of that value. 1 indicates to include the average value in the criteria. This attribute is ignored if type is not equal to `aboveAverage`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
operator (Operator)	The operator in a "cell value is" conditional formatting rule. This attribute is ignored if type is not equal to `cellIs` The possible values for this attribute are defined by the ST_ConditionalFormattingOperator simple type (§18.18.15).
percent (Top 10 Percent)	Indicates whether a "top/bottom n" rule is a "top/bottom n percent" rule. This attribute is ignored if type is not equal to `top10`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
priority (Priority)	The priority of this conditional formatting rule. This value is used to determine which format should be evaluated and rendered. Lower numeric values are higher priority than higher numeric values, where 1 is the highest priority. The possible values for this attribute are defined by the W3C XML Schema int datatype.
rank (Rank)	The value of "n" in a "top/bottom n" conditional formatting rule. This attribute is ignored if type is not equal to `top10`. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
stdDev (StdDev)	The number of standard deviations to include above or below the average in the conditional formatting rule. This attribute is ignored if type is not equal to `aboveAverage`. If a value is present for stdDev and the rule type = `aboveAverage`, then this rule is automatically an "above or below N standard deviations" rule. The possible values for this attribute are defined by the W3C XML Schema int datatype.
stopIfTrue (Stop If True)	If this flag is 1, no rules with lower priority shall be applied over this rule, when this rule evaluates to true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
text (Text)	The text value in a "text contains" conditional formatting rule. This attribute is ignored if type is not equal to `containsText`. The possible values for this attribute are defined by the W3C XML Schema string datatype.

Attributes	Description
timePeriod (Time Period)	The applicable time period in a "date occurring…" conditional formatting rule. This attribute is ignored if type is not equal to `timePeriod`. The possible values for this attribute are defined by the ST_TimePeriod simple type (§18.18.82).
type (Type)	Type of conditional formatting rule. The possible values for this attribute are defined by the ST_CfType simple type (§18.18.12).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CfRule) is located in §A.2. *end note*]

18.3.1.11 cfvo (Conditional Format Value Object)

Describes the values of the interpolation points in a gradient scale.

[*Example*: This example demonstrates a color scale conditional formatting rule, which defines a color for the minimum value in the range of cell values, a color for the midpoint value, and a color for the maximum value in the in the range of cell values. Information is given about how to define the midpoint. In this case, it is the 50 percent mark.

```
<colorScale>
  <cfvo type="min" val="0"/>
  <cfvo type="percent" val="50"/>
  <cfvo type="max" val="0"/>
  <color rgb="FFFF0000"/>
  <color rgb="FFFFFF00"/>
  <color rgb="FF00B050"/>
</colorScale>
```

The first `<cfvo>` element corresponds with the first `<color>` definition, and so on.

end example]

Attributes	Description
gte (Greater Than Or Equal)	For icon sets, determines whether this threshold value uses the greater than or equal to operator. 0 indicates 'greater than' is used instead of 'greater than or equal to'. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
type (Type)	The type of this conditional formatting value object. [*Example*: 'min' and 'max' would be used (in conjunction with `@val`) to express the lower and upper values to be used in a gradient. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_CfvoType simple type (§18.18.13).
val (Value)	The value of this conditional formatting value object. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Cfvo) is located in §A.2. *end note*]

18.3.1.12 chartsheet (Chart Sheet)

This is the root element of Chartsheet Parts in a SpreadsheetML document.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Chartsheet) is located in §A.2. *end note*]

18.3.1.13 col (Column Width & Formatting)

Defines column width and column formatting for one or more columns of the worksheet.

[*Example*: This example shows that column 5 (E) has width and style information applied.

```
<col min="5" max="5" width="9.140625" style="3"/>
```

end example]

Attributes	Description
bestFit (Best Fit Column Width)	Flag indicating if the specified column(s) is set to 'best fit'. 'Best fit' is set to `true` under these conditions: • The column width has never been manually set by the user, AND • The column width is not the default width • • 'Best fit' means that when numbers are typed into a cell contained in a 'best fit' column, the column width should automatically resize to display the number. [*Note*: In best fit cases, column width must not be made smaller, only larger. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
collapsed (Collapsed)	Flag indicating if the outlining of the affected column(s) is in the collapsed state. See description of row collapsed and outlinePr element's summaryBelow and summaryRight attributes for detailed information. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
customWidth (Custom Width)	Flag indicating that the column width for the affected column(s) is different from the default or has been manually set. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
hidden (Hidden Columns)	Flag indicating if the affected column(s) are hidden on this worksheet. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
max (Maximum Column)	Last column affected by this 'column info' record. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
min (Minimum Column)	First column affected by this 'column info' record. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
outlineLevel (Outline Level)	Outline level of affected column(s). Range is 0 to 7. See description of outlinePr element's summaryBelow and summaryRight attributes for detailed information. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
phonetic (Show Phonetic Information)	Flag indicating if the phonetic information should be displayed by default for the affected column(s) of the worksheet. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
style (Style)	Default style for the affected column(s). Affects cells not yet allocated in the column(s). In other words, this style applies to new columns. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
width (Column Width)	Column width measured as the number of characters of the maximum digit width of the numbers 0, 1, 2, …, 9 as rendered in the normal style's font. There are 4 pixels of margin padding (two on each side), plus 1 pixel padding for the gridlines. width = Truncate([{Number of Characters} * {Maximum Digit Width} + {5 pixel padding}]/{Maximum Digit Width}*256)/256 [*Example*: Using the Calibri font as an example, the maximum digit width of 11 point font size is 7 pixels (at 96 dpi). In fact, each digit is the same width for this font. Therefore, if the cell width is 8 characters wide, the value of this attribute must be Truncate([8*7+5]/7*256)/256 = 8.7109375. *end example*]

Attributes	Description
	To translate the value of width in the file into the column width value at runtime (expressed in terms of pixels), use this calculation: =Truncate(((256 * {width} + Truncate(128/{Maximum Digit Width}))/256)*{Maximum Digit Width}) [*Example*: Using the same example as above, the calculation would be Truncate(((256*8.7109375+Truncate(128/7))/256)*7) = 61 pixels. *end example*] To translate from pixels to character width, use this calculation: =Truncate((({pixels}-5)/{Maximum Digit Width} * 100+0.5)/100 [*Example*: Using the example above, the calculation would be Truncate((61-5)/7*100+0.5)/100 = 8 characters. *end example*] [*Note*: when wide borders are applied, part of the left/right border must overlap with the 2 pixel padding on each side. Wide borders do not affect the width calculation of the column. *end note*] [*Note*: When the sheet is in the mode to view formulas instead of values, the pixel width of the column is doubled. *end note*] The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Col) is located in §A.2. *end note*]

18.3.1.14 colBreaks (Vertical Page Breaks)

Vertical page break information used for print layout view, page layout view, drawing print breaks in normal view, and for printing the worksheet.

[*Example*:

In this example, a page break has been inserted at C3 (the break occurs left and above C3).

```
<colBreaks count="1" manualBreakCount="1">
  <brk id="2" max="1048575" man="1"/>
</colBreaks>
```

end example]

Attributes	Description
count (Page Break Count)	Number of breaks in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
manualBreakCount (Manual Break Count)	Number of manual breaks in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageBreak) is located in §A.2. *end note*]

18.3.1.15 color (Data Bar Color)

One of the colors associated with the data bar or color scale.

The auto attribute shall not be used in the context of data bars.

Attributes	Description
auto (Automatic)	A boolean value indicating the color is automatic and system color dependent. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
indexed (Index)	Indexed color value. Only used for backwards compatibility. References a color in indexedColors. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rgb (Alpha Red Green Blue Color Value)	Standard Alpha Red Green Blue color value (ARGB). The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
theme (Theme Color)	A zero-based index into the `<clrScheme>` collection (§20.1.6.2), referencing a particular `<sysClr>` or `<srgbClr>` value expressed in the Theme part. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
tint (Tint)	Specifies the tint value applied to the color. If tint is supplied, then it is applied to the RGB value of the color to determine the final color applied. The tint value is stored as a double from -1.0 .. 1.0, where -1.0 means 100% darken and

Attributes	Description
	1.0 means 100% lighten. Also, 0.0 means no change. In loading the RGB value, it is converted to HLS where HLS values are (0..HLSMAX), where HLSMAX is currently 255. [*Example*: Here are some examples of how to apply tint to color: **If (tint < 0)** Lum' = Lum * (1.0 + tint) For example: Lum = 200; tint = -0.5; Darken 50% Lum' = 200 * (0.5) => 100 For example: Lum = 200; tint = -1.0; Darken 100% (make black) Lum' = 200 * (1.0-1.0) => 0 **If (tint > 0)** Lum' = Lum * (1.0-tint) + (HLSMAX – HLSMAX * (1.0-tint)) For example: Lum = 100; tint = 0.75; Lighten 75% Lum' = 100 * (1-.75) + (HLSMAX – HLSMAX*(1-.75)) = 100 * .25 + (255 – 255 * .25) = 25 + (255 – 63) = 25 + 192 = 217 For example: Lum = 100; tint = 1.0; Lighten 100% (make white) Lum' = 100 * (1-1) + (HLSMAX – HLSMAX*(1-1)) = 100 * 0 + (255 – 255 * 0) = 0 + (255 – 0) = 255 *end example*] The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Color) is located in §A.2. *end note*]

18.3.1.16 colorScale (Color Scale)

Describes a gradated color scale in this conditional formatting rule.

[*Example*:

```
<colorScale>
  <cfvo type="min" val="0"/>
  <cfvo type="max" val="0"/>
  <color theme="5"/>
  <color rgb="FFFFEF9C"/>
</colorScale>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_ColorScale) is located in §A.2. *end note*]

18.3.1.17 cols (Column Information)

Information about whole columns of the worksheet.

[*Example*:

This example shows that column 4 (D) has 'best fit' applied to it, which is also a custom width. Also, column 5 (E) is listed as having a custom width and a style applied at the column level (as opposed to the cell level).

```
<cols>
  <col min="4" max="4" width="12" bestFit="1" customWidth="1"/>
  <col min="5" max="5" width="9.140625" style="3"/>
</cols>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Cols) is located in §A.2. *end note*]

18.3.1.18 conditionalFormatting (Conditional Formatting)

A Conditional Format is a format, such as cell shading or font color, that a spreadsheet application can automatically apply to cells if a specified condition is true. This collection expresses conditional formatting rules applied to a particular cell or range.

[*Example*: This example applies a 'top10' rule to the cells C3:C8. The `@dxfId` references the formatting (defined in the styles part) to be applied to cells that match the criteria.

```
<conditionalFormatting sqref="C3:C8">
  <cfRule type="top10" dxfId="1" priority="3" rank="2"/>
</conditionalFormatting>
```

end example]

Attributes	Description
pivot (PivotTable Conditional	Flag indicating if this is conditional formatting associated with a PivotTable.

Attributes	Description
Formatting)	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sqref (Sequence of Refernces)	Range over which these conditional formatting rules apply. The possible values for this attribute are defined by the ST_Sqref simple type (§18.18.76).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ConditionalFormatting) is located in §A.2. *end note*]

18.3.1.19 control (Embedded Control)

A single embedded control.

Attributes	Description
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	This relationship ID references an Embedded Control Data part which contains control-specific properties and state information about this particular embedded control. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
name (Control Name)	The code name of the control. The possible values for this attribute are defined by the W3C XML Schema string datatype.
shapeId (Shape Id)	ID of the drawing shape in the DrawingML part with which this control is associated. The drawing is used to draw the control in the sheet. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Control) is located in §A.2. *end note*]

18.3.1.20 controlPr (Embedded Control Properties)

This element specifies the visual, positional and cell linkage properties of an embedded control.

[*Example*: The following example demonstrates an non-printing embedded control linked to cell A4 that is represented as an image:

```
<controls>
  <control … >
```



```
      <controlPr print="false" autoLine="false" linkedCell="$A$4" cf="pict"
  r:id="rId5">
        <anchor sizeWithCells="true">
          <from> … </from>
          <to> … </to>
        </anchor>
      </controlPr>
    </control>
  </controls>
```

end example]

Attributes	Description
altText (Alternative Text)	Specifies alternative text for the object, for use by assistive technologies or applications. [*Example*: `<controlPr altText="Alternate text" … />` *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
autoFill (Automatic Fill Flag)	Specifies whether the object's fill formatting is provided automatically by the application. [*Example*: `<controlPr autoFill="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoLine (Automatic Line Flag)	Specifies whether the object's line formatting is provided automatically by the application. [*Example*: `<controlPr autoLine="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoPict (Automatic	Specifies whether the object's size is formatted automatically by the application.

Attributes	Description
Size Flag)	[*Example*: `<controlPr autoPict="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
cf (Image Format)	Specifies the image format used to render the object. [*Example*: `<controlPr cf="pict" … />` *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
defaultSize (Default Size Flag)	Specifies whether the object is at its default size. [*Example*: `<controlPr defaultSize="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
disabled (Disabled Flag)	Specifies whether the object is allowed to run an attached macro. [*Example*: `<controlPr disabled="true" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
id (Relationship ID for Embedded Control Properties) Namespace: http://purl.oclc.or	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 package. The relationship explicitly targeted by this attribute shall be of relationship type http://purl.oclc.org/ooxml/officeDocument/relationships/control or the document shall be conisdered non-conformant.

Attributes	Description
g/ooxml/officeDocument/relationships	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:id="CheckBox1" w:name="CheckBox1" w:shapeid="_x0000_s1027" w:class="shape" w:w="145" w:h="28" w:align="left" />` 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).
linkedCell (Linked Formula)	Specifies the cell the control is linked to, using standard cell A1-style reference syntax as described in §18.17.2.3.1. The value in the linked cell and the index of the selected item in the object are linked together. This link is ignored if the control allows multiple selections. [*Example*: `<controlPr linkedCell="$A$4" … />` *end example*] The possible values for this attribute are defined by the ST_Formula simple type (§18.18.35).
listFillRange (List Items Source Range)	Specifies the range of source data cells used to populate the list box, using standard A1-style cell reference syntax as described in §18.17.2.3.1. [*Example*: `<controlPr listFillRange="$A$1:$A$15" … />` *end example*] The possible values for this attribute are defined by the ST_Formula simple type (§18.18.35).
locked (Locked Flag)	Specifies that the object is locked when the sheet is protected. [*Example*:

Attributes	Description
	`<controlPr locked="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
macro (Custom Function)	Specifies the custom function associated with the object. [*Example*: A macro script, add-in function, and so on. *end example*] [*Example*: `<controlPr macro="Button1_Click()" … />` *end example*] The possible values for this attribute are defined by the ST_Formula simple type (§18.18.35).
print (Print Flag)	Specifies whether the object is printed when the document is printed. [*Example*: `<controlPr print="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
recalcAlways (Recalculation Flag)	Specifies whether the object is always included in recalculation. This is used by controls that reference cells in the spreadsheet to update themselves when the spreadsheet changes. [*Example*: `<controlPr recalcAlways="true" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
uiObject (UI Object Flag)	Specifies whether the object is a UI-only object. Applications should prevent UI-only objects from being selected and edited in their user interface. [*Example*: `<objectPr uiObject="true" … />`

Attributes	Description
	end example] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ControlPr) is located in §A.2. *end note*]

18.3.1.21 controls (Embedded Controls)

Worksheets can have embedded controls embedded in them. This collection is a listing of embedded controls in this worksheet. This collection is used to reference individual Embedded Control Data part definitions, enumerate the code name of each control, and reference drawing information used to draw the control.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Controls) is located in §A.2. *end note*]

18.3.1.22 customPr (Custom Property)

The custom property element provides a mechanism to store name/value pairs of arbitrary user-defined data. The name is stored in the attribute name, the arbitrary data is stored in the binary part referenced by the relationshipId.

[*Note*: There is nothing in the binary part except the arbitrary data itself.

Custom XML Data Properties provide a preferred mechanism for storing arbitrary data. The customPr supports legacy third-party document components, as well as those situations that have a stringent need for binary parts. *end note*]

Attributes	Description
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	This relationship references the binary part containing the specified custom properties. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
name (Custom Property Name)	Name of the custom property The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomProperty) is located in §A.2. *end note*]

18.3.1.23 customProperties (Custom Properties)

This collection is used to reference binary parts containing arbitrary user-defined data.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomProperties) is located in §A.2. *end note*]

18.3.1.24 customSheetView (Custom Chart Sheet View)

This element defines custom view properties for chart sheets. [*Note*: See customSheetView (§18.3.1.25) for an example. *end note*]

Attributes	Description
guid (GUID)	Unique identifier of this custom view The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).
scale (Print Scale)	Print scaling, representing percent values. The values of this attribute shall be restricted to the range from 10 to 400. Horizontal & Vertical scale together. [*Example*: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
state (Visible State)	Visibility state of the sheet. The possible values for this attribute are defined by the ST_SheetState simple type (§18.18.68).
zoomToFit (Zoom To Fit)	Flag indicating whether chart sheet is zoom to fit window. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomChartsheetView) is located in §A.2. *end note*]

18.3.1.25 customSheetView (Custom Sheet View)

This collection stores information pertaining to one custom sheet view definition. A custom view is a collection of settings defining a particular view of the sheet. These views can be selected by the user for quick access to predefined views of the sheet.

[*Example*: This example indicates that there is both a horizontal and vertical split in the sheet view, and that the top left cell of the bottom right pane is F7. Page margin, print options, page setup, and header / footer information is also stored with this view.

```
<customSheetView guid="{F3A061A9-D5FD-4F9C-A7CD-483AD476BA25}"
  sizeWithWindow="0">
  <pane xSplit="5" ySplit="6" topLeftCell="F7"/>
  <selection/>
  <pageMargins left="0.7" right="0.7" top="0.75" bottom="0.75" header="0.3"
    footer="0.3"/>
  <printOptions gridLinesSet="0"/>
  <pageSetup paperSize="0" scale="0" orientation="portrait" printDriver="0"
    horizontalDpi="0" verticalDpi="0" copies="0"/>
  <headerFooter/>
</customSheetView>
```

end example]

Attributes	Description
colorId (Color Id)	Index to the color value for the text in row/column headings and gridlines for this custom view. This is an 'index color value' (ICV) rather than rgb value. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
filter (Filtered List)	Flag indicating whether the view contains a filtered range. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
filterUnique (Filter)	Indicates whether an advanced filter has been applied, and the option to filter out duplicate records from the data list has been selected, in this custom view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
fitToPage (Fit To Page)	Flag indicating whether this view should be fit to page when printing this custom view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
guid (GUID)	Unique identifier of this custom view. This is used to ensure uniqueness. It is generated when the view is created. Shall correspond to a customWorkbookView guid value in the

Attributes	Description
	workbook Start Part. The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).
hiddenColumns (Hidden Columns)	Flag indicating that there is one or more hidden column(s) in this custom view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
hiddenRows (Hidden Rows)	Flag indicating that there is one or more hidden row(s) in this custom view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
outlineSymbols (Show Outline Symbols)	Flag indicating whether outline symbols are displayed in this custom view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
printArea (Print Area Defined)	Flag indicating whether a print area is defined as part of this custom view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
scale (Print Scale)	Print scaling for this custom view. The values of this attribute shall be restricted to the range from 10 to 400. [*Example*: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
showAutoFilter (Show AutoFilter Drop Down Controls)	Flag indicating whether the autofilter dropdown buttons are visible in this custom view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showFormulas (Show Formulas)	Flag indicating whether formulas are shown in this custom view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showGridLines	Flag indicating whether gridlines are shown in this custom view.

Attributes	Description
(Show Grid Lines)	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showPageBreaks (Show Page Breaks)	Flag indicating whether page breaks are shown in this custom view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showRowCol (Show Headers)	Flag indicating whether row and column headers are shown in this custom view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showRuler (Show Ruler)	Flag indicating whether to show the ruler in this custom view. Only applicable if this Custom View is in Page Layout View. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
state (Visible State)	Visibility state for this custom view. The possible values for this attribute are defined by the ST_SheetState simple type (§18.18.68).
topLeftCell (Top Left Visible Cell)	Location of the top left visible cell in the bottom right pane in this custom view (when in Left-to-Right mode). The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
view (View Type)	Indicates the view type for this Custom View The possible values for this attribute are defined by the ST_SheetViewType simple type (§18.18.69).
zeroValues (Show Zero Values)	Flag indicating whether the window should display 0 (zero) values in this custom view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomSheetView) is located in §A.2. *end note*]

18.3.1.26 customSheetViews (Custom Chart Sheet Views)

Collection of custom Chart Sheet View information.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomChartsheetViews) is located in §A.2. *end note*]

18.3.1.27 customSheetViews (Custom Sheet Views)

This is a collection of custom sheet views.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomSheetViews) is located in §A.2. *end note*]

18.3.1.28 dataBar (Data Bar)

Describes a data bar conditional formatting rule.

[*Example*:

In this example a data bar conditional format is expressed, which spreads across all cell values in the cell range, and whose color is blue.

```
<dataBar>
  <cfvo type="min" val="0"/>
  <cfvo type="max" val="0"/>
  <color rgb="FF638EC6"/>
</dataBar>
```

end example]

The length of the data bar for any cell can be calculated as follows:

Data bar length = minLength + (cell value - minimum value in the range) / (maximum value in the range - minimum value in the range) * (maxLength - minLength),

where min and max length are a fixed percentage of the column width (by default, 10% and 90% respectively.)

The minimum difference in length (or increment amount) is 1 pixel.

Attributes	Description
maxLength (Maximum Length)	The maximum length of the data bar, as a percentage of the cell width. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
minLength (Minimum Length)	The minimum length of the data bar, as a percentage of the cell width. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
showValue (Show Values)	Indicates whether to show the values of the cells on which this data bar is applied. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DataBar) is located in §A.2. *end note*]

18.3.1.29 dataConsolidate (Data Consolidate)

Data consolidation settings. The dataRefs are the set of source ranges containing data to consolidate. The function indicates the function that shall be used to consolidate the data.

[*Example*:

This example demonstrates consolidating the ranges A1:C1 and A3:C3 by using the 'count' function.

```
<dataConsolidate function="count">
  <dataRefs count="2">
    <dataRef ref="A1:C1" sheet="Sheet1"/>
    <dataRef ref="A3:C3" sheet="Sheet1"/>
  </dataRefs>
</dataConsolidate>
```

end example]

Attributes	Description
function (Function Index)	Indicates which function to use when consolidating the ranges. The possible values for this attribute are defined by the ST_DataConsolidateFunction simple type (§18.18.17).
link (Link)	Create links to source data. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
startLabels (Use Starting Column Labels)	Use labels in first column. Both startLabels and topLabels can be `true` at the same time. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
topLabels (Labels In Top Row)	Use labels in top row. Both leftLabels and topLabels can be `true` at the same time. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
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. 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_DataConsolidate) is located in §A.2. *end note*]

18.3.1.30 dataRef (Data Consolidation Reference)

A single data consolidate reference. One dataRef shall use either name or sheet & ref, but not both on the same dataRef.

Attributes	Description
id (relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Used only when the source range is external to this workbook. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
name (Named Range)	Named range, either in this workbook or the external workbook referenced by r:Id. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
ref (Reference)	Cell range. The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
sheet (Sheet Name)	Sheet name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DataRef) is located in §A.2. *end note*]

18.3.1.31 dataRefs (Data Consolidation References)

Data consolidate reference collection.

Attributes	Description
count (Data Consolidation Reference Count)	Count of data consolidate references. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DataRefs) is located in §A.2. *end note*]

18.3.1.32 dataValidation (Data Validation)

A single item of data validation defined on a range of the worksheet.

Attributes	Description
allowBlank (Allow Blank)	A boolean value indicating whether the data validation allows the use of empty or blank entries. 1 means empty entries are OK and do not violate the validation constraints. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
error (Error Message)	Message text of error alert. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
errorStyle (Data Validation Error Style)	The style of error alert used for this data validation. The possible values for this attribute are defined by the ST_DataValidationErrorStyle simple type (§18.18.18).
errorTitle (Error Alert Text)	Title bar text of error alert. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
imeMode (IME Mode Enforced)	The IME (input method editor) mode enforced by this data validation. Only applies for these languages: • Chinese Simplified • Chinese Traditional • Japanese • Korean • • When imeMode is set, the input for the cell can be restricted to specific sets of characters, as specified by the value of imeMode. See the simple type referenced below for additional details. • When imeMode is set but the application's language is not one of the languages listed above, then the default value is `noControl`. The possible values for this attribute are defined by the ST_DataValidationImeMode simple type (§18.18.19).
operator (Operator)	The relational operator used with this data validation. The possible values for this attribute are defined by the ST_DataValidationOperator simple type (§18.18.20).

Attributes	Description
prompt (Input Prompt)	Message text of input prompt. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
promptTitle (Prompt Title)	Title bar text of input prompt. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
showDropDown (Show Drop Down)	A boolean value indicating whether to display a dropdown combo box for a list type data validation. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showErrorMessage (Show Error Message)	A boolean value indicating whether to display the error alert message when an invalid value has been entered, according to the criteria specified. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showInputMessage (Show Input Message)	A boolean value indicating whether to display the input prompt message. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sqref (Sequence of References)	Range over which data validation is applied. The possible values for this attribute are defined by the ST_Sqref simple type (§18.18.76).
type (Data Validation Type)	The type of data validation. The possible values for this attribute are defined by the ST_DataValidationType simple type (§18.18.21).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DataValidation) is located in §A.2. *end note*]

18.3.1.33 dataValidations (Data Validations)

This collection expresses all data validation information for cells in a sheet which have data validation features applied.

Data validation is used to specify constraints on the data that can be entered into a cell. Additional UI can be provided to help the user select values (e.g., a dropdown control on the cell or hover text when the cell is active), and to help the user understand why a particular entry was disallowed (e.g., alerts and messages).

Various data types can be selected, and logical operators (e.g., greater than, less than, equal to, etc) can be used. Additionally, instead of specifying an explicit set of values that are permitted, a cell or range reference can be used.

An input message can be specified to help the user know what kind of value is expected, and a warning message (and warning type) can be specified to alert the user when they've entered data which is not permitted based on the data validations specified in the worksheet.

[*Example*:

```
<dataValidations count="1">
  <dataValidation type="whole" errorStyle="warning" operator="greaterThan"
    showInputMessage="1" showErrorMessage="1" errorTitle="Invalid Data"
    error="The value must be a whole number greater than 0."
    promptTitle="Whole Number"
    prompt="Please enter a whole number greater than 0." sqref="A1">
    <formula1>0</formula1>
  </dataValidation>
</dataValidations>
```

end example]

Attributes	Description
count (Data Validation Item Count)	The expected number of data validation items for this worksheet. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
disablePrompts (Disable Prompts)	A boolean value indicating whether all input prompts for the worksheet are disabled. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
xWindow (Top Left Corner (X Coodrinate))	The x-coordinate (relative to window) of top-left corner of the data validation input prompt (textbox). This is per sheet, not per cell. Units in pixels. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
yWindow (Top Left Corner (Y Coordinate))	The y-coordinate (relative to window) of top-left corner of the data validation input prompt (textbox). This is per sheet, not per cell. Units in pixels. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DataValidations) is located in §A.2. *end note*]

18.3.1.34 dialogsheet (Dialog Sheet)

This is the root element for Dialogsheet parts within a SpreadsheetML document.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Dialogsheet) is located in §A.2. *end note*]

18.3.1.35 dimension (Worksheet Dimensions)

This element specifies the used range of the worksheet. It specifies the row and column bounds of used cells in the worksheet. This is optional and is not required. Used cells include cells with formulas, text content, and cell formatting. When an entire column is formatted, only the first cell in that column is considered used.

[*Example*:

```
<dimension ref="A1:C2"/>
```

end example]

Attributes	Description
ref (Reference)	The row and column bounds of all cells in this worksheet. Corresponds to the range that would contain all c elements written under sheetData. Does not support whole column or whole row reference notation. The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SheetDimension) is located in §A.2. *end note*]

18.3.1.36 drawing (Drawing)

This element indicates that the sheet contains drawing components built on the drawingML platform. The relationship Id references the part containing the drawingML definitions.

Attributes	Description
id (Relationship id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Relationship Id referencing a part containing drawingML definitions for this worksheet. 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_Drawing) is located in §A.2. *end note*]

18.3.1.37 drawingHF (Drawing Reference in Header Footer)

This element specifies the usage of drawing objects rendered in the header / footer of the sheet. It specifies an explicit relationship to the part containing the DrawingML shapes used in the header / footer. It also indicates where in the header / footer each shape belongs. One drawing object can appear in each of the left side, center and right side of the header and footer.

[*Example*: This example shows a worksheet with graphics in the header. The DrawingML part referred to by rId2 contains at least two objects. The object with ID 6 is shown in the left side of the header on the first page only. The object with ID 7 is shown in the left side of the header for the other pages.

```
<worksheet … >
  …
  <headerFooter differentFirst="1" … >
    …
  </headerFooter>
  <drawingHF r:id="rId2" lho="7" lhf="6"/>
</worksheet>
```

end example]

Attributes	Description
cfe (Center Footer for Even Pages)	Specifies the DrawingML shape to be used for the center of the footer on even pages if the differentOddEven attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*: `<drawingHF … cfe="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
cff (Center Footer for First Page)	Specifies the DrawingML shape to be used for the center of the footer on the first page if the differentFirst attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*: `<drawingHF … cff="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

Attributes	Description
cfo (Center Footer for Odd Pages)	Specifies the DrawingML shape to be used for the center of the footer on odd pages if the differentOddEven attribute of the corresponding headerFooter element (§18.3.1.46) is true. If the differentOddEven attribute is false, this attribute specifies the DrawingML shape to be used for the center of the footer on both odd and even pages. [*Example*: `<drawingHF … cfo="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
che (Center Header for Even Pages)	Specifies the DrawingML shape to be used for the center of the header on even pages if the differentOddEven attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*: `<drawingHF … che="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
chf (Center Header for First Page)	Specifies the DrawingML shape to be used for the center of the header on the first page if the differentFirst attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*: `<drawingHF … chf="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
cho (Center Header for Odd Pages)	Specifies the DrawingML shape to be used for the center of the header on odd pages if the differentOddEven attribute of the corresponding headerFooter element (§18.3.1.46) is true. If the differentOddEven attribute is false, this attribute specifies the DrawingML shape to be used for the center of the header on both odd and even pages. [*Example*: `<drawingHF … cho="5"/>`

Attributes	Description
	end example] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
id (Relationship ID for Embedded Control Properties) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID for the relationship to the DrawingML part that contains the drawing objects used in the header and footer. This DrawingML part is a separate part within the package. [*Example*: `<drawingHF r:id="rId2" lho="7" lhf="6"/>` The id attribute in the relationship reference namespace specifies that the relationship with relationship ID `rId5` must contain the drawing objects used in the header and footer. *end example*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
lfe (Left Footer for Even Pages)	Specifies the DrawingML shape to be used for the left side of the footer on even pages if the differentOddEven attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*: `<drawingHF … lfe="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
lff (Left Footer for First Page)	Specifies the DrawingML shape to be used for the left side of the footer on the first page if the differentFirst attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*: `<drawingHF … lff="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
lfo (Left Footer for Odd Pages)	Specifies the DrawingML shape to be used for the left side of the footer on odd pages if the differentOddEven attribute of the corresponding headerFooter element

Attributes	Description
	(§18.3.1.46) is true. If the differentOddEven attribute is false, this attribute specifies the DrawingML shape to be used for the left side of the footer on both odd and even pages. [*Example*: `<drawingHF … lfo="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
lhe (Left Header for Even Pages)	Specifies the DrawingML shape to be used for the left side of the header on even pages if the differentOddEven attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*: `<drawingHF … lhe="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
lhf (Left Header for First Page)	Specifies the DrawingML shape to be used for the left side of the header on the first page if the differentFirst attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*: `<drawingHF … lhf="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
lho (Left Header for Odd Pages)	Specifies the DrawingML shape to be used for the left side of the header on odd pages if the differentOddEven attribute of the corresponding headerFooter element (§18.3.1.46) is true. If the differentOddEven attribute is false, this attribute specifies the DrawingML shape to be used for the left side of the header on both odd and even pages. [*Example*: `<drawingHF … lho="5"/>` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rfe (Right Footer for Even Pages)	Specifies the DrawingML shape to be used for the right side of the footer on even pages if the differentOddEven attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*: `<drawingHF … rfe="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rff (Right Footer for First Page)	Specifies the DrawingML shape to be used for the right side of the footer on the first page if the differentFirst attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*: `<drawingHF … rff="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rfo (Right Footer for Odd Pages)	Specifies the DrawingML shape to be used for the right side of the footer on odd pages if the differentOddEven attribute of the corresponding headerFooter element (§18.3.1.46) is true. If the differentOddEven attribute is false, this attribute specifies the DrawingML shape to be used for the right side of the footer on both odd and even pages. [*Example*: `<drawingHF … rfo="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rhe (Right Header for Even Pages)	Specifies the DrawingML shape to be used for the right side of the header on even pages if the differentOddEven attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*:

Attributes	Description
	`<drawingHF … rhe="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rhf (Right Header for First Page)	Specifies the DrawingML shape to be used for the right side of the header on the first page if the differentFirst attribute of the corresponding headerFooter element (§18.3.1.46) is true. [*Example*: `<drawingHF … rhf="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rho (Right Header for Odd Pages)	Specifies the DrawingML shape to be used for the right side of the header on odd pages if the differentOddEven attribute of the corresponding headerFooter element (§18.3.1.46) is true. If the differentOddEven attribute is false, this attribute specifies the DrawingML shape to be used for the right side of the header on both odd and even pages. [*Example*: `<drawingHF … rho="5"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DrawingHF) is located in §A.2. *end note*]

18.3.1.38 evenFooter (Even Page Footer)

Even page footer value. Corresponds to even printed pages. [*Example*: Even page(s) in the sheet can not be printed if the print area is specified to be a range such that it falls outside an even page's scope. *end example*]

If no even footer is specified, then the odd footer's value is assumed for even page footers. See the evenHeader element (§18.3.1.39) description for full discussion of value content.

The possible values for this element are defined by the ST_Xstring simple type (§22.9.2.19).

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. 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 (ST_Xstring) is located in §A.6.9. *end note*]

18.3.1.39 evenHeader (Even Page Header)

Even page header value. Corresponds to even printed pages. [*Example*: Even page(s) in the sheet can not be printed if the print area is specified to be a range such that it falls outside an even page's scope. *end example*]

If no even header is specified, then odd header value is assumed for even page headers.

Header/Footer Formatting Syntax

There are a number of formatting codes that can be written inline with the actual header / footer text, which affect the formatting in the header or footer.

[*Example*:

This example shows the text "Center **Bold** Header" on the first line (center section), and the date on the second line (center section).

```
<headerFooter>
  <oddHeader>&amp;CCenter &amp;"-,Bold"Bold
    &amp;"-,Regular"Header_x000A_&amp;D</oddHeader>
</headerFooter>
```

end example]

General Rules:

There is no required order in which these codes need to appear.

The first occurrence of the following codes turns the formatting ON, the second occurrence turns it OFF again:

- strikethrough
- superscript

- subscript

Superscript and subscript cannot both be ON at same time. Whichever comes first wins and the other is ignored, while the first is ON.

&L - code for "left section" (there are three header / footer locations, "left", "center", and "right"). When two or more occurrences of this section marker exist, the contents from all markers are concatenated, in the order of appearance, and placed into the left section.

&P - code for "current page #"

&N - code for "total pages"

&*font size* - code for "text font size", where *font size* is a font size in points.

&K - code for "text font color"

RGB Color is specified as RRGGBB

Theme Color is specified as TTSNN where TT is the theme color Id, S is either "+" or "-" of the tint/shade value, NN is the tint/shade value.

&S - code for "text strikethrough" on / off

&X - code for "text super script" on / off

&Y - code for "text subscript" on / off

&C - code for "center section". When two or more occurrences of this section marker exist, the contents from all markers are concatenated, in the order of appearance, and placed into the center section.

&D - code for "date"

&T - code for "time"

&G - code for "picture as background"

&U - code for "text single underline"

&E - code for "double underline"

&R - code for "right section". When two or more occurrences of this section marker exist, the contents from all markers are concatenated, in the order of appearance, and placed into the right section.

&Z - code for "this workbook's file path"

&F - code for "this workbook's file name"

&A - code for "sheet tab name"

&+ - code for add to page #.

&- - code for subtract from page #.

&"*font name*,*font type*" - code for "text font name" and "text font type", where *font name* and *font type* are strings specifying the name and type of the font, separated by a comma. When a hyphen appears in *font name*, it means "none specified". Both of *font name* and *font type* can be localized values. Although ISO/IEC 14496-22 permits commas in font family/subfamily/full names, name and font type, the lexically first comma in the string is the one recognized as the separating comma.

&"-,Bold" - code for "bold font style"

&B - also means "bold font style".

&"-,Regular" - code for "regular font style"

&"-,Italic" - code for "italic font style"

&I - also means "italic font style"

&"-,Bold Italic" code for "bold italic font style"

&O - code for "outline style"

&H - code for "shadow style"

The possible values for this element are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (ST_Xstring) is located in §A.6.9. *end note*]

18.3.1.40 f (Formula)

Formula for the cell. The formula expression is contained in the character node of this element.

[*Example*:

```
<f>SUM(C4:E4)</f>
```

end example]

The possible values for the t attribute are defined by the simple type ST_CellFormulaType, and are as follows:

Value	Description
array (Array formula)	Array formula. An array formula is a single formula, applied across a range of one or more cells. An array formula can return multiple results from one single calculation, the results spanning the cells in which it is contained (§18.17.2.7).
dataTable (Table	Data table. A *data table* is a range of cells that shows how changing certain values in one

Value	Description
formula)	or more formulas affects the results of those formulas. A data table provides a shortcut for calculating multiple versions in one operation, and a way to include the results of all of the different variations in a sheet. Both one- and two-input variable data tables can be created (see attribute dt2D).[*Example*: A one-input variable data table might be used to calculate how different interest rates affect a monthly mortgage payment, while a two-input variable data table might be used to calculate how different interest rates and loan terms will affect the mortgage payment. *end example*] In a one-input variable data table, values are listed either down a column (column-oriented) or across a row (row-oriented) (see attribute dtr). Formulas that are used in a one-input variable data table shall refer to an input cell (see attribute r1), the cell in which each input value from a data table is substituted. Any cell on a worksheet can be the input cell. Although the input cell does not need to be part of the data table, the formulas in data tables shall refer to that input cell. Two-input variable data tables use only one formula with two lists of input values. The formula shall refer to two input cells (see attributes r1 and r2). The top-left cell in the data table is called the *master cell*. [*Guidance*: It is recommended that Spreadsheet applications recalculate data tables whenever a worksheet is recalculated. *end guidance*]
normal (Normal formula)	Normal cell formula (§18.17).
shared (Shared formula)	Shared formula. If a cell contains the same formula as another cell, the “shared” value can be used for the t attribute and the si attribute can be used to refer to the cell containing the formula. Two formulas are considered to be the same when their respective representations in R1C1-reference notation, are the same.

Attributes	Description
aca (Always Calculate Array)	Only applies to array formulas. `true` indicates that the entire array shall be calculated in full. If `false` the individual cells of the array shall be calculated as needed. The `aca` value shall be ignored unless the value of the corresponding t attribute is array. [*Note*: The primary case where an array formula must be calculated in part instead of in full is when some cells in the array depend on other cells that are semi-calculated, e.g., contains the function =RAND(). *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
bx (Assigns Value to Name)	Specifies that this formula assigns a value to a name. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ca (Calculate Cell)	Indicates that this formula needs to be recalculated the next time calculation is performed. [*Example*: This is always set on volatile functions, like =RAND(), and circular references. *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
del1 (Input 1 Deleted)	Whether the first input cell for data table has been deleted. Applies to data table formula only. Written on master cell of data table formula only. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
del2 (Input 2 Deleted)	Whether the second input cell for data table has been deleted. Applies to data table formula only. Written on master cell of data table formula only. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dt2D (Data Table 2-D)	Data table is two-dimentional. Only applies to the data tables function. Written on master cell of data table formula only. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dtr (Data Table Row)	`true` if one-dimentional data table is a row, otherwise it's a column. Only applies to the data tables function. Written on master cell of data table formula only. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
r1 (Data Table Cell 1)	First input cell for data table. Only applies to the data tables array function "TABLE()". Written on master cell of data table formula only. The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
r2 (Input Cell 2)	Second input cell for data table when dt2D is 1. Only applies to the data tables array function "TABLE()".Written on master cell of data table formula only. The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
ref (Range of Cells)	Range of cells which the formula applies to. Only required for shared formula, array formula or data table. Only written on the master formula, not subsequent formulas belonging to the same shared group, array, or data table.

Attributes	Description
	The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
si (Shared Group Index)	Optional attribute to optimize load performance by sharing formulas. When a formula is a shared formula (t value is shared) then this value indicates the group to which this particular cell's formula belongs. The first formula in a group of shared formulas is saved in the f element. This is considered the 'master' formula cell. Subsequent cells sharing this formula need not have the formula written in their f element. Instead, the attribute si value for a particular cell is used to figure what the formula expression should be based on the cell's relative location to the master formula cell. A cell is shared only when si is used and t is shared. The formula expression for a cell that is specified to be part of a shared formula (and is not the master) shall be ignored, and the master formula shall override. If a master cell of a shared formula range specifies that a particular cell is part of the shared formula range, and that particular cell does not use the si and t attributes to indicate that it is shared, then the particular cell's formula shall override the shared master formula. If this cell occurs in the middle of a range of shared formula cells, the earlier and later formulas shall continue sharing the master formula, and the cell in question shall not share the formula of the master cell formula. Loading and handling of a cell and formula using an si attribute and whose t value is shared, located outside the range specified in the master cell associated with the si group, is implementation defined. Master cell references on the same sheet shall not overlap with each other. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
t (Formula Type)	Type of formula. The possible values for this attribute are defined by the ST_CellFormulaType simple type (§18.18.6).
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. 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_CellFormula) is located in §A.2. *end note*]

18.3.1.41 firstFooter (First Page Footer)

First page footer content. Only used when `headerFooter@differentFirst` is '1'. Corresponds to first printed page. [*Example*:The first logical page in the sheet can not be printed if the print area is specified to be a range such that it falls outside the first page's scope. *end example*]

See evenHeader (§18.3.1.39) description for full discussion of value content.

The possible values for this element are defined by the ST_Xstring simple type (§22.9.2.19).

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. 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 (ST_Xstring) is located in §A.6.9. *end note*]

18.3.1.42 firstHeader (First Page Header)

First page header content. Only used when `headerFooter@differentFirst` is 1. Corresponds to first printed page. [*Example*: The first logical page in the sheet can not be printed if the print area is specified to be a range such that it falls outside the first page's scope. *end example*]

See evenHeader (§18.3.1.39) description for full discussion of value content.

The possible values for this element are defined by the ST_Xstring simple type (§22.9.2.19).

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. 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 (ST_Xstring) is located in §A.6.9. *end note*]

18.3.1.43 formula (Formula)

The content of this element is a formula whose calculated value specifies the criteria for the conditional formatting rule.

The possible values for this element are defined by the ST_Formula simple type (§18.18.35).

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. 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 (ST_Formula) is located in §A.2. *end note*]

18.3.1.44 formula1 (Formula 1)

The first formula in the Data Validation dropdown. It is used as a bounds for 'between' and 'notBetween' relational operators, and the only formula used for other relational operators (equal, notEqual, lessThan, lessThanOrEqual, greaterThan, greaterThanOrEqual), or for custom or list type data validation. The content can be a formula or a constant or a list series (comma separated values).

The possible values for this element are defined by the ST_Formula simple type (§18.18.35).

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. 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 (ST_Formula) is located in §A.2. *end note*]

18.3.1.45 formula2 (Formula 2)

The second formula in the DataValidation dropdown. It is used as a bounds for 'between' and 'notBetween' relational operators only.

The possible values for this element are defined by the ST_Formula simple type (§18.18.35).

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. 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 (ST_Formula) is located in §A.2. *end note*]

18.3.1.46 headerFooter (Header Footer Settings)

Header and footer settings.

[*Example*:

This example demonstrates "Header" at the top and "Footer" at the bottom of a page.


```
<headerFooter>
  <oddHeader>&amp;CHeader</oddHeader>
  <oddFooter>&amp;CFooter</oddFooter>
</headerFooter>
```

end example]

The tokens in the header & footer elements can be localized. An application can decide which locales are supported. Even when a locale is not supported, the header and footer text shall be loaded, and only the formatting is discarded.

Attributes	Description
alignWithMargins (Align Margins)	Align header footer margins with page margins. When `true`, as left/right margins grow and shrink, the header and footer edges stay aligned with the margins. When `false`, headers and footers are aligned on the paper edges, regardless of margins. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
differentFirst (Different First Page)	Different first page header and footer. When `true` then firstHeader and firstFooter specify first page header and footer values. If `false` and firstHeader / firstFooter are present, they are ignored. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
differentOddEven (Different Odd Even Header Footer)	Different odd and even page headers and footers. When `true` then oddHeader / oddFooter and evenHeader / evenFooter specify page header and footer values for odd and even pages. If `false` then oddHeader / oddFooter is used, even when evenHeader / evenFooter are present. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
scaleWithDoc (Scale Header & Footer With Document)	Scale header and footer with document scaling. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_HeaderFooter) is located in §A.2. *end note*]

18.3.1.47 hyperlink (Hyperlink)

A single hyperlink

Attributes	Description
display (Display	Display string, if different from string in string table. This is a property on the hyperlink

Attributes	Description
String)	object, but does not need to appear in the spreadsheet application UI. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Relationship Id in this sheet's relationships part, expressing the target location of the resource. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
location (Location)	Location within target. If target is a workbook (or this workbook) this shall refer to a sheet and cell or a defined name. Can also be an HTML anchor if target is HTML file. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
ref (Reference)	Cell location of hyperlink on worksheet. The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
tooltip (Tool Tip)	This is additional text to help the user understand more about the hyperlink. [*Example*: This can be displayed as hover text when the mouse is over the link. *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Hyperlink) is located in §A.2. *end note*]

18.3.1.48 hyperlinks (Hyperlinks)

Collection of hyperlinks.

[*Example*:

This example shows a hyperlink in cell A11, with hover text displaying "Search Page". The relationship Id references a relationship from the sheet to the external target resource.

```
<hyperlinks>
  <hyperlink ref="A11" r:id="rId1" tooltip="Search Page"/>
</hyperlinks>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Hyperlinks) is located in §A.2. *end note*]

18.3.1.49 iconSet (Icon Set)

Describes an icon set conditional formatting rule.

[*Example*: This example demonstrates the "3Arrows" style of icons. The first icon in the set must be shown if the cell's value is less than the 33rd percentile. The second icon in the set must be shown if the cell's value is less than the 67th percentile, and greater than or equal to the 33rd percentile. The third icon in the set must be shown if the cell's value is greater than or equal to the 67th percentile.

```
<iconSet iconSet="3Arrows">
  <cfvo type="percentile" val="0"/>
  <cfvo type="percentile" val="33"/>
  <cfvo type="percentile" val="67"/>
</iconSet>
```

end example]

Attributes	Description
iconSet (Icon Set)	The icon set to display. The possible values for this attribute are defined by the ST_IconSetType simple type (§18.18.42).
percent (Percent)	Indicates whether the thresholds indicate percentile values, instead of number values. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
reverse (Reverse Icons)	If 1, reverses the default order of the icons in this icon set. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showValue (Show Value)	Indicates whether to show the values of the cells on which this icon set is applied. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_IconSet) is located in §A.2. *end note*]

18.3.1.50 ignoredError (Ignored Error)

A single ignored error for a range of cells.

A cell is considered to have an error condition when it meets one of the conditions specified in the attribute descriptions below. [*Example*: If a cell is formatted as text but contains a numeric value, this is considered to be a potential error because the number won't be treated as a number, for example, in calculations. *end example*]

This is simply a guess by the implementing application, and a recommendation to the user. Cells with the errors specified below might be deliberately defined as such. [*Example*: A cell formatted as text which contains numeric Postal Codes or Order numbers. It is useful to format these cells as text so that leading zeros remain as part of the value instead of being removed. *end example*]

An <ignoreError> element is not written in the file unless the user has specifically reviewed the error and decided to keep the cell state as it is, and no longer wishes to be alerted about it for this cell. This can be helpful for the application to decide which errors should be surfaced to the user vs kept quiet because the user doesn't want these to be surfaced (e.g., because they are legitimate cell states).

[*Example*: This example shows that cells A1 and B2 both contain numbers stored as text, and this error has been reviewed and specifically flagged to be no longer surfaced as an error to the user.

```
<ignoredErrors>
  <ignoredError sqref="A1 B2" numberStoredAsText="1"/>
</ignoredErrors>
```

end example]

More than one kind of error can exist on a cell. These flags are not mutually exclusive.

Attributes	Description
calculatedColumn (Calculated Column)	Ignore errors when cells contain a value different from a calculated column formula. In other words, for a calculated column, a cell in that column is considered to have an error if its formula is different from the calculated column formula, or doesn't contain a formula at all. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
emptyCellReference (Empty Cell Reference)	Ignore errors when formulas refer to empty cells. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
evalError (Evaluation Error)	Ignore errors when cells contain formulas that result in an error. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
formula (Formula)	Ignore errors when a formula in a region of your worksheet differs from other formulas in the same region. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
	datatype.
formulaRange (Formula Range)	Ignore errors when formulas omit certain cells in a region. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
listDataValidation (List Data Validation)	Ignore errors when a cell's value in a Table does not comply with the Data Validation rules specified. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
numberStoredAsText (Number Stored As Text)	Ignore errors when numbers are formatted as text or are preceded by an apostrophe. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sqref (Sequence of References)	Reference to a range of cells that have this ignored error. The possible values for this attribute are defined by the ST_Sqref simple type (§18.18.76).
twoDigitTextYear (Two Digit Text Year)	Ignore errors when formulas contain text formatted cells with years represented as 2 digits. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
unlockedFormula (Unlocked Formula)	Ignore errors when unlocked cells contain formulas. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_IgnoredError) is located in §A.2. *end note*]

18.3.1.51 ignoredErrors (Ignored Errors)

A collection of ignored errors, by cell range.

[*Note*: The W3C XML Schema definition of this element's content model (CT_IgnoredErrors) is located in §A.2. *end note*]

18.3.1.52 inputCells (Input Cells)

This collection describes each input cell for the scenario.

Attributes	Description
deleted (Deleted)	Input cell was deleted. This input cell shall be present in the file format, but shall not be presented to the user as part of the scenario inputs, nor run as part of the scenario. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
numFmtId (Number Format Id)	This number format Id is used only when displaying the scenario manager input UI, and is used to properly format for display the cached input values (see val attribute) for the scenario. The possible values for this attribute are defined by the ST_NumFmtId simple type (§18.18.47).
r (Reference)	Cell reference indicating the input cell address. The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
undone (Undone)	Cell's deletion was undone. When true the r (reference) value shall not adjust in response to the cell moving due to row / column insert or delete,or cell move. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
val (Value)	Value that should be used for the cell when this scenario is run. val does not need a corresponding data type, the value is put into the cell when the scenario is run. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_InputCells) is located in §A.2. *end note*]

18.3.1.53 is (Rich Text Inline)

This element allows for strings to be expressed directly in the cell definition instead of implementing the shared string table.

[*Example*:

```
<c r="A1">
  <is>
    <t>String</t>
  </is>
</c>
```


end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rst) is located in §A.2. *end note*]

18.3.1.54 mergeCell (Merged Cell)

A single merged cell

[*Note*: The W3C XML Schema definition of this element's content model (CT_MergeCell) is located in §A.2. *end note*]

18.3.1.55 mergeCells (Merge Cells)

This collection expresses all the merged cells in the sheet.

[*Example*:

This example shows that three ranges are merged. The formatting and content for the merged range is always stored in the top left cell.

```
<mergeCells>
  <mergeCell ref="C2:F2"/>
  <mergeCell ref="B19:C20"/>
  <mergeCell ref="E19:G19"/>
</mergeCells>
```

end example]

Attributes	Description
count (Count)	A count of merged cell collections. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MergeCells) is located in §A.2. *end note*]

18.3.1.56 objectPr (Embedded Object Properties)

This element specifies the visual, positional and cell linkage properties of an embedded object.

[*Example*: The following example demonstrates an embedded object that does not print and that is resized when the cells underlying it are resized:

```
<oleObjects>
  <oleObject … >
    <objectPr print="false" autoLine="false" r:id="rId5">
      <anchor sizeWithCells="true">
```



```
        <from> … </from>
        <to> … </to>
      </anchor>
    </objectPr>
  </oleObject>
</oleObjects>
```

end example]

Attributes	Description
altText (Alternative Text)	Specifies alternative text for the object, for use by assistive technologies or applications. [*Example*: `<objectPr altText="Alternate text" … />` *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
autoFill (Automatic Fill Flag)	Specifies whether the object's fill formatting is provided automatically by the application. [*Example*: `<objectPr autoFill="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoLine (Automatic Line Flag)	Specifies whether the object's line formatting is provided automatically by the application. [*Example*: `<objectPr autoLine="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoPict (Automatic Size Flag)	Specifies whether the object's size is formatted automatically by the application. [*Example*: `<objectPr autoPict="false" … />`

Attributes	Description
	end example] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dde (Dynamic Data Exchange Flag)	Specifies whether the object is a Dynamic Data Exchange link. [*Example*: `<objectPr dde="true" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
defaultSize (Default Size Flag)	Specifies whether the object is at its default size. [*Example*: `<objectPr defaultSize="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
disabled (Disabled Flag)	Specifies whether the object is allowed to run an attached macro. [*Example*: `<objectPr disabled="true" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
id (Relationship ID 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" />`

Attributes	Description
	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).
locked (Locked Flag)	Specifies that the object is locked when the sheet is protected. [*Example*: `<objectPr locked="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
macro (Custom Function)	Specifies the custom function associated with the object. [*Example*: A macro script, add-in function, and so on. *end example*] [*Example*: `<objectPr macro="Button1_Click()" … />` *end example*] The possible values for this attribute are defined by the ST_Formula simple type (§18.18.35).
print (Print Flag)	Specifies whether the object is printed when the document is printed. [*Example*: `<objectPr print="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
uiObject (UI Object Flag)	Specifies whether the object is a UI-only object. Applications should prevent UI-only objects from being selected and edited in their user interface. [*Example*: `<objectPr uiObject="true" … />` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ObjectPr) is located in §A.2. *end note*]

18.3.1.57 oddFooter (Odd Page Footer)

Odd page footer value. Corresponds to odd printed pages. [*Example*: Odd page(s) in the sheet can not be printed if the print area is specified to be a range such that it falls outside an odd page's scope. *end example*]

See evenHeader (§18.3.1.39) description for full discussion of value content.

The possible values for this element are defined by the ST_Xstring simple type (§22.9.2.19).

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. 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 (ST_Xstring) is located in §A.6.9. *end note*]

18.3.1.58 oddHeader (Odd Header)

Odd page header value. Corresponds to odd printed pages. [*Example*: Odd page(s) in the sheet can not be printed if the print area is specified to be a range such that it falls outside an odd page's scope. *end example*]

See evenHeader (§18.3.1.39) description for full discussion of value content.

The possible values for this element are defined by the ST_Xstring simple type (§22.9.2.19).

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. 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 (ST_Xstring) is located in §A.6.9. *end note*]

18.3.1.59 oleObject (Embedded Object)

Information for an individual embedded object.

Attributes	Description
autoLoad (Auto Load)	Specifies whether the host application for the embedded object shall be called to load the object data automatically when the parent workbook is opened. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dvAspect (Data or View Aspect)	Specifies the desired Data or View Aspect of the object when drawing or getting data The possible values for this attribute are defined by the ST_DvAspect simple type (§18.18.24).
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Relationship Id of the relationship pointing to the object persistence part. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
link (Embedded Object's Link Moniker)	The embedded object's link moniker. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
oleUpdate (Linked Embedded Object Update)	Indicates whether the linked object updates the cached data automatically or only when the container requests an update, only present if the embedded object is linked. The possible values for this attribute are defined by the ST_OleUpdate simple type (§18.18.49).

Attributes	Description
progId (Embedded Object ProgId)	ProgId of the embedded object. The possible values for this attribute are defined by the W3C XML Schema string datatype.
shapeId (Shape Id)	Id of the shape this object is associated with. Corresponds with the shape @id in the drawingML part. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_OleObject) is located in §A.2. *end note*]

18.3.1.60 oleObjects (Embedded Objects)

Embedded objects collection in this worksheet.

[*Example*:

This example shows two embedded objects.

```
<oleObjects>
  <oleObject progId="Word.Document.12" shapeId="1025" r:id="rId4"/>
  <oleObject progId="PowerPoint.Show.12" shapeId="1026" r:id="rId5"/>
</oleObjects>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_OleObjects) is located in §A.2. *end note*]

18.3.1.61 outlinePr (Outline Properties)

Outline properties of the worksheet.

[*Example*: This example indicates that when an outline is applied to data, formatting must be applied to the outline result.

```
<sheetPr>
  <outlinePr applyStyles="1"/>
</sheetPr>
```

end example]

Attributes	Description
applyStyles (Apply Styles in Outline)	Flag indicating whether to apply styles in an outline, when outline is applied. Outline styles are described in Styles (§18.8). The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showOutlineSymbols (Show Outline Symbols)	Flag indicating whether the sheet has outline symbols visible. This flag shall always be overridden by the showOutlineSymbols attribute on sheetView when there is a conflict. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
summaryBelow (Summary Below)	Flag indicating whether summary rows appear below detail in an outline, when applying an outline. When `true` a summary row is inserted below the detailed data being summarized and a new outline level is established on that row. When `false` a summary row is inserted above the detailed data being summarized and a new outline level is established on that row. Note that toggling this flag on existing outlines requires an update to cell table, specifically, putting the summary functions in the proper rows, and flagging these rows as new outline levels, and possibly resetting their collapsed state. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
summaryRight (Summary Right)	Flag indicating whether summary columns appear to the right of detail in an outline, when applying an outline. When `true` a summary column is inserted to the right of the detailed data being summarized and a new outline level is established on that column. When `false` a summary column is inserted to the left of the detailed data being summarized and a new outline level is established on that column. Note that toggling this flag on existing outlines requires an update to cell table, specifically, putting the summary functions in the proper columns, and flagging these columns as new outline levels, and possibly resetting their collapsed state. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_OutlinePr) is located in §A.2. *end note*]

18.3.1.62 pageMargins (Page Margins)

Page margins for a sheet or a custom sheet view.

[*Example*:

```
<pageMargins left="0.7" right="0.7" top="0.75" bottom="0.75" header="0.3"
  footer="0.3"/>
```

end example]

Attributes	Description
bottom (Bottom Page Margin)	Bottom Page Margin in inches. The possible values for this attribute are defined by the W3C XML Schema double datatype.
footer (Footer Page Margin)	Footer Page Margin in inches. The possible values for this attribute are defined by the W3C XML Schema double datatype.
header (Header Page Margin)	Header Page Margin in inches. The possible values for this attribute are defined by the W3C XML Schema double datatype.
left (Left Page Margin)	Left Page Margin in inches. The possible values for this attribute are defined by the W3C XML Schema double datatype.
right (Right Page Margin)	Right page margin in inches. The possible values for this attribute are defined by the W3C XML Schema double datatype.
top (Top Page Margin)	Top Page Margin in inches. The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageMargins) is located in §A.2. *end note*]

18.3.1.63 pageSetup (Page Setup Settings)

Page setup settings for the worksheet.

[*Example*: The following example shows the pageSetup element for ISO A0 paper, printed in black and white, with graphics:

```
<pageSetup blackAndWhite="true" draft="false" paperHeight="1189mm"
paperWidth="841mm" />
```

end example]

Attributes	Description
blackAndWhite (Black And White)	Print black and white. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
cellComments (Print Cell Comments)	This attribute specifies how to print cell comments. The possible values for this attribute are defined by the ST_CellComments simple type (§18.18.5).
copies (Number Of Copies)	Number of copies to print. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
draft (Draft)	Print without graphics. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
errors (Print Error Handling)	Specifies how to print cell values for cells with errors. The possible values for this attribute are defined by the ST_PrintError simple type (§18.18.60).
firstPageNumber (First Page Number)	Page number for first printed page. If no value is specified, then 'automatic' is assumed. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
fitToHeight (Fit To Height)	Number of vertical pages to fit on. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
fitToWidth (Fit To Width)	Number of horizontal pages to fit on. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
horizontalDpi (Horizontal DPI)	Horizontal print resolution of the device. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
id (Id)	Relationship Id of the devMode printer settings part.

Attributes	Description
Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
orientation (Orientation)	Orientation of the page. The possible values for this attribute are defined by the ST_Orientation simple type (§18.18.50).
pageOrder (Page Order)	Order of printed pages. The possible values for this attribute are defined by the ST_PageOrder simple type (§18.18.51).
paperHeight (Paper Height)	Height of custom paper as a number followed by a unit identifier. [*Example*: `297mm`, `11in` *end example*] When paperHeight and paperWidth are specified, paperSize shall be ignored. The possible values for this attribute are defined by the ST_PositiveUniversalMeasure simple type (§22.9.2.12).
paperSize (Paper Size)	Paper size 1 = Letter paper (8.5 in. by 11 in.) 2 = Letter small paper (8.5 in. by 11 in.) 3 = Tabloid paper (11 in. by 17 in.) 4 = Ledger paper (17 in. by 11 in.) 5 = Legal paper (8.5 in. by 14 in.) 6 = Statement paper (5.5 in. by 8.5 in.) 7 = Executive paper (7.25 in. by 10.5 in.) 8 = A3 paper (297 mm by 420 mm) 9 = A4 paper (210 mm by 297 mm) 10 = A4 small paper (210 mm by 297 mm) 11 = A5 paper (148 mm by 210 mm) 12 = B4 paper (250 mm by 353 mm) 13 = B5 paper (176 mm by 250 mm) 14 = Folio paper (8.5 in. by 13 in.) 15 = Quarto paper (215 mm by 275 mm) 16 = Standard paper (10 in. by 14 in.) 17 = Standard paper (11 in. by 17 in.) 18 = Note paper (8.5 in. by 11 in.) 19 = #9 envelope (3.875 in. by 8.875 in.) 20 = #10 envelope (4.125 in. by 9.5 in.) 21 = #11 envelope (4.5 in. by 10.375 in.) 22 = #12 envelope (4.75 in. by 11 in.) 23 = #14 envelope (5 in. by 11.5 in.) 24 = C paper (17 in. by 22 in.)

Attributes	Description
	25 = D paper (22 in. by 34 in.) 26 = E paper (34 in. by 44 in.) 27 = DL envelope (110 mm by 220 mm) 28 = C5 envelope (162 mm by 229 mm) 29 = C3 envelope (324 mm by 458 mm) 30 = C4 envelope (229 mm by 324 mm) 31 = C6 envelope (114 mm by 162 mm) 32 = C65 envelope (114 mm by 229 mm) 33 = B4 envelope (250 mm by 353 mm) 34 = B5 envelope (176 mm by 250 mm) 35 = B6 envelope (176 mm by 125 mm) 36 = Italy envelope (110 mm by 230 mm) 37 = Monarch envelope (3.875 in. by 7.5 in.). 38 = 6 3/4 envelope (3.625 in. by 6.5 in.) 39 = US standard fanfold (14.875 in. by 11 in.) 40 = German standard fanfold (8.5 in. by 12 in.) 41 = German legal fanfold (8.5 in. by 13 in.) 42 = ISO B4 (250 mm by 353 mm) 43 = Japanese double postcard (200 mm by 148 mm) 44 = Standard paper (9 in. by 11 in.) 45 = Standard paper (10 in. by 11 in.) 46 = Standard paper (15 in. by 11 in.) 47 = Invite envelope (220 mm by 220 mm) 50 = Letter extra paper (9.275 in. by 12 in.) 51 = Legal extra paper (9.275 in. by 15 in.) 52 = Tabloid extra paper (11.69 in. by 18 in.) 53 = A4 extra paper (236 mm by 322 mm) 54 = Letter transverse paper (8.275 in. by 11 in.) 55 = A4 transverse paper (210 mm by 297 mm) 56 = Letter extra transverse paper (9.275 in. by 12 in.) 57 = SuperA/SuperA/A4 paper (227 mm by 356 mm) 58 = SuperB/SuperB/A3 paper (305 mm by 487 mm) 59 = Letter plus paper (8.5 in. by 12.69 in.) 60 = A4 plus paper (210 mm by 330 mm) 61 = A5 transverse paper (148 mm by 210 mm) 62 = JIS B5 transverse paper (182 mm by 257 mm) 63 = A3 extra paper (322 mm by 445 mm) 64 = A5 extra paper (174 mm by 235 mm) 65 = ISO B5 extra paper (201 mm by 276 mm) 66 = A2 paper (420 mm by 594 mm) 67 = A3 transverse paper (297 mm by 420 mm) 68 = A3 extra transverse paper (322 mm by 445 mm) 69 = Japanese Double Postcard (200 mm x 148 mm) 70 = A6 (105 mm x 148 mm) 71 = Japanese Envelope Kaku #2 72 = Japanese Envelope Kaku #3

Attributes	Description
	73 = Japanese Envelope Chou #3 74 = Japanese Envelope Chou #4 75 = Letter Rotated (11in x 8 1/2 11 in) 76 = A3 Rotated (420 mm x 297 mm) 77 = A4 Rotated (297 mm x 210 mm) 78 = A5 Rotated (210 mm x 148 mm) 79 = B4 (JIS) Rotated (364 mm x 257 mm) 80 = B5 (JIS) Rotated (257 mm x 182 mm) 81 = Japanese Postcard Rotated (148 mm x 100 mm) 82 = Double Japanese Postcard Rotated (148 mm x 200 mm) 83 = A6 Rotated (148 mm x 105 mm) 84 = Japanese Envelope Kaku #2 Rotated 85 = Japanese Envelope Kaku #3 Rotated 86 = Japanese Envelope Chou #3 Rotated 87 = Japanese Envelope Chou #4 Rotated 88 = B6 (JIS) (128 mm x 182 mm) 89 = B6 (JIS) Rotated (182 mm x 128 mm) 90 = (12 in x 11 in) 91 = Japanese Envelope You #4 92 = Japanese Envelope You #4 Rotated 93 = PRC 16K (146 mm x 215 mm) 94 = PRC 32K (97 mm x 151 mm) 95 = PRC 32K(Big) (97 mm x 151 mm) 96 = PRC Envelope #1 (102 mm x 165 mm) 97 = PRC Envelope #2 (102 mm x 176 mm) 98 = PRC Envelope #3 (125 mm x 176 mm) 99 = PRC Envelope #4 (110 mm x 208 mm) 100 = PRC Envelope #5 (110 mm x 220 mm) 101 = PRC Envelope #6 (120 mm x 230 mm) 102 = PRC Envelope #7 (160 mm x 230 mm) 103 = PRC Envelope #8 (120 mm x 309 mm) 104 = PRC Envelope #9 (229 mm x 324 mm) 105 = PRC Envelope #10 (324 mm x 458 mm) 106 = PRC 16K Rotated 107 = PRC 32K Rotated 108 = PRC 32K(Big) Rotated 109 = PRC Envelope #1 Rotated (165 mm x 102 mm) 110 = PRC Envelope #2 Rotated (176 mm x 102 mm) 111 = PRC Envelope #3 Rotated (176 mm x 125 mm) 112 = PRC Envelope #4 Rotated (208 mm x 110 mm) 113 = PRC Envelope #5 Rotated (220 mm x 110 mm) 114 = PRC Envelope #6 Rotated (230 mm x 120 mm) 115 = PRC Envelope #7 Rotated (230 mm x 160 mm) 116 = PRC Envelope #8 Rotated (309 mm x 120 mm) 117 = PRC Envelope #9 Rotated (324 mm x 229 mm) 118 = PRC Envelope #10 Rotated (458 mm x 324 mm)

Attributes	Description
	When paperHeight and paperWidth are specified, paperSize should be ignored. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
paperWidth (Paper Width)	Width of custom paper as a number followed by a unit identifier. [*Example*: `21cm`, `8.5in` *end example*] When paperHeight and paperWidth are specified, paperSize shall be ignored. The possible values for this attribute are defined by the ST_PositiveUniversalMeasure simple type (§22.9.2.12).
scale (Print Scale)	Print scaling. This attribute is restricted to values ranging from 10 to 400. [*Example*: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% *end example*] This setting is overridden when fitToWidth and/or fitToHeight are in use. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
useFirstPageNumber (Use First Page Number)	Use firstPageNumber value for first page number, and do not auto number the pages. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
usePrinterDefaults (Use Printer Defaults)	Use the printer's defaults settings for page setup values and don't use the default values specified in the schema. [*Example*: If dpi is not present or specified in the XML, the application must not assume 600dpi as specified in the schema as a default and instead must let the printer specify the default dpi. *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
verticalDpi (Vertical DPI)	Vertical print resolution of the device. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageSetup) is located in §A.2. *end note*]

18.3.1.64 pageSetup (Chart Sheet Page Setup)

This element provides page setup properties for chart sheets.

[*Example*: The following example shows the pageSetup element for ISO A0 paper, printed in black and white, with graphics:

```
<pageSetup blackAndWhite="true" draft="false" paperHeight="1189mm"
paperWidth="841mm" />
```

end example]

Attributes	Description
blackAndWhite (Black And White)	Print black and white. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
copies (Number Of Copies)	Number of copies to print. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
draft (Draft)	Print draft quality. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
firstPageNumber (First Page Number)	Page number for first printed page. If no value is specified, then 'automatic' is assumed. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
horizontalDpi (Horizontal DPI)	Horizontal print resolution of the device. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
id (Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Relationship Id of the devMode printer settings part. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
orientation	Orientation of the page.

Attributes	Description
(Orientation)	The possible values for this attribute are defined by the ST_Orientation simple type (§18.18.50).
paperHeight (Paper Height)	Height of custom paper as a number followed by a unit identifier. [*Example*: `297mm`, `11in` *end example*] When paperHeight and paperWidth are specified, paperSize shall be ignored. The possible values for this attribute are defined by the ST_PositiveUniversalMeasure simple type (§22.9.2.12).
paperSize (Paper Size)	1 = Letter paper (8.5 in. by 11 in.) 2 = Letter small paper (8.5 in. by 11 in.) 3 = Tabloid paper (11 in. by 17 in.) 4 = Ledger paper (17 in. by 11 in.) 5 = Legal paper (8.5 in. by 14 in.) 6 = Statement paper (5.5 in. by 8.5 in.) 7 = Executive paper (7.25 in. by 10.5 in.) 8 = A3 paper (297 mm by 420 mm) 9 = A4 paper (210 mm by 297 mm) 10 = A4 small paper (210 mm by 297 mm) 11 = A5 paper (148 mm by 210 mm) 12 = B4 paper (250 mm by 353 mm) 13 = B5 paper (176 mm by 250 mm) 14 = Folio paper (8.5 in. by 13 in.) 15 = Quarto paper (215 mm by 275 mm) 16 = Standard paper (10 in. by 14 in.) 17 = Standard paper (11 in. by 17 in.) 18 = Note paper (8.5 in. by 11 in.) 19 = #9 envelope (3.875 in. by 8.875 in.) 20 = #10 envelope (4.125 in. by 9.5 in.) 21 = #11 envelope (4.5 in. by 10.375 in.) 22 = #12 envelope (4.75 in. by 11 in.) 23 = #14 envelope (5 in. by 11.5 in.) 24 = C paper (17 in. by 22 in.) 25 = D paper (22 in. by 34 in.) 26 = E paper (34 in. by 44 in.) 27 = DL envelope (110 mm by 220 mm) 28 = C5 envelope (162 mm by 229 mm) 29 = C3 envelope (324 mm by 458 mm) 30 = C4 envelope (229 mm by 324 mm) 31 = C6 envelope (114 mm by 162 mm) 32 = C65 envelope (114 mm by 229 mm) 33 = B4 envelope (250 mm by 353 mm) 34 = B5 envelope (176 mm by 250 mm) 35 = B6 envelope (176 mm by 125 mm)

Attributes	Description
	36 = Italy envelope (110 mm by 230 mm) 37 = Monarch envelope (3.875 in. by 7.5 in.). 38 = 6 3/4 envelope (3.625 in. by 6.5 in.) 39 = US standard fanfold (14.875 in. by 11 in.) 40 = German standard fanfold (8.5 in. by 12 in.) 41 = German legal fanfold (8.5 in. by 13 in.) 42 = ISO B4 (250 mm by 353 mm) 43 = Japanese double postcard (200 mm by 148 mm) 44 = Standard paper (9 in. by 11 in.) 45 = Standard paper (10 in. by 11 in.) 46 = Standard paper (15 in. by 11 in.) 47 = Invite envelope (220 mm by 220 mm) 50 = Letter extra paper (9.275 in. by 12 in.) 51 = Legal extra paper (9.275 in. by 15 in.) 52 = Tabloid extra paper (11.69 in. by 18 in.) 53 = A4 extra paper (236 mm by 322 mm) 54 = Letter transverse paper (8.275 in. by 11 in.) 55 = A4 transverse paper (210 mm by 297 mm) 56 = Letter extra transverse paper (9.275 in. by 12 in.) 57 = SuperA/SuperA/A4 paper (227 mm by 356 mm) 58 = SuperB/SuperB/A3 paper (305 mm by 487 mm) 59 = Letter plus paper (8.5 in. by 12.69 in.) 60 = A4 plus paper (210 mm by 330 mm) 61 = A5 transverse paper (148 mm by 210 mm) 62 = JIS B5 transverse paper (182 mm by 257 mm) 63 = A3 extra paper (322 mm by 445 mm) 64 = A5 extra paper (174 mm by 235 mm) 65 = ISO B5 extra paper (201 mm by 276 mm) 66 = A2 paper (420 mm by 594 mm) 67 = A3 transverse paper (297 mm by 420 mm) 68 = A3 extra transverse paper (322 mm by 445 mm) When paperHeight and paperWidth are specified, paperSize should be ignored. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
paperWidth (Paper Width)	Width of custom paper as a number followed by a unit identifier. [*Example*: `21cm`, `8.5in` *end example*] When paperHeight and paperWidth are specified, paperSize shall be ignored. The possible values for this attribute are defined by the ST_PositiveUniversalMeasure simple type (§22.9.2.12).
useFirstPageNumber (Use First Page	Use firstPageNumber value for first page number, and do not auto number the pages.

Attributes	Description
Number)	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
usePrinterDefaults (Use Printer Defaults)	Use the printer's defaults settings for page setup values and don't use the default values specified in the schema. [*Example*: If dpi is not present or specified in the XML, the application must not assume 600dpi as specified in the schema as a default and instead must let the printer specify the default dpi. *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
verticalDpi (Vertical DPI)	Vertical print resolution of the device. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CsPageSetup) is located in §A.2. *end note*]

18.3.1.65 pageSetUpPr (Page Setup Properties)

Page setup properties of the worksheet

Attributes	Description
autoPageBreaks (Show Auto Page Breaks)	Flag indicating whether the sheet displays Automatic Page Breaks. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
fitToPage (Fit To Page)	Flag indicating whether the Fit to Page print option is enabled. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageSetUpPr) is located in §A.2. *end note*]

18.3.1.66 pane (View Pane)

Worksheet view pane

Attributes	Description
activePane (Active Pane)	The pane that is active. The possible values for this attribute are defined by the ST_Pane simple type (§18.18.52).

Attributes	Description
state (Split State)	Indicates whether the pane has horizontal / vertical splits, and whether those splits are frozen. The possible values for this attribute are defined by the ST_PaneState simple type (§18.18.53).
topLeftCell (Top Left Visible Cell)	Location of the top left visible cell in the bottom right pane (when in Left-To-Right mode). The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
xSplit (Horizontal Split Position)	Horizontal position of the split, in 1/20th of a point; 0 (zero) if none. If the pane is frozen, this value indicates the number of columns visible in the top pane. The possible values for this attribute are defined by the W3C XML Schema double datatype.
ySplit (Vertical Split Position)	Vertical position of the split, in 1/20th of a point; 0 (zero) if none. If the pane is frozen, this value indicates the number of rows visible in the left pane. The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Pane) is located in §A.2. *end note*]

18.3.1.67 picture (Background Image)

Background sheet image.

[*Example*:

```
<picture r:id="rId1"/>
```

end example]

Attributes	Description
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Relationship Id pointing to the image part. 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_SheetBackgroundPicture) is located in §A.2. *end note*]

18.3.1.68 pivotArea (Pivot Area)

Rule describing a PivotTable selection.

Attributes	Description
axis (Axis)	The region of the PivotTable to which this rule applies. The possible values for this attribute are defined by the ST_Axis simple type (§18.18.1).
cacheIndex (Cache Index)	Flag indicating whether any indexes refer to fields or items in the Pivot cache and not the view. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
collapsedLevelsAreSubtotals (Collapsed Levels Are Subtotals)	Flag indicating if collapsed levels/dimensions are considered subtotals. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dataOnly (Data Only)	Flag indicating whether only the data values (in the data area of the view) for an item selection are selected and does not include the item labels. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
field (Field Index)	Index of the field that this selection rule refers to. The possible values for this attribute are defined by the W3C XML Schema int datatype.
fieldPosition (Field Position)	Position of the field within the axis to which this rule applies. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
grandCol (Include Column Grand Total)	Flag indicating whether the column grand total is included. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
grandRow (Include Row Grand Total)	Flag indicating whether the row grand total is included. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
labelOnly (Labels Only)	Flag indicating whether only the item labels for an item selection are selected and does not include the data values (in the data area of the view). The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
offset (Offset Reference)	A Reference that specifies a subset of the selection area. Points are relative to the top left of the selection area.

Attributes	Description
	The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
outline (Outline)	Flag indicating whether the rule refers to an area that is in outline mode. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
type (Rule Type)	Indicates the type of selection rule. The possible values for this attribute are defined by the ST_PivotAreaType simple type (§18.18.58).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotArea) is located in §A.2. *end note*]

18.3.1.69 pivotSelection (PivotTable Selection)

A collection of PivotTable structure selections. A PivotTable structure selection is a way of specifying what cells in the PivotTable are selected. Instead of specifying cell addresses in a sqref, a particular area or structure within the PivotTable is specified. In this way there is semantic meaning regarding what is selected, rather than simply a list of cell or ranges contained in the selection. Typically fields on the row or column axis are selected.

[*Example*: For example, the innermost field (Product SubCategory) is selected in this PivotTable:

	A	B	C
1			
2		State	(All)
3		City	(All)
4			
5			Column Labels
6			⊟2001
7			⊟3
8			July
9		Row Labels	Sum of Sales Amount
10		⊟Bikes	209652.9046
11		⊟Mountain Bikes	64424.81
12		Mountain-100 Black, 38	3374.99
13		Mountain-100 Black, 42	3374.99
14		Mountain-100 Black, 44	13499.96
15		Mountain-100 Black, 48	3374.99
16		Mountain-100 Silver, 38	6799.98
17		Mountain-100 Silver, 42	6799.98
18		Mountain-100 Silver, 44	16999.95
19		Mountain-100 Silver, 48	10199.97
20		⊟Road Bikes	145228.0946
21		Road-150 Red, 44	25047.89
22		Road-150 Red, 48	42939.24
23		Road-150 Red, 52	21469.62
24		Road-150 Red, 56	25047.89
25		Road-150 Red, 62	28626.16
26		Road-650 Black, 44	699.0982
27		Road-650 Black, 52	
28		Road-650 Black, 62	699.0982
29		Road-650 Red, 44	699.0982
30		Road-650 Red, 48	
31		Road-650 Red, 52	
32		Road-650 Red, 58	
33		Road-650 Red, 60	
34		Grand Total	209652.9046

The corresponding pivotSelection XML should look like this:

```
<pivotSelection pane="bottomRight" showHeader="1" axis="axisRow" dimension="2"
  activeRow="11" activeCol="1" previousRow="11" previousCol="1" click="1"
  r:id="rId1">
```

```
    <pivotArea dataOnly="0" labelOnly="1" fieldPosition="0">
      <references count="1">
        <reference field="9" count="0"/>
      </references>
    </pivotArea>
  </pivotSelection>
```

axis indicates that this selection is on the row axis, dimension indicates the field level within the row axis that is selected (zero-based index), activeCol and activeRow respectively indicate where in the grid the selection is located, and reference field indicates to which particular field the selection corresponds.

end example]

Attributes	Description
activeCol (Active Column)	The column (zero-based) of active cell for structure selection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
activeRow (Active Row)	The row (zero-based) of active cell for structure selection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
axis (Axis)	Axis of the PivotTable on which this selection lies. The possible values for this attribute are defined by the ST_Axis simple type (§18.18.1).
click (Click Count)	Number of clicks for this structure selection. For some selection combinations, subsequent clicks on the same target area cycles the actual selection through some variances. Therefore number of clicks on the selection shall be recorded, if it is desirable to restore this state of the selection cycle on load. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
count (Selection Count)	Number of selections for the structure selection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
data (Data Selection)	Flag indicating whether the structure selection is for data only. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dimension (Dimension)	Indicates the field level within the axis that is selected (zero-based index). The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

Attributes	Description
extendable (Extendable)	Flag indicating whether the structure selection can have additional selections added to it. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Relationship Id pointing to the particular PivotTable Part corresponding to this selection. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
label (Label)	Flag indicating whether the structure selection is for labels only (e.g., a grand total row is selected). The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
max (Maximum)	The maximum line the structure selection contains. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
min (Minimum)	The minimum line the structure selection contains. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
pane (Pane)	The pane to which this PivotTable structure selection belongs. The possible values for this attribute are defined by the ST_Pane simple type (§18.18.52).
previousCol (Previous Column Selection)	1-based index to the column immediately left of the structure selection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
previousRow (Previous Row)	1-based index to the row immediately above the structure selection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
showHeader (Show Header)	Flag indicating whether selection toggle from data only to header only to both is enabled. `False` means disabled. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
start (Start)	The line the structure selection begins (zero-based). This is the line clicked to initiate the structure selection.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotSelection) is located in §A.2. *end note*]

18.3.1.70 printOptions (Print Options)

Print options for the sheet. Printer-specific settings are stored separately in the Printer Settings part as defined in §15.2.15.

Attributes	Description
gridLines (Print Grid Lines)	Used in conjunction with gridLinesSet. If both gridLines and gridlinesSet are `true`, then grid lines shall print. Otherwise, they shall not (i.e., one or both have `false` values). The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
gridLinesSet (Grid Lines Set)	Used in conjunction with gridLines. If both gridLines and gridLinesSet are `true`, then grid lines shall print. Otherwise, they shall not (i.e., one or both have `false` values). The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
headings (Print Headings)	Print row and column headings. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
horizontalCentered (Horizontal Centered)	Center on page horizontally when printing. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
verticalCentered (Vertical Centered)	Center on page vertically when printing. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PrintOptions) is located in §A.2. *end note*]

18.3.1.71 protectedRange (Protected Range)

A specified range to be protected. Ranges listed here are protected only when the sheet protection is ON and the cell is flagged as being locked. If no password is specified here, then read/write permissions are automatically given to all users, regardless of additional security descriptor information. In other words, the security descriptor information (specific types of access) at the user level is only applied if a password for this range is specified.

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.

When a password is specified, then users not listed specifically as having access should be prompted with a password. If that user supplies the correct password, then they can edit the range or cell in question. This protection is optional and can be ignored by applications that choose not to support this functionality.

<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><code>MD2</code></td><td>Specifies that the MD2 algorithm, as defined by RFC 1319, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
<tr><td><code>MD4</code></td><td>Specifies that the MD4 algorithm, as defined by RFC 1320, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
<tr><td><code>MD5</code></td><td>Specifies that the MD5 algorithm, as defined by RFC 1321, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</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.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</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-</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>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>
<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_Xstring simple type (§22.9.2.19).
</td></tr>
<tr><td>hashValue (Password Hash Value)</td><td>
Specifies the hash value for the password required to edit this range. 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>
</table>

Attributes	Description
name (Name)	Range title. This is used as a descriptor, not as a named range definition. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
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

Attributes	Description
	hashValue attribute. *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sqref (Sequence of References)	The range to be protected. The possible values for this attribute are defined by the ST_Sqref simple type (§18.18.76).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ProtectedRange) is located in §A.2. *end note*]

18.3.1.72 protectedRanges (Protected Ranges)

This collection specifies all protected ranges on this worksheet.

[*Example*:

This example demonstrates that A1:C5 have been protected, with no password specified.

```
<protectedRanges>
  <protectedRange sqref="A1:C5" name="Range1"/>
</protectedRanges>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_ProtectedRanges) is located in §A.2. *end note*]

18.3.1.73 row (Row)

The element expresses information about an entire row of a worksheet, and contains all cell definitions for a particular row in the worksheet.

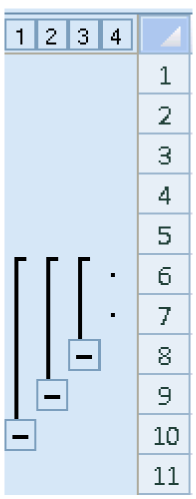
[*Example*:

This row expresses information about row 2 in the worksheet, and contains 3 cell definitions.

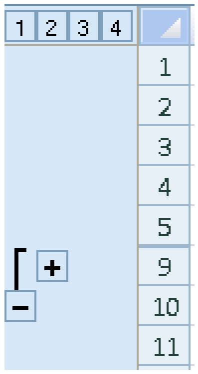
```
<row r="2" spans="2:12">
  <c r="C2" s="1">
    <f>PMT(B3/12,B4,-B5)</f>
    <v>672.68336574300008</v>
  </c>
  <c r="D2">
    <v>180</v>
  </c>
```

```
    <c r="E2">
      <v>360</v>
    </c>
  </row>
```

end example]

Attributes	Description
collapsed (Collapsed)	1 if the rows 1 level of outlining deeper than the current row are in the collapsed outline state. It means that the rows which are 1 outline level deeper (numerically higher value) than the current row are currently hidden due to a collapsed outline state. It is possible for collapsed to be `false` and yet still have the rows in question hidden. This can be achieved by having a lower outline level collapsed, thus hiding all the child rows. [*Example*: This example shows 3 levels of outlining:  In the XML must be: `<sheetData>` `  <row r="6" outlineLevel="3"/>` `  <row r="7" outlineLevel="3"/>` `  <row r="8" outlineLevel="2"/>` `  <row r="9" outlineLevel="1"/>` `</sheetData>` *end example*] [*Example*: This example shows the same outline feature, with the middle level collapsed:

Attributes	Description

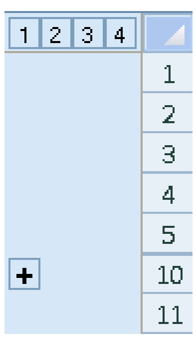


In the XML must be:

```
<sheetData>
  <row r="6" hidden="1" outlineLevel="3"/>
  <row r="7" hidden="1" outlineLevel="3"/>
  <row r="8" hidden="1" outlineLevel="2"/>
  <row r="9" outlineLevel="1" collapsed="1"/>
</sheetData>
```

end example]

[*Example*:
This example shows the same outline feature as above, where both the middle and lowest level are collapsed:



In the XML must be:

```
<sheetData>
  <row r="6" hidden="1" outlineLevel="3"/>
  <row r="7" hidden="1" outlineLevel="3"/>
  <row r="8" hidden="1" outlineLevel="2"/>
  <row r="9" hidden="1" outlineLevel="1" collapsed="1"/>
  <row r="10" collapsed="1"/>
</sheetData>
```

Note that in this case, if the lowest level were expanded, the middle level would remain

Attributes	Description
	collapsed due to collapsed being true on row 9. *end example*] See description of outlinePr element's summaryBelow and summaryRight attributes for detailed information. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
customFormat (Custom Format)	1 if the row style should be applied. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
customHeight (Custom Height)	1 if the row height has been manually set. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
hidden (Hidden)	1 if the row is hidden, e.g., due to a collapsed outline or by manually selecting and hiding a row. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ht (Row Height)	Row height measured in point size. There is no margin padding on row height. The possible values for this attribute are defined by the W3C XML Schema double datatype.
outlineLevel (Outline Level)	Outlining level of the row, when outlining is on. See description of outlinePr element's summaryBelow and summaryRight attributes for detailed information. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
ph (Show Phonetic)	1 if the row should show phonetic. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
r (Row Index)	Row index. Indicates to which row in the sheet this <row> definition corresponds. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
s (Style Index)	Index to style record for the row (only applied if customFormat attribute is '1') The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>spans (Spans)</td><td>

Optimization only, and not required. Specifies the range of non-empty columns (in the format X:Y) for the block of rows to which the current row belongs. To achieve the optimization, span attribute values in a single block should be the same.

There are 16 rows per block, beginning with the first row.

[*Note*: this is an optimization, and is purely optional. Different span values within the same row block is allowed. Not writing the span value at all is also allowed. *end note*]

Blank rows are not required to write out span values.

[*Example*: If cells F8, E9, and D10 have data in them and the rest of the sheet is empty, then for those three rows (8,9, and 10), the spans value should each be "4:6":

```
<sheetData>
  <row r="8" spans="4:6">
    <c r="F8">
      <v>1</v>
    </c>
  </row>
  <row r="9" spans="4:6">
    <c r="E9">
      <v>2</v>
    </c>
  </row>
  <row r="10" spans="4:6">
    <c r="D10">
      <v>3</v>
    </c>
  </row>
</sheetData>
```

If cells A1 and J10 have data in them and the rest of the sheet is empty, then the rows should be written like this:

```
<sheetData>
  <row r="1" spans="1:10">
    <c r="A1">
      <v>1</v>
    </c>
  </row>
  <row r="10" spans="1:10">
    <c r="J10">
      <v>2</v>
    </c>
  </row>
</sheetData>
```

</td></tr>
</table>

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_CellSpans simple type (§18.18.9).
thickBot (Thick Bottom)	1 if any cell in the row has a medium or thick bottom border, or if any cell in the row directly below the current row has a thick top border. When `true` and customHeight is `false`, this flag means that the row height has been adjusted higher by .75 points of the normal style font height. This also means that if the row no longer contains these borders, then the height is automatically re-adjusted down. This adjustment is in addition to any adjustment of height due to thickTop. Medium borders are these enumeration values from the Styles Part: • mediumDashDotDot • slantDashDot • mediumDashDot • mediumDashed • medium • • Thick borders are these enumeration values from the Styles Part: • thick • double The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
thickTop (Thick Top Border)	`True` if the row has a medium or thick top border, or if any cell in the row directly above the current row has a thick bottom border. When `true` and customHeight is `false`, this flag means that the row height has been adjusted higher by .75 points of the normal style font height. This also means that if the row no longer contains these borders, then the height is automatically re-adjusted down. This adjustment is in addition to any adjustment of height due to thickBot. Medium borders are these enumeration values from the Styles Part: • mediumDashDotDot • slantDashDot • mediumDashDot • mediumDashed • medium Thick borders are these enumeration values from the Styles Part: • thick • double

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Row) is located in §A.2. *end note*]

18.3.1.74 rowBreaks (Horizontal Page Breaks (Row))

Horizontal page break information used for print layout view, page layout view, drawing print breaks in normal view, and for printing the worksheet.

[*Example*: This example shows a break inserted at cell B25:

```
<rowBreaks count="1" manualBreakCount="1">
  <brk id="24" max="16383" man="1"/>
</rowBreaks>
```

end example]

Attributes	Description
count (Page Break Count)	Number of breaks in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
manualBreakCount (Manual Break Count)	Number of manual breaks in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageBreak) is located in §A.2. *end note*]

18.3.1.75 scenario (Scenario)

An individual scenario description. [*Note*: See parent element for an example. *end note*]

Attributes	Description
comment (Scenario Comment)	Comment for this scenario, rich text not supported. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
count (Changing Cell Count)	Number of input cells. The possible values for this attribute are defined by the W3C XML Schema unsignedInt

Attributes	Description
	datatype.
hidden (Hidden Scenario)	Scenario is hidden when the sheet is protected and 'edit scenarios' is not enabled in sheet protection options. If the scenario is marked as hidden but sheet protection options specify to allow editing scenarios, then the scenario shall not be hidden. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
locked (Scenario Locked)	Scenario is locked for editing when the sheet is protected. If sheet is protected and "edit scenarios" is enabled, then this setting is ignored. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
name (Scenario Name)	Scenario's name (user input). Shall be unique for the worksheet The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
user (User Name)	Name of user who last changed the scenario. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Scenario) is located in §A.2. *end note*]

18.3.1.76 scenarios (Scenarios)

A collection of Scenarios. A scenario is a named what-if model that includes variable cells linked together by one or more formulas.

[*Example*: For example, you might want to compare best-case and worst-case scenarios for sales in a coffee shop, based on the number of cups of coffee sold in a week.

```
<scenarios current="1" show="0" sqref="G4 G6 G7 G8">
  <scenario name="Best Case" locked="1" count="3" user="anonymous"
    comment="Created on 6/9/2006_x000a_Modified on 6/9/2006">
    <inputCells r="D5" val="151" numFmtId="37"/>
    <inputCells r="D9" val="226"/>
    <inputCells r="D13" val="126"/>
  </scenario>
```

```
    <scenario name="Worst Case" locked="1" count="3" user="anonymous"
      comment="Created on 6/9/2006">
      <inputCells r="D5" val="50" numFmtId="37"/>
      <inputCells r="D9" val="40"/>
      <inputCells r="D13" val="30"/>
    </scenario>
  </scenarios>
```

end example]

Attributes	Description
current (Current Scenario)	Zero-based index to current scenario selected. Can correspond to selection UI. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
show (Last Shown Scenario)	Zero-based index to last shown scenario. Indicates which scenario was last selected by the user to be run/shown. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sqref (Sequence of References)	Range or sequence of cells used for scenario results summary. The possible values for this attribute are defined by the ST_Sqref simple type (§18.18.76).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Scenarios) is located in §A.2. *end note*]

18.3.1.77 securityDescriptor (Security Descriptor)

Optional setting to specify the relative security descriptor. The security descriptor defines user accounts who may edit this range without providing a password to access the range.

[*Note*: The format of a securityDescriptor is application defined; however, it is recommended that the following format be used for interoperability between implementations: `username@domain`. This format follows the form of `addr-spec` as defined in RFC 822, Standard for ARPA Internet Text Messages. *end note*]

[*Example*: This example demonstrates two user accounts in the security descriptor attribute:

```
  <protectedRanges>
    <protectedRange sqref="A1:C5" name="Range1">
      <securityDescriptor>user1@iso.org</securityDescriptor>
      <securityDescriptor>user2@iso.org</securityDescriptor>
    </protectedRange>
  </protectedRanges>
```

end example]

The possible values for this element are defined by the W3C XML Schema string datatype.

18.3.1.78 selection (Selection)

Worksheet view selection.

Attributes	Description
activeCell (Active Cell Location)	Location of the active cell. The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
activeCellId (Active Cell Index)	0-based index of the range reference (in the array of references listed in sqref) containing the active cell. Only used when the selection in sqref is not contiguous. Therefore, this value needs to be aware of the order in which the range references are written in sqref. When this value is out of range then activeCell can be used. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
pane (Pane)	The pane to which this selection belongs. The possible values for this attribute are defined by the ST_Pane simple type (§18.18.52).
sqref (Sequence of References)	Range of the selection. Can be non-contiguous set of ranges. The possible values for this attribute are defined by the ST_Sqref simple type (§18.18.76).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Selection) is located in §A.2. *end note*]

18.3.1.79 sheetCalcPr (Sheet Calculation Properties)

This element contains calculation properties for the worksheet.

Attributes	Description
fullCalcOnLoad (Full Calculation On Load)	Indicates whether the application should do a full calculate on load due to contents on this sheet. After load and successful calc, the application shall set this value to false. Set this to true when the application should calculate the workbook on load. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SheetCalcPr) is located in §A.2. *end note*]

18.3.1.80 sheetData (Sheet Data)

This collection represents the cell table itself. This collection expresses information about each cell, grouped together by rows in the worksheet.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SheetData) is located in §A.2. *end note*]

18.3.1.81 sheetFormatPr (Sheet Format Properties)

Sheet formatting properties.

Attributes	Description
baseColWidth (Base Column Width)	Specifies the number of characters of the maximum digit width of the normal style's font. This value does not include margin padding or extra padding for gridlines. It is only the number of characters. See defaultColWidth description in this section for details on calculating this value. See the col element description, particularly the width attribute description, for more information on what is meant by "maximum digit width". The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
customHeight (Custom Height)	'True' if defaultRowHeight value has been manually set, or is different from the default value. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
defaultColWidth (Default Column Width)	Default column width measured as the number of characters of the maximum digit width of the normal style's font. If the user has not set this manually, then it can be calculated: defaultColWidth = baseColumnWidth + {margin padding (2 pixels on each side, totalling 4 pixels)} + {gridline (1pixel)} If the user has set this manually, then there is no calculation, and simply a value is specified. The possible values for this attribute are defined by the W3C XML Schema double datatype.
defaultRowHeight (Default Row Height)	Default row height measured in point size. Optimization so we don't have to write the height on all rows. This can be written out if most rows have custom height, to achieve the optimization.

Attributes	Description
	When the row height of all rows in a sheet is the default value, then that value is written here, and customHeight is not set. If a few rows have a different height, that information is written directly on each row. However, if most or all of the rows in the sheet have the same height, but that height isn't the default height, then that height value should be written here (as an optimization), and the customHeight flag should also be set. In this case, all rows having this height do not need to express the height, only rows whose height differs from this value need to be explicitly expressed. The possible values for this attribute are defined by the W3C XML Schema double datatype.
outlineLevelCol (Column Outline Level)	Highest number of outline levels for columns in this sheet. These values shall be in synch with the actual sheet outline levels. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
outlineLevelRow (Maximum Outline Row)	Highest number of outline level for rows in this sheet. These values shall be in synch with the actual sheet outline levels. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
thickBottom (Thick Bottom Border)	'True' if rows have a thick bottom border by default. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
thickTop (Thick Top Border)	'True' if rows have a thick top border by default. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
zeroHeight (Hidden By Default)	'True' if rows are hidden by default. This setting is an optimization used when most rows of the sheet are hidden. In this case, instead of writing out every row and specifying hidden, it is much shorter to only write out the rows that are not hidden, and specify here that rows are hidden by default, and only not hidden if specified. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SheetFormatPr) is located in §A.2. *end note*]

18.3.1.82 sheetPr (Sheet Properties)

Sheet-level properties.

Attributes	Description
codeName (Code Name)	Specifies a stable name of the sheet, which should not change over time, and does not change from user input. This name should be used by code to reference a particular sheet. The possible values for this attribute are defined by the W3C XML Schema string datatype.
enableFormatConditionsCalculation (Enable Conditional Formatting Calculations)	Flag indicating whether the conditional formatting calculations shall be evaluated. If set to `false`, then the min/max values of color scales or databars or threshold values in Top N rules shall not be updated. Essentially the conditional formatting "calc" is off. This is useful when conditional formats are being set programmatically at runtime, recalculation of the conditional formatting does not need to be done until the program execution has finished setting all the conditional formatting properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
filterMode (Filter Mode)	Flag indicating whether the worksheet has one or more autofilters or advanced filters on. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
published (Published)	Flag indicating whether the worksheet is published. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
syncHorizontal (Synch Horizontal)	Flag indicating whether this worksheet is horizontally synced to the synchRef anchor point. When true and scroll location is missing from the window properties, the window view shall be scrolled to the horizontal (row) aspect of the synchRef value. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
syncRef (Synch Reference)	Anchor point for worksheet's window. The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
syncVertical (Synch Vertical)	Flag indicating whether this worksheet is vertically synced to the synchRef anchor point. When true and scroll location is missing from the window properties, the window view shall be scrolled to the vertical (column) aspect of the synchRef value. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
transitionEntry (Transition Formula Entry)	Flag indicating whether the Transition Formula Entry (Lotus compatibility) option is enabled. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
transitionEvaluation (Transition Formula Evaluation)	Flag indicating whether the Transition Formula Evaluation (Lotus compatibility) option is enabled. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SheetPr) is located in §A.2. *end note*]

18.3.1.83 sheetPr (Chart Sheet Properties)

This element specifies chart sheet properties.

Attributes	Description
codeName (Code Name)	Specifies a stable name of the sheet, which should not change over time, and does not change from user input. This name should be used by code to reference a particular sheet. The possible values for this attribute are defined by the W3C XML Schema string datatype.
published (Published)	Flag indicating whether the chart sheet is published. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ChartsheetPr) is located in §A.2. *end note*]

18.3.1.84 sheetProtection (Chart Sheet Protection)

This collection expresses the chart sheet protection options to enforce when the chart sheet is protected.

<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.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known</td></tr>
</table></td></tr>
</table>

Attributes	Description

	breaks. *end note*]
`MD4`	Specifies that the MD4 algorithm, as defined by RFC 1320, shall be used. [*Note*: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. *end note*]
`MD5`	Specifies that the MD5 algorithm, as defined by RFC 1321, shall be used. [*Note*: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. *end note*]
`RIPEMD-128`	Specifies that the RIPEMD-128 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used. [*Note*: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. *end note*]
`RIPEMD-160`	Specifies that the RIPEMD-160 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.
`SHA-1`	Specifies that the SHA-1 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.
`SHA-256`	Specifies that the SHA-256 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.
`SHA-384`	Specifies that the SHA-384 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.
`SHA-512`	Specifies that the SHA-512 algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.
`WHIRLPOOL`	Specifies that the WHIRLPOOL algorithm, as defined by ISO/IEC 10118-3:2004 shall be used.

[*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*]

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
content (Contents)	When true prevents users from making changes to items that are part of the chart, such as data series, axes, and legends. The chart continues to reflect changes made to its source data. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
hashValue (Password Hash Value)	Specifies the hash value for the password required to edit this chartsheet. 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 shall 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.
objects (Objects Locked)	When true prevents users from making changes to graphic objects— including shapes, text boxes, and controls— unless you unlock the specific objects before you protect the chart sheet. The possible values for this attribute are defined by the W3C XML Schema boolean 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

Attributes	Description
	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 hashValue attribute. *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ChartsheetProtection) is located in §A.2. *end note*]

18.3.1.85 sheetProtection (Sheet Protection Options)

This collection expresses the sheet protection options to enforce when the sheet is protected.

[*Example*:

This example demonstrates that the sheet is protected, objects and scenarios can be edited, cell formatting is allowed,and selection of locked cells is not allowed:

```
<sheetProtection sheet="1" objects="1" scenarios="1" formatCells="0"
  selectLockedCells="1"/>
```

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.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
<tr><td>MD4</td><td>Specifies that the MD4 algorithm, as defined by RFC 1320, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
<tr><td>MD5</td><td>Specifies that the MD5 algorithm, as defined by RFC 1321, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</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.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</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>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<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>
<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_Xstring simple type (§22.9.2.19).</td></tr>
<tr><td>autoFilter (AutoFilter Locked)</td><td>If 1 or true then AutoFilters should not be allowed to operate when the sheet is protected.

If 0 or false then AutoFilters should be allowed to operate when the sheet is protected.

The possible values for this attribute are defined by the W3C XML Schema boolean datatype.</td></tr>
<tr><td>deleteColumns (Delete Columns Locked)</td><td>If 1 or true then deleting columns should not be allowed when the sheet is protected.

If 0 or false then deleting columns should be allowed when the sheet is protected.

The possible values for this attribute are defined by the W3C XML Schema boolean datatype.</td></tr>
<tr><td>deleteRows (Delete Rows Locked)</td><td>If 1 or true then deleting rows should not be allowed when the sheet is protected.

If 0 or false then deleting rows should be allowed when the sheet is protected.

The possible values for this attribute are defined by the W3C XML Schema boolean datatype.</td></tr>
<tr><td>formatCells (Format Cells Locked)</td><td>If 1 or true then formatting cells should not be allowed when the sheet is protected.

If 0 or false then formatting cells should be allowed when the sheet is protected.

The possible values for this attribute are defined by the W3C XML Schema boolean</td></tr>
</table>

Attributes	Description
	datatype.
formatColumns (Format Columns Locked)	If 1 or true then formatting columns should not be allowed when the sheet is protected. If 0 or false then formatting columns should be allowed when the sheet is protected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
formatRows (Format Rows Locked)	If 1 or true then formatting rows should not be allowed when the sheet is protected. If 0 or false then formatting rows should be allowed when the sheet is protected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
hashValue (Password Hash Value)	Specifies the hash value for the password required to edit this worksheet. 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 hash 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.
insertColumns (Insert Columns Locked)	If 1 or true then inserting columns should not be allowed when the sheet is protected. If 0 or false then inserting columns should be allowed when the sheet is protected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
insertHyperlinks (Insert Hyperlinks	If 1 or true then inserting hyperlinks should not be allowed when the sheet is protected.

Attributes	Description
Locked)	If 0 or false then inserting hyperlinks should be allowed when the sheet is protected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
insertRows (Insert Rows Locked)	If `1` or `true` then inserting rows should not be allowed when the sheet is protected. If 0 or false then inserting rows should be allowed when the sheet is protected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
objects (Objects Locked)	If `1` or `true` then editing of objects should not be allowed when the sheet is protected. If `0` or `false` then objects are allowed to be edited when the sheet is protected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
pivotTables (Pivot Tables Locked)	If `1` or `true` then PivotTables should not be allowed to operate when the sheet is protected. If 0 or false then PivotTables should be allowed to operate when the sheet is protected. The possible values for this attribute are defined by the W3C XML Schema boolean 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

Attributes	Description
	datatype.
scenarios (Scenarios Locked)	If `1` or `true` then Scenarios should not be edited when the sheet is protected. If `0` or `false` then Scenarios are allowed to be edited when the sheet is protected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
selectLockedCells (Select Locked Cells Locked)	If `1` or `true` then selection of locked cells should not be allowed when the sheet is protected. If 0 or false then selection of locked cells should be allowed when the sheet is protected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
selectUnlockedCells (Select Unlocked Cells Locked)	If `1` or `true` then selection of unlocked cells should not be allowed when the sheet is protected. If 0 or false then selection of unlocked cells should be allowed when the sheet is protected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sheet (Sheet Locked)	If `1` or `true` then the sheet is protected. If `0` or `false` then the sheet is not protected. The value of this attribute dictates whether the other attributes of sheetProtection should be applied. If 1 or true then the other attributes of sheetProtection should be applied. If 0 or false then the other attributes of sheetProtection should not be applied. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sort (Sort Locked)	If `1` or `true` then sorting should not be allowed when the sheet is protected. If `0` or `false` then sorting should be allowed when the sheet is protected. The possible values for this attribute are defined by the W3C XML Schema boolean 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

Attributes	Description
	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 hashValue attribute. *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SheetProtection) is located in §A.2. *end note*]

18.3.1.86 sheetView (Chart Sheet View)

This element specifies a chart sheet view. [*Note*: See sheetView (§18.3.1.87) for an example. *end note*]

Attributes	Description
tabSelected (Sheet Tab Selected)	Flag indicating whether the sheet tab is selected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
workbookViewId (Workbook View Id)	Zero-based index of this workbook view, pointing to a workbookView element in the bookViews collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
zoomScale (Window Zoom Scale)	Window zoom magnification, representing percent values. This attribute is restricted to values ranging from 10 to 400. Horizontal & Vertical scale together. [*Example*: 10 - 10% 20 - 20% … 100 - 100%

Attributes	Description
	… 400 - 400% *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
zoomToFit (Zoom To Fit)	Flag indicating whether chart sheet is zoom to fit window. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ChartsheetView) is located in §A.2. *end note*]

18.3.1.87 sheetView (Worksheet View)

A single sheet view definition. When more than one sheet view is defined in the file, it means that when opening the workbook, each sheet view corresponds to a separate window within the spreadsheet application, where each window is showing the particular sheet containing the same workbookViewId value, the last sheetView definition is loaded, and the others are discarded. When multiple windows are viewing the same sheet, multiple sheetView elements (with corresponding workbookView entries) are saved.

Attributes	Description
colorId (Color Id)	Index to the color value for row/column text headings and gridlines. This is an 'index color value' (ICV) rather than rgb value. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
defaultGridColor (Default Grid Color)	Flag indicating that the consuming application should use the default grid lines color (system dependent). Overrides any color specified in colorId. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rightToLeft (Right To Left)	Flag indicating whether the sheet is in 'right to left' display mode. When in this mode, Column A is on the far right, Column B ;is one column left of Column A, and so on. Also, information in cells is displayed in the Right to Left format. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showFormulas (Show Formulas)	Flag indicating whether this sheet should display formulas. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
	datatype.
showGridLines (Show Grid Lines)	Flag indicating whether this sheet should display gridlines. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showOutlineSymbols (Show Outline Symbols)	Flag indicating whether the sheet has outline symbols visible. This flag shall always override SheetPr element's outlinePr child element whose attribute is named showOutlineSymbols when there is a conflict. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showRowColHeaders (Show Headers)	Flag indicating whether the sheet should display row and column headings. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showRuler (Show Ruler)	Show the ruler in Page Layout View. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showWhiteSpace (Show White Space)	Flag indicating whether page layout view shall display margins. False means do not display left, right, top (header), and bottom (footer) margins (even when there is data in the header or footer). The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showZeros (Show Zero Values)	Flag indicating whether the window should show 0 (zero) in cells containing zero value. When false, cells with zero value appear blank instead of showing the number zero. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
tabSelected (Sheet Tab Selected)	Flag indicating whether this sheet is selected. When only 1 sheet is selected and active, this value should be in synch with the activeTab value. In case of a conflict, the Start Part setting wins and sets the active sheet tab. Multiple sheets can be selected, but only one sheet shall be active at one time. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
topLeftCell (Top Left Visible Cell)	Location of the top left visible cell Location of the top left visible cell in the bottom right pane (when in Left-to-Right mode). The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).

Attributes	Description
view (View Type)	Indicates the view type. The possible values for this attribute are defined by the ST_SheetViewType simple type (§18.18.69).
windowProtection (Window Protection)	Flag indicating whether the panes in the window are locked due to workbook protection. This is an option when the workbook structure is protected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
workbookViewId (Workbook View Index)	Zero-based index of this workbook view, pointing to a workbookView element in the bookViews collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
zoomScale (Zoom Scale)	Window zoom magnification for current view representing percent values. This attribute is restricted to values ranging from 10 to 400. Horizontal & Vertical scale together. [*Example*: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% *end example*] Current view can be Normal, Page Layout, or Page Break Preview. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
zoomScaleNormal (Zoom Scale Normal View)	Zoom magnification to use when in normal view, representing percent values. This attribute is restricted to values ranging from 10 to 400. Horizontal & Vertical scale together. [*Example*: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% *end example*]

Attributes	Description
	Applies for worksheets only; zero implies the automatic setting. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
zoomScalePageLayoutView (Zoom Scale Page Layout View)	Zoom magnification to use when in page layout view, representing percent values. This attribute is restricted to values ranging from 10 to 400. Horizontal & Vertical scale together. [*Example*: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% *end example*] Applies for worksheets only; zero implies the automatic setting. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
zoomScaleSheetLayoutView (Zoom Scale Page Break Preview)	Zoom magnification to use when in page break preview, representing percent values. This attribute is restricted to values ranging from 10 to 400. Horizontal & Vertical scale together. [*Example*: 10 - 10% 20 - 20% … 100 - 100% … 400 - 400% *end example*] Applies for worksheet only; zero implies the automatic setting. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SheetView) is located in §A.2. *end note*]

18.3.1.88 sheetViews (Sheet Views)

Worksheet views collection.

[*Example*:

This example shows one sheet view definition. The definition indicates that the current sheet is the active/selected sheet, and that there is a split pane applied to the view. This definition also indicates for each of the four window panes of the split which cell is the active cell for that pane.

```
<sheetViews>
  <sheetView tabSelected="1" workbookViewId="0">
    <pane xSplit="2310" ySplit="2070" topLeftCell="C1"
      activePane="bottomRight"/>
    <selection/>
    <selection pane="bottomLeft" activeCell="A6" sqref="A6"/>
    <selection pane="topRight" activeCell="C1" sqref="C1"/>
    <selection pane="bottomRight" activeCell="E13" sqref="E13"/>
  </sheetView>
</sheetViews>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SheetViews) is located in §A.2. *end note*]

18.3.1.89 sheetViews (Chart Sheet Views)

This element specifies chart sheet views.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ChartsheetViews) is located in §A.2. *end note*]

18.3.1.90 smartTags (Smart Tags)

This collection expresses all smart tags associated with cells on this sheet. There can be multiple smart tags associated with a particular cell, and many cells with smart tags for a given worksheet.

[*Example*:

This example shows three smart tags, each one associated with a unique cell on the worksheet.


```
<smartTags>
  <cellSmartTags r="A1">
    <cellSmartTag type="0"/>
  </cellSmartTags>
  <cellSmartTags r="B1">
    <cellSmartTag type="0"/>
  </cellSmartTags>
  <cellSmartTags r="B2">
    <cellSmartTag type="0"/>
  </cellSmartTags>
</smartTags>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SmartTags) is located in §A.2. *end note*]

18.3.1.91 sortCondition (Sort Condition)

Sort condition. When more than one sortCondition is specified, the first condition is applied first, then the second condition is applied, and so on.

Attributes	Description
customList (Custom List)	Sort by a custom list. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
descending (Descending)	Sort descending. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dxfId (Format Id)	Format Id when sortBy=cellColor or fontColor The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).
iconId (Icon Id)	Zero-based index of an icon in an icon set. The absence of this attribute means "no icon" The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
iconSet (Icon Set)	Icon set index when sortBy=icon. The possible values for this attribute are defined by the ST_IconSetType simple type (§18.18.42).
ref (Reference)	Column/Row that this sort condition applies to. This shall be contained within the ref in CT_SortState.

Attributes	Description
	The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
sortBy (Sort By)	Type of sort. The possible values for this attribute are defined by the ST_SortBy simple type (§18.18.72).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SortCondition) is located in §A.2. *end note*]

18.3.1.92 sortState (Sort State)

This collection preserves the AutoFilter sort state.

[*Example*: This example shows a sort which is case-sensitive, descending sort. While the range of data to sort is B4:E8, the range to sort by is B4:B8.

```
<sortState caseSensitive="1" ref="B4:E8">
  <sortCondition descending="1" ref="B4:B8"/>
</sortState>
```

end example]

Attributes	Description
caseSensitive (Case Sensitive)	Flag indicating whether or not the sort is case-sensitive. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
columnSort (Sort by Columns)	Flag indicating whether or not to sort by columns. Only applies to ranges that don't have AutoFilter applied. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ref (Sort Range)	The whole range of data to sort (not just the sort-by column). The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
sortMethod (Sort Method)	Strokes or PinYin sort method. Applies only to these application UI languages: • Chinese Simplified • Chinese Traditional • Japanese • For these languages, alternate sort methods can be selected, affecting how the data is sorted.

Attributes	Description
	The possible values for this attribute are defined by the ST_SortMethod simple type (§18.18.73).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SortState) is located in §A.2. *end note*]

18.3.1.93 tabColor (Sheet Tab Color)

Background color of the sheet tab.

Attributes	Description
auto (Automatic)	A boolean value indicating the color is automatic and system color dependent. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
indexed (Index)	Indexed color value. Only used for backwards compatibility. References a color in indexedColors. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rgb (Alpha Red Green Blue Color Value)	Standard Alpha Red Green Blue color value (ARGB). The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
theme (Theme Color)	A zero-based index into the `<clrScheme>` collection (§20.1.6.2), referencing a particular `<sysClr>` or `<srgbClr>` value expressed in the Theme part. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
tint (Tint)	Specifies the tint value applied to the color. If tint is supplied, then it is applied to the RGB value of the color to determine the final color applied. The tint value is stored as a double from -1.0 .. 1.0, where -1.0 means 100% darken and 1.0 means 100% lighten. Also, 0.0 means no change. In loading the RGB value, it is converted to HLS where HLS values are (0..HLSMAX), where HLSMAX is currently 255. [*Example*:

Attributes	Description
	Here are some examples of how to apply tint to color: **If (tint < 0)** Lum' = Lum * (1.0 + tint) For example: Lum = 200; tint = -0.5; Darken 50% Lum' = 200 * (0.5) => 100 For example: Lum = 200; tint = -1.0; Darken 100% (make black) Lum' = 200 * (1.0-1.0) => 0 **If (tint > 0)** Lum' = Lum * (1.0-tint) + (HLSMAX – HLSMAX * (1.0-tint)) For example: Lum = 100; tint = 0.75; Lighten 75% Lum' = 100 * (1-.75) + (HLSMAX – HLSMAX*(1-.75)) = 100 * .25 + (255 – 255 * .25) = 25 + (255 – 63) = 25 + 192 = 217 For example: Lum = 100; tint = 1.0; Lighten 100% (make white) Lum' = 100 * (1-1) + (HLSMAX – HLSMAX*(1-1)) = 100 * 0 + (255 – 255 * 0) = 0 + (255 – 0) = 255 *end example*] The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Color) is located in §A.2. *end note*]

18.3.1.94 tablePart (Table Part)

A single Table Part reference.

Attributes	Description
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	This relationship Id is used to locate a particular table definition part. 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_TablePart) is located in §A.2. *end note*]

18.3.1.95 tableParts (Table Parts)

This collection expresses a relationship Id pointing to every table on this sheet.

[*Example*: This example indicates that the current sheet has two tables, and their definitions can be found by locating the appropriate relationships from the sheet:

```
<tableParts count="2">
  <tablePart r:id="rId1"/>
  <tablePart r:id="rId2"/>
</tableParts>
```

end example]

Attributes	Description
count (Count)	A count of table elements in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TableParts) is located in §A.2. *end note*]

18.3.1.96 v (Cell Value)

This element expresses the value contained in a cell. If the cell contains a string, then this value is an index into the shared string table, pointing to the actual string value. Otherwise, the value of the cell is expressed directly in this element. Cells containing formulas express the last calculated result of the formula in this element.

For applications not wanting to implement the shared string table, an 'inline string' can be expressed in an <is> element under <c> (instead of a <v> element under <c>),in the same way a string would be expressed in the shared string table. [*Note*: See <is> for an example. *end note*]

[*Example*: In this example, cell B4 contains the number "360", cell C4 contains the local date and time 22 November 1976, 08:30, and cell C5 contains the 1900 date system serial date-time for the date-time in cell C4.

```
<c r="B4">
  <v>360</v>
</c>
<c r="C4" t="d">
  <v>1976-11-22T08:30</v>
</c>
```



```
<c r="C5">
  <f>C4</f>
   <v>28086.3541666667</v>
</c>
```

end example]

The possible values for this element are defined by the ST_Xstring simple type (§22.9.2.19).

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. 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 (ST_Xstring) is located in §A.6.9. *end note*]

18.3.1.97 webPublishItem (Web Publishing Item)

This element represents information for a single item or object which can be published to HTML.

Attributes	Description
autoRepublish (Automatically Publish)	Automatically publish this item every time the workbook is saved. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
destinationFile (Destination File Name)	Destination file name. Indicates where to save the HTML publish file. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
divId (Destination Bookmark)	Destination bookmark. Identifies a specific <div> section in the published HTML file when a subset of the workbook is published to HTML. Each item that has been published from a workbook is written to a unique <div>element in HTML. On re-publishing a particular item from the workbook, only that item's corresponding <div> content is updated. Therefore each publish item corresponds to a unique <div> element. It is possible to add new publish items to an existing published page, and it is possible to re-publish individual items without republishing the entire workbook. The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§22.9.2.19).
id (Id)	This is a unique number "nnnnn" of the webPublishItem. This value is used to generate the divId and styleId values. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sourceObject (Source Object Name)	Source object name (required for sourceType = pivotTable, query, or label). The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
sourceRef (Source Id)	Source range (required for sourceType = 'range'). The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
sourceType (Web Source Type)	Type of web source (or objects to publish). The possible values for this attribute are defined by the ST_WebSourceType simple type (§18.18.92).
title (Title)	HTML title of published item. [*Example*: This value can appear in the web browser window's title bar. *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_WebPublishItem) is located in §A.2. *end note*]

18.3.1.98 webPublishItems (Web Publishing Items)

This represents a listing of individual objects in this workbook that have been published (to HTML).

When one of these objects is selected to be published, just the object is published to HTML, not the entire workbook contents.

[*Example*: This example shows two items which have been previously selected for publishing. One is a range (A6:C6), the other is a chart, named "Chart 1".

```
<webPublishItems count="2">
  <webPublishItem id="11289" divId="Views_11289" sourceType="range"
    sourceRef="A6:C6" destinationFile="D:\Publish.htm" published="0"/>
  <webPublishItem id="6433" divId="Views_6433" sourceType="chart"
    sourceObject="Chart 1" destinationFile="D:\Publish.mht" published="0"/>
</webPublishItems>
```


end example]

Attributes	Description
count (Web Publishing Items Count)	Number of items. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_WebPublishItems) is located in §A.2. *end note*]

18.3.1.99 worksheet (Worksheet)

This is the root element of Worksheet parts within a SpreadsheetML document.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Worksheet) is located in §A.2. *end note*]

18.3.2 AutoFilter Settings

The following subclause defines the settings which can be specified as part of an AutoFilter definition. An *AutoFilter* temporarily hides rows based on a filter criteria, which is applied column by column to a table of data in the worksheet.

18.3.2.1 colorFilter (Color Filter Criteria)

This element specifies the color to filter by and whether to use the cell's fill or font color in the filter criteria. If the cell's font or fill color does not match the color specified in the criteria, the rows corresponding to those cells are hidden from view.

[*Example*:

```
<filterColumn colId="1">
  <colorFilter dxfId="0" cellColor="0"/>
</filterColumn>
```

end example]

Attributes	Description
cellColor (Filter By Cell Color)	Flag indicating whether or not to filter by the cell's fill color. 1 indicates to filter by cell fill. 0 indicates to filter by the cell's font color. For rich text in cells, if the color specified appears in the cell at all, it shall be included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
dxfId (Differential Format Record Id)	Id of differential format record (dxf) in the Styles Part which expresses the color value to filter by. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ColorFilter) is located in §A.2. *end note*]

18.3.2.2 customFilter (Custom Filter Criteria)

A custom AutoFilter specifies an operator and a value. There can be at most two customFilters specified, and in that case the parent element specifies whether the two conditions are joined by 'and' or 'or'. For any cells whose values do not meet the specified criteria, the corresponding rows shall be hidden from view when the filter is applied.

[*Example*:

```
<customFilters and="1">
  <customFilter operator="greaterThanOrEqual" val="0.2"/>
  <customFilter operator="lessThanOrEqual" val="0.5"/>
</customFilters>
```

end example]

Attributes	Description
operator (Filter Comparison Operator)	Operator used by the filter comparison. The possible values for this attribute are defined by the ST_FilterOperator simple type (§18.18.31).
val (Top or Bottom Value)	Top or bottom value used in the filter criteria. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomFilter) is located in §A.2. *end note*]

18.3.2.3 customFilters (Custom Filters)

When there is more than one custom filter criteria to apply (an 'and' or 'or' joining two criteria), then this element groups the customFilter elements together.

Attributes	Description
and (And)	Flag indicating whether the two criteria have an "and" relationship. 1 indicates "and", 0 indicates "or". The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomFilters) is located in §A.2. *end note*]

18.3.2.4 dateGroupItem (Date Grouping)

This collection is used to express a group of dates or times which are used in an AutoFilter criteria. [*Note*: See parent element for an example. *end note*] Values are always written in the calendar type of the first date encountered in the filter range, so that all subsequent dates, even when formatted or represented by other calendar types, can be correctly compared for the purposes of filtering.

Attributes	Description
dateTimeGrouping (Date Time Grouping)	Grouping level. The possible values for this attribute are defined by the ST_DateTimeGrouping simple type (§18.18.22).
day (Day)	Day (1-31) The possible values for this attribute are defined by the W3C XML Schema unsignedShort datatype.
hour (Hour)	Hour (0-23) The possible values for this attribute are defined by the W3C XML Schema unsignedShort datatype.
minute (Minute)	Minute (0-59) The possible values for this attribute are defined by the W3C XML Schema unsignedShort datatype.
month (Month)	Month (1-12) The possible values for this attribute are defined by the W3C XML Schema unsignedShort datatype.
second (Second)	Second (0-59) The possible values for this attribute are defined by the W3C XML Schema unsignedShort datatype.
year (Year)	Year (4 digits)

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema unsignedShort datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DateGroupItem) is located in §A.2. *end note*]

18.3.2.5 dynamicFilter (Dynamic Filter)

This collection specifies dynamic filter criteria. These criteria are considered dynamic because they can change, either with the data itself (e.g., "above average") or with the current system date (e.g., show values for "today"). For any cells whose values do not meet the specified criteria, the corresponding rows shall be hidden from view when the filter is applied.

[*Example*:

```
<filterColumn colId="0">
  <dynamicFilter type="today"/>
</filterColumn>
```

end example]

Attributes	Description
maxValIso (Max ISO Value)	A maximum value for dynamic filter. maxValIso shall be required for today, yesterday, tomorrow, nextWeek, thisWeek, lastWeek, nextMonth, thisMonth, lastMonth, nextQuarter, thisQuarter, lastQuarter, nextYear, thisYear, lastYear, and yearToDate. The above criteria are based on a value range; that is, if today's date is September 22nd, then the range for thisWeek is the values greater than or equal to September 17 and less than September 24. In the thisWeek range, the lower value is expressed valIso. The higher value is expressed using maxValIso. These dynamic filters shall not require valIso or maxValIso: Q1, Q2, Q3, Q4, M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11 and M12. The above criteria shall not specify the range using valIso and maxValIso because Q1 always starts from M1 to M3, and M1 is always January. These types of dynamic filters shall use valIso and shall not use maxValIso: • aboveAverage and belowAverage The possible values for this attribute are defined by the W3C XML Schema dateTime datatype.

Attributes	Description
type (Dynamic filter type)	Dynamic filter type, e.g., "today" or "nextWeek". The possible values for this attribute are defined by the ST_DynamicFilterType simple type (§18.18.26).
val (Value)	A minimum numeric value for dynamic filter. (See description of valIso to understand when val is required.) The possible values for this attribute are defined by the W3C XML Schema double datatype.
valIso (ISO Value)	A minimum value for dynamic filter. (See description of maxValIso to understand when val/valIso is required.) Only these types of dynamic filters use numeric data, and therefore shall use val and shall not use valIso: • aboveAverage and belowAverage The possible values for this attribute are defined by the W3C XML Schema dateTime datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DynamicFilter) is located in §A.2. *end note*]

18.3.2.6 filter (Filter)

This element expresses a filter criteria value.

[*Example*:

```
<filters>
  <filter val="0.316588716"/>
  <filter val="0.667439395"/>
  <filter val="0.823086999"/>
</filters>
```

end example]

Attributes	Description
val (Filter Value)	Filter value used in the criteria. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Filter) is located in §A.2. *end note*]

18.3.2.7 filterColumn (AutoFilter Column)

The filterColumn collection identifies a particular column in the AutoFilter range and specifies filter information that has been applied to this column. If a column in the AutoFilter range has no criteria specified, then there is no corresponding filterColumn collection expressed for that column.

Attributes	Description
colId (Filter Column Data)	Zero-based index indicating the AutoFilter column to which this filter information applies. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
hiddenButton (Hidden AutoFilter Button)	Flag indicating whether the AutoFilter button for this column is hidden. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showButton (Show Filter Button)	Flag indicating whether the filter button is visible. [*Example*: When the cell containing the filter button is merged with another cell, the filter button can be hidden, and not drawn. *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_FilterColumn) is located in §A.2. *end note*]

18.3.2.8 filters (Filter Criteria)

When multiple values are chosen to filter by, or when a group of date values are chosen to filter by, this element groups those criteria together.

[*Example*:

```
<filters>
  <dateGroupItem year="2006" month="1" day="2" dateTimeGrouping="day"/>
  <dateGroupItem year="2005" month="1" day="2" dateTimeGrouping="day"/>
</filters>
```

end example]

Attributes	Description
blank (Filter by Blank)	Flag indicating whether to filter by blank. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
calendarType	Calendar type for date grouped items. Used to interpret the values in dateGroupItem.

Attributes	Description
(Calendar Type)	This is the calendar type used to evaluate all dates in the filter column, even when those dates are not using the same calendar system / date formatting. 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_Filters) is located in §A.2. *end note*]

18.3.2.9 iconFilter (Icon Filter)

This element specifies the icon set and particular icon within that set to filter by. For any cells whose icon does not match the specified criteria, the corresponding rows shall be hidden from view when the filter is applied.

[*Example*:

```
<filterColumn colId="3">
  <iconFilter iconSet="3Arrows" iconId="0"/>
</filterColumn>
```

end example]

Attributes	Description
iconId (Icon Id)	Zero-based index of an icon in an icon set. The absence of this attribute means "no icon" The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
iconSet (Icon Set)	Specifies which icon set is used in the filter criteria. The possible values for this attribute are defined by the ST_IconSetType simple type (§18.18.42).

[*Note*: The W3C XML Schema definition of this element's content model (CT_IconFilter) is located in §A.2. *end note*]

18.3.2.10 top10 (Top 10)

This element specifies the top N (percent or number of items) to filter by.

[*Example*: This example filters the first column by the top 10 percent of the values in that column. For all cells in the column whose value falls outside the top 10 percent of the value in that column, the rows corresponding to those cells are hidden from the view. In this example, there are 6 cells in the range, containing 1, 2, 3, 4, 5, 6 respectively.


```
<filterColumn colId="0">
  <top10 percent="1" val="5" filterVal="6"/>
</filterColumn
```

end example]

Attributes	Description
filterVal (Filter Value)	The actual cell value in the range which is used to perform the comparison for this filter. The possible values for this attribute are defined by the W3C XML Schema double datatype.
percent (Filter by Percent)	Flag indicating whether or not to filter by percent value of the column. A false value filters by number of items. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
top (Top)	Flag indicating whether or not to filter by top order. A false value filters by bottom order. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
val (Top or Bottom Value)	Top or bottom value to use as the filter criteria. [*Example*: "Filter by Top 10 Percent" or "Filter by Top 5 Items". *end example*] The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Top10) is located in §A.2. *end note*]

18.4 Shared String Table

A workbook can contain thousands of cells containing string (non-numeric) data. Furthermore this data is very likely to be repeated across many rows or columns. The goal of implementing a single string table that is shared across the workbook is to improve performance in opening and saving the file by only reading and writing the repetitive information once.

[*Example*: Consider for example a workbook summarizing information for cities within various countries. There can be a column for the name of the country, a column for the name of each city in that country, and a column containing the data for each city. In this case the country name is repetitive, being duplicated in many cells. *end example*] In many cases the repetition is extensive, and significant savings are realized by making use of a shared string table when saving the workbook. When displaying text in the spreadsheet, the cell table will just contain an index into the string table as the value of a cell, instead of the full string.

The shared string table is permitted to contain all the necessary information for displaying the string: the text, formatting properties, and phonetic properties (for East Asian languages).

Most strings in a workbook have formatting applied at the cell level, that is, the entire string in the cell has the same formatting applied. In these cases, the formatting for the cell is stored in the styles part, and the string for the cell can be stored in the shared strings table. In this case, the strings stored in the shared strings table are very simple text elements.

[*Example*:

```
<sst xmlns="http://purl.oclc.org/ooxml/spreadsheetml/main"
  count="8" uniqueCount="4">
  <si>
    <t>United States</t>
  </si>
  <si>
    <t>Seattle</t>
  </si>
  <si>
    <t>Denver</t>
  </si>
  <si>
    <t>New York</t>
  </si>
</sst>
```

In the above example we can see that the string table is just a collection of string items that consist of simple text elements. Note that any numeric data in the workbook is not shown in the shared string table. *end example*]

Some strings in the workbook can have formatting applied at a level that is more granular than the cell level. For instance, specific characters within the string can be bolded, have coloring, italicizing, etc. In these cases, the formatting is stored along with the text in the string table, and is treated as a unique entry in the table.

[*Example*:

```
<sst xmlns="http://purl.oclc.org/ooxml/spreadsheetml/main"
  count="8" uniqueCount="4">
  <si>
    <r>
      <t xml:space="preserve">United </t>
    </r>
```



```
      <r>
        <rPr>
          <sz val="11"/>
          <color rgb="FFFF0000"/>
          <rFont val="Calibri"/>
          <family val="2"/>
          <scheme val="minor"/>
        </rPr>
        <t>States</t>
      </r>
    </si>
    <si>
      <t>Seattle</t>
    </si>
    <si>
      <t>Denver</t>
    </si>
    <si>
      <t>New York</t>
    </si>
  </sst>
```

In the above example you can see that this time, the text "United States" has specific, colored, formatting applied to the text, "States." *end example*]

18.4.1 charset (Character Set)

This element defines the font character set of this font.

This field is used in font creation and selection if a font of the given facename is not available on the system. Although it is not required to have around when resolving font facename, the information can be stored for when needed to help resolve which font face to use of all available fonts on a system.

Charset represents the basic set of characters associated with a font (that it can display), and roughly corresponds to the ANSI codepage (8-bit or DBCS) of that character set used by a given language. Given more common use of Unicode where many fonts support more than one of the traditional charset categories, and the use of font linking, using charset to resolve font name is less and less common, but still can be useful.

These are operating-system-dependent values.

[*Note*: The following are some of the possible the character sets:

INT Value	Character Set
0	ANSI_CHARSET

INT Value	Character Set
1	DEFAULT_CHARSET
2	SYMBOL_CHARSET
77	MAC_CHARSET
128	SHIFTJIS_CHARSET
129	HANGUL_CHARSET
130	JOHAB_CHARSET
134	GB2312_CHARSET
136	CHINESEBIG5_CHARSET
161	GREEK_CHARSET
162	TURKISH_CHARSET
163	VIETNAMESE_CHARSET
177	HEBREW_CHARSET
178	ARABIC_CHARSET
186	BALTIC_CHARSET
204	RUSSIAN_CHARSET
222	THAI_CHARSET
238	EASTEUROPE_CHARSET
255	OEM_CHARSET

The OEM_CHARSET value specifies a character set that is operating-system dependent. *end note*]

Fonts with other character sets can exist in the operating system. If an application uses a font with an unknown character set, it should not attempt to translate or interpret strings that are rendered with that font.

Attributes	Description
val (Value)	The value of an integer, where each value corresponds to a different character set. This attribute is restricted to values ranging from 0 to 255. The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_IntProperty) is located in §A.2. *end note*]

18.4.2 outline (Outline)

This element displays only the inner and outer borders of each character. This is very similar to Bold in behavior.

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_BooleanProperty) is located in §A.2. *end note*]

18.4.3 phoneticPr (Phonetic Properties)

This element represents a collection of phonetic properties that affect the display of phonetic text for this String Item (si).

Phonetic text is used to give hints as to the pronunciation of an East Asian language, and the hints are displayed as text within the spreadsheet cells across the top portion of the cell. Since the phonetic hints are text, every phonetic hint is expressed as a phonetic run (rPh), and these properties specify how to display that phonetic run.

[*Example*:

```
<si>
  <t>課きく 毛こ</t>
  <rPh sb="0" eb="1">
    <t>カ</t>
  </rPh>
  <rPh sb="4" eb="5">
    <t>ケ</t>
  </rPh>
  <phoneticPr fontId="1"/>
</si>
```

The above example shows a String Item that displays some Japanese text "課きく　毛こ." It also displays some phonetic text across the top of the cell. The phonetic text character, "カ" is displayed over the "課" character and the phonetic text "ケ" is displayed above the "毛" character, using the font record in the style sheet at index 1. *end example*]

Attributes	Description
alignment (Alignment)	Specifies how the text for the phonetic run is aligned across the top of the cells, with respect to the main text in the body of the cell. The possible values for this attribute are defined by the ST_PhoneticAlignment simple

Attributes	Description
	type (§18.18.56).
fontId (Font Id)	An integer that is a zero-based index into the font record in the style sheet. Represents the font to be used to display this phonetic run. If this index is out of bounds, then the default font of the Normal style should be used in its place. This default font should be at index 0. The possible values for this attribute are defined by the ST_FontId simple type (§18.18.32).
type (Character Type)	An enumeration which specifies which East Asian character set should be used to display the phonetic run The possible values for this attribute are defined by the ST_PhoneticType simple type (§18.18.57).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PhoneticPr) is located in §A.2. *end note*]

18.4.4 r (Rich Text Run)

This element represents a run of rich text. A rich text run is a region of text that share a common set of properties, such as formatting properties. The properties are defined in the rPr element, and the text displayed to the user is defined in the Text (t) element.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RElt) is located in §A.2. *end note*]

18.4.5 rFont (Font)

This element is a string representing the name of the font assigned to display this run.

Attributes	Description
val (String Value)	A string representing the name of the font. If the font doesn't exist (because it isn't installed on the system), or the charset not supported by that font, then another font should be substituted. The string length for this attribute shall be 0 to 31 characters. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FontName) is located in §A.2. *end note*]

18.4.6 rPh (Phonetic Run)

This element represents a run of text which displays a phonetic hint for this String Item (si).

Phonetic hints are used to give information about the pronunciation of an East Asian language. The hints are displayed as text within the spreadsheet cells across the top portion of the cell.

Attributes	Description
eb (Base Text End Index)	An integer used as a zero-based index representing the ending offset into the base text for this phonetic run. This represents the ending point in the base text the phonetic hint applies to. This value shall be between 0 and the total length of the base text. The following condition shall be true: sb < eb. It is recommended that the following condition also be satisfied: That for any two consecutive phonetic runs, sb1 < eb1 <= sb2 < eb2 to avoid overlapping phonetic runs The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sb (Base Text Start Index)	An integer used as a zero-based index representing the starting offset into the base text for this phonetic run. This represents the starting point in the base text the phonetic hint applies to. This value shall be between 0 and the total length of the base text. The following condition shall be true: sb < eb. It is recommended that the following condition also be satisfied: That for any two consecutive phonetic runs, sb1 < eb1 <= sb2 < eb2 to avoid overlapping phonetic runs. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PhoneticRun) is located in §A.2. *end note*]

18.4.7 rPr (Run Properties)

This element represents a set of properties to apply to the contents of this rich text run.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RPrElt) is located in §A.2. *end note*]

18.4.8 si (String Item)

This element is the representation of an individual string in the Shared String table.

If the string is just a simple string with formatting applied at the cell level, then the String Item (si) should contain a single text element used to express the string. However, if the string in the cell is more complex - i.e., has formatting applied at the character level - then the string item shall consist of multiple rich text runs which collectively are used to express the string.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rst) is located in §A.2. *end note*]

18.4.9 sst (Shared String Table)

This element is the root of the Shared String Table, which serves as a collection of individual String Items (si).

Attributes	Description
count (String Count)	An integer representing the total count of strings in the workbook. This count does not include any numbers, it counts only the total of text strings in the workbook. This attribute is optional unless `uniqueCount` is used, in which case it is required. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
uniqueCount (Unique String Count)	An integer representing the total count of unique strings in the Shared String Table. A string is unique even if it is a copy of another string, but has different formatting applied at the character level. [*Example*: World, *World*, and *W*orld. The `count` would be 3, and the `uniqueCount` would be 2. Only one entry for "World" would show in the table because it is the same string, just with different formatting applied at the cell level (i.e., applied to the entire string in the cell). The "*W*orld" string would get a separate unique entry in the shared string table because it has different formatting applied to specific characters. *end example*] This attribute is optional unless `count` is used, in which case it is required. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Sst) is located in §A.2. *end note*]

18.4.10 strike (Strike Through)

This element draws a strikethrough line through the horizontal middle of the text.

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element.

Attributes	Description
	If omitted, the default value is `true`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_BooleanProperty) is located in §A.2. *end note*]

18.4.11 sz (Font Size)

This element represents the point size (1/72 of an inch) of the Latin and East Asian text.

Attributes	Description
val (Value)	A double representing the value of a positive measurement in points (1/72 of an inch). The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_FontSize) is located in §A.2. *end note*]

18.4.12 t (Text)

This element represents the text content shown as part of a string.

The possible values for this element are defined by the ST_Xstring simple type (§22.9.2.19).

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. 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 (ST_Xstring) is located in §A.6.9. *end note*]

18.4.13 u (Underline)

This element represents the underline formatting style.

Attributes	Description
val (Underline Value)	An enumeration representing the style of underlining that is used. The none style is equivalent to not using underlining at all. The possible values for this attribute are defined by the ST_UnderlineValues simple type (§18.18.85).

[*Note*: The W3C XML Schema definition of this element's content model (CT_UnderlineProperty) is located in §A.2. *end note*]

18.4.14 vertAlign (Vertical Alignment)

This element adjusts the vertical position of the text relative to the text's default appearance for this run. It is used to get 'superscript' or 'subscript' texts, and shall reduce the font size (if a smaller size is available) accordingly.

Attributes	Description
val (Value)	An enumeration representing the vertical-alignment setting. Setting this to either `subscript` or `superscript` shall make the font size smaller if a smaller font size is available. 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_VerticalAlignFontProperty) is located in §A.2. *end note*]

18.5 Tables

A table helps organize and provide structure to lists of information in a worksheet. Tables have clearly labeled columns, rows, and data regions. Tables make it easier for users to sort, analyze, format, manage, add, and delete information.

If a region of data is designated as a Table, then special behaviors can be applied which help the user perform useful actions. [*Example*: if the user types additional data in the row adjacent to the bottom of the table, the table can expand and automatically add that data to the data region of the table. Similarly, adding a column is as easy as typing a new column heading to the right or left of the current column headings. Filter and sort abilities can automatically be surfaced to the user via the drop down arrows. Special calculated columns can be created which summarize or calculate data in the table. These columns have the ability to expand and shrink according to size of the table, and maintain proper formula referencing. *end example*]

Tables can be created from data already present in the worksheet, from an external data query, or from mapping a collection of repeating XML elements to a worksheet range.

The sheet XML stores the numeric and textual data. The table XML records the various attributes for the particular table object.

[*Example*:

```
<table xmlns="http://purl.oclc.org/ooxml/spreadsheetml/main"
  id="1" name="MarginTable" displayName="MarginTable" ref="D3:G6"
  totalsRowShown="0">
  <autoFilter ref="D3:G6"/>
  <tableColumns count="4">
    <tableColumn id="1" name="Product"/>
    <tableColumn id="2" name="Wholesale"/>
    <tableColumn id="3" name="Retail"/>
    <tableColumn id="4" name="Margin" dataDxfId="0">
      <calculatedColumnFormula d="1">[Retail]-
        [Wholesale]</calculatedColumnFormula>
    </tableColumn>
  </tableColumns>
  <tableStyleInfo name="TableStyleMedium9" showFirstColumn="0"
    showLastColumn="0" showRowStripes="1" showColumnStripes="0"/>
</table>
```

The above xml example shows a table that spans cells D3 through G6, and has four columns: Product, Wholesale, Retail, and Margin. Margin is a column where each cell has its values calculated based on the formula (Retail - Wholesale), where those values are taken from the cells in the table columns on the corresponding row. The table has a style applied, "TableStyleMedium9", but the styles formatting isn't applied to the first column and the column striping isn't shown. Note that all the data and text values are stored in the sheet xml; the table xml just stores the properties that are specific to this table, and it is referenced by the sheet. *end example*]

18.5.1 Tables

Tables are ranges of data in the worksheet that have special behavior applied which allow users to better sort, analyze, format, manage, add, and delete data. Tables and table columns can also be referenced through formulas by the spreadsheet application using friendly names, making formula calculations that use tables much easier to understand and maintain. Tables provide a natural way for working with large sets of tabular data.

The tables described in this section are of the multi cell variety, as opposed to single cell tables created from XML mappings.

Each table gets its own xml part, and the relationship between a table part and the sheet is defined in the sheet's _rels directory. The sheet xml also references this id since there can be more than one table on a sheet. The sheet xml contains all the numeric and textual data, and the table xml records properties of the table as well as some formatting rules for data and text displayed in the table cells.

18.5.1.1 calculatedColumnFormula (Calculated Column Formula)

Columns in a table can have cells that are calculated, usually based on values in other cells in the table. This element stores the formula that is used to perform the calculation for each cell in this column.

It shall be understood that formulas which reference columns of this table, shall be calculated using the cells in those columns on the same row of the table as the cell that the formula resides in.

See §18.17 for details on the format required for formulas.

Attributes	Description
array (Array)	A Boolean value that indicates whether this formula is an array style formula. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
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. 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_TableFormula) is located in §A.2. *end note*]

18.5.1.2 table (Table)

This element is the root element for a table that is not a single cell XML table.

Attributes	Description
comment (Table Comment)	A string representing a textual comment about the table. [*Note*: This can be used by the spreadsheet application in other UI. *end note*] [*Example*: There can be name UI that is used to organize defined names and function references, if tables are listed in that UI the comment can give more information about the table. *end example*] The maximum length of this string should be 32767 characters. The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§22.9.2.19).
connectionId (Connection ID)	An integer representing an ID to indicate which connection from the connections collection is used by this table. This shall only be used for tables that are based off of xml maps. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
dataCellStyle (Data Style Name)	A string representing the name of the cell style that is applied to the data area cells of the table. If this string is missing or does not correspond to the name of a cell style, then the data cell style specified by the current table style should be applied. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
dataDxfId (Data Area Format Id)	A zero based integer index into the differential formatting records `<dxfs>` in the styleSheet indicating which format to apply to the data area of this table. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).
displayName (Table Name)	A string representing the name of the table. This is the name that shall be used in formula references, and displayed in the UI to the spreadsheet user. This name shall not have any spaces in it, and it shall be unique amongst all other `displayNames` and `definedNames` in the workbook. The character lengths and restrictions are the same as for `definedNames`. See *SpreadsheetML Reference - Workbook definedNames* section for details The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
headerRowBorderDxfId (Header Row Border Format Id)	A zero based integer index into the differential formatting records `<dxfs>` in the styleSheet indicating what border formatting to apply to the header row of this table. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).
headerRowCellStyle (Header Row Style)	A string representing the name of the cell style that is applied to the header row cells of the table. If this string is missing or does not correspond to the name of a cell style, then the header

Attributes	Description
	row style specified by the current table style should be applied. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
headerRowCount (Header Row Count)	An integer representing the number of header rows showing at the top of the table. 0 means that the header row is not shown. It is up to the spreadsheet application to determine if numbers greater than 1 are allowed. Unless the spreadsheet application has a feature where there might ever be more than one header row, this number should not be higher than 1. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
headerRowDxfId (Header Row Format Id)	A zero based integer index into the differential formatting records <dxfs>in the styleSheet indicating which format to apply to the header row of this table. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).
id (Table Id)	A non zero integer representing the unique identifier for this table. Each table in the workbook shall have a unique id. Ids can be used to refer to the specific table in the workbook. [*Note*: For instance a future records bucket could refer to the table using this id. *end note*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
insertRow (Insert Row Showing)	A Boolean value indicating whether the insert row is showing. True when the insert row is showing, false otherwise. The insert row should only be shown if the table has no data. When a user clicks the insert row in the UI, it provides them an easy way to enter data into a table. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
insertRowShift (Insert Row Shift)	A Boolean that indicates whether cells in the sheet had to be inserted when the insert row was shown for this table. True if the cells were shifted, false otherwise. [*Note*: This happens when there are values in cells immediately below the table when the table is created and the insert row is shown. In this case blank cells for the insert row are inserted, and the existing values in the sheet are shifted down by one row to make

Attributes	Description
	room. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
name (Name)	A string representing the name of the table that is used to reference the table programmatically through the spreadsheet applications object model. This string shall be unique per table per sheet. It has the same length and character restrictions as for displayName. By default this should be the same as the table's `displayName`. This name should also be kept in synch with the `displayName` when the `displayName` is updated in the UI by the spreadsheet user. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
published (Published)	A Boolean representing whether this table is marked as published for viewing by a server based spreadsheet application. True if it should be viewed by the server spreadsheet application, false otherwise. [*Note*: Such an application might only display objects from the workbook that are marked as published, thus being able to load and calculate the entire workbook but only show the specific items that are marked as published. This can allow the server spreadsheet rendering to provide a more restricted view of the workbook. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ref (Reference)	The range on the relevant sheet that the table occupies expressed using A1 style referencing. The reference shall include the totals row if it is shown. The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
tableBorderDxfId (Table Border Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating what border formatting to apply to the borders of this table. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).
tableType (Table Type)	An optional enumeration specifying the type or source of the table. Indicates whether the table is based off of an external data query, data in a worksheet, or from an xml data mapped to a worksheet.

Attributes	Description
	The possible values for this attribute are defined by the ST_TableType simple type (§18.18.78).
totalsRowBorderDxfId (Totals Row Border Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating what border formatting to apply to the totals row of this table. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).
totalsRowCellStyle (Totals Row Style)	A string representing the name of the cell style that is applied to the totals row cells of the table. If this string is missing or does not correspond to the name of a cell style, then the totals row style specified by the current table style should be applied. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
totalsRowCount (Totals Row Count)	An integer representing the number of totals rows that shall be shown at the bottom of the table. 0 means that the totals row is not shown. It is up to the spreadsheet application to determine if numbers greater than 1 are allowed. Unless the spreadsheet application has a feature where their might ever be more than one totals row, this number should not be higher than 1. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
totalsRowDxfId (Totals Row Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating which format to apply to the totals row of this table. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).
totalsRowShown (Totals Row Shown)	A Boolean indicating whether the totals row has ever been shown in the past for this table. True if the totals row has been shown, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Table) is located in §A.2. *end note*]

18.5.1.3 tableColumn (Table Column)

An element representing a single column for this table.

Attributes	Description
dataCellStyle (Data Area Style Name)	A string representing the name of the cell style that is applied to the cells in the data area of this table column. If this string is missing or does not correspond to the name of a cell style, then the data cell style specified by the current table style should be applied. This cell style should get precedence over the `dataCellStyle` defined by the `table`. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
dataDxfId (Data & Insert Row Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating which format to apply to the data area of this column. This formatting shall also apply to cells on the insert row for this column. The spreadsheet should fail to load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).
headerRowCellStyle (Header Row Cell Style)	A string representing the name of the cell style that is applied to the header row cell of this column. If this string is missing or does not correspond to the name of a cell style, then header row style specified by the current table style should be applied. This cell style should get precedence over the `headerRowCellStyle` defined by the `table`. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
headerRowDxfId (Header Row Cell Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating which format to apply to the header cell of this column. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).
id (Table Field Id)	An integer representing the unique identifier of this column. This shall be unique per table. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (Column name)	A string representing the unique caption of the table column. This is what shall be displayed in the header row in the UI, and is referenced through functions. This name

Attributes	Description
	shall be unique per table. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
queryTableFieldId (Query Table Field Id)	An integer representing the query table field ID corresponding to this table column. The relationship between this table and the corresponding query table is expressed in _rels part for this table. Each queryTableField has a unique id attribute, and this id is what is referenced here. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
totalsRowCellStyle (Totals Row Style Name)	A string representing the name of the cell style that is applied to the Totals Row cell of this column. If this string is missing or does not correspond to the name of a cell style, then the totals row cell style specified by the current table style should be applied. This cell style should get precedence over the totalsRowCellStyle defined by the table. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
totalsRowDxfId (Totals Row Format Id)	A zero based integer index into the differential formatting records <dxfs> in the styleSheet indicating which format to apply to the totals row cell of this column. The spreadsheet shall not load if this index is out of bounds. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).
totalsRowFunction (Totals Row Function)	An enumeration indicating which type of aggregation to show in the totals row cell for this column. The possible values for this attribute are defined by the ST_TotalsRowFunction simple type (§18.18.83).
totalsRowLabel (Totals Row Label)	A String to show in the totals row cell for this column. This string shall be ignored unless the totalsRowFunction="none" for this column, in which case it is displayed in the totals row. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
uniqueName (Unique Name)	An optional string representing the unique name of the table column. This string is used to bind the column to a field in a data table, so it should only be used when this table's

Attributes	Description
	`tableType` is `queryTable` or `xml`. This name shall be unique per table when it is used. For tables created from xml mappings, by default this should be the same as the name of the column, and should be kept in synch with the name of the column if that name is altered by the spreadsheet application. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TableColumn) is located in §A.2. *end note*]

18.5.1.4 tableColumns (Table Columns)

An element representing the collection of all table columns for this table.

Attributes	Description
count (Column Count)	An integer representing the total count of how many columns there are in this Table. This count shall include both query-defined and user-defined columns. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TableColumns) is located in §A.2. *end note*]

18.5.1.5 tableStyleInfo (Table Style)

This element describes which style is used to display this table, and specifies which portions of the table have the style applied.

Styles define a set of formatting properties that can be easily referenced by cells or other objects in the spreadsheet application. A style can be applied to a table, but tables can define specific parts of the table that should not have the style applied independently of other table parts. For instance a table can not apply the row striping of the style, and can not show the style's formatting of the last column, but will apply the column striping and the formatting to the first column.

Attributes	Description
name (Style Name)	A string representing the name of the table style to use with this table. If the style name does not correspond to the name of a table style then the spreadsheet

Attributes	Description
	application should use default style. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
showColumnStripes (Show Column Stripes)	A Boolean indicating whether column stripe formatting is applied. True when style column stripe formatting is applied, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showFirstColumn (Show First Column)	A Boolean indicating whether the first column in the table should have the style applied. True if the first column has the style applied, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showLastColumn (Show Last Column)	A Boolean indicating whether the last column in the table should have the style applied. True if the last column has the style applied, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showRowStripes (Show Row Stripes)	A Boolean indicating whether row stripe formatting is applied. True when style row stripe formatting is applied, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TableStyleInfo) is located in §A.2. *end note*]

18.5.1.6 totalsRowFormula (Totals Row Formula)

This element contains a custom formula for aggregating values from the column.

Each tableColumn has a `totalsRowFunction` that can be used for simple aggregations such as average, standard deviation, min, max, count, and others. If a more custom calculation is desired, then this element should be used, and the `totalsRowFunction` shall be set to "`custom`".

Attributes	Description
array (Array)	A Boolean value that indicates whether this formula is an array style formula. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TableFormula) is located in §A.2. *end note*]

18.5.1.7 xmlColumnPr (XML Column Properties)

An element defining the XML column properties for a column. This is only used for tables created from XML mappings.

[*Example*: Here is a simple example showing a table column that has an xmlColumPr.

```
<tableColumn id="1" uniqueName="SomeElement" name="SomeElement">
  <xmlColumnPr mapId="1" xpath="/xml/foo/element" xmlDataType="string"/>
</tableColumn>
```

end example]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>denormalized (Denormalized)</td><td>A Boolean that indicates whether the contents of the column have been filled down due to flattening. <code>True</code> if it has been filled down (denormilized), <code>false</code> otherwise.

This should be used when an XML mapping parent value has many children, and both the parent and child fields are mapped to their own column in the table.

[Example:
<code><Order ID="3"></code>
<code> <Item>Milk</Item></code>
<code> <Item>Bread</Item></code>
<code> <Item>Cheese</Item></code>
<code></Order></code>

The resulting table in the spreadsheet application would have two columns, the first with the item ID, filled down for each item in the table as follows:

Item ID   Item
3   Milk
3   Bread
3   Cheese

end example]

The possible values for this attribute are defined by the W3C XML Schema boolean datatype.</td></tr>
<tr><td>mapId (XML Map Id)</td><td>An integer representing the ID of the XML map this table field is associated with.

The XML map is defined in the xml maps part, and the Map element should have the corresponding <code>id</code>.</td></tr>
</table>

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
xmlDataType (XML Data Type)	An enumeration indicating which XML data type is used by this column. The possible values for this attribute are defined by the ST_XmlDataType simple type (§18.18.93).
xpath (XPath)	A string representing the XML path to the element this column is associated with. The spreadsheet application should support XPath limited to the following • The XPath is an absolute path to a simple-content element or attribute • [*Example*: "/ns1:root/ns1:row/ns1:column1" is supported if 'column1' is a child-most node, but not "/ns1:root/ns1:row" for the same document since 'row' is not a child. *end example*] • The XPath does not express axes, but uses the default child axes • [*Example*: "/ns1:root/ns1:row" is supported but not "/ns1:root/child::ns1:row *end example*] • An optional filter can be expressed at the end of the xpath • [*Example*: "/ns1:root/ns1:row/ns1:column1[@foo='abc']" is supported but not "/ns1:root/ns1:row[@foo='abc']/ns1:column1" *end example*] • The filter can only contain a single expression comparing a named attribute to a specific value • • Filters are only supported on XPaths that resolve to a simple-content element (not attributes) • • The named attribute shall be defined as an attribute of the simple-content element • • The attribute name shall be preceded by the short-hand (@) symbol representing the axes 'attribute' • [*Example*: "/ns1:root/ns1:row/ns1:column1[@foo='abc']" is supported not "/ns1:root/ns1:row/ns1:column1[attribute::foo='abc']" *end example*]

Attributes	Description
	• An arbitrary amount of white-space can be embedded between filter tokens • [*Example*: "/ns1:root/ns1:row/ns1:column1[@ foo='abc']" is permitted*end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_XmlColumnPr) is located in §A.2. *end note*]

18.5.2 Single Cell Tables

A single cell table is generated from an XML mapping. These really just look like regular cells to the spreadsheet user, but shall be implemented as Tables "under the covers."

These tables don't have the full set of properties that multi cell tables do. They only have the various XML properties, and core table properties (such as `id` and `name`) that are needed to create a table and XML mapping. For instance the formatting properties, totals row, and headers row don't exist for the single cell XML tables. The formatting for these cells is maintained in the style sheet.

18.5.2.1 singleXmlCell (Table Properties)

This element represents the table properties for a single cell XML table.

Attributes	Description
connectionId (Connection ID)	An integer representing an ID to indicate which connection from the connections collection is used by this table. This is only used for tables that are based off of xml maps The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
id (Table Id)	An integer representing the unique identifier of the table. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
r (Reference)	An A1 cell style reference to the cell that the single cell xml table occupies. The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SingleXmlCell) is located in §A.2. *end note*]

18.5.2.2 singleXmlCells (Single Cells)

This element is a container for a collection of singleXmlCell tables.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SingleXmlCells) is located in §A.2. *end note*]

18.5.2.3 xmlCellPr (Cell Properties)

This element stores the XML properties for the cell of a single cell xml table.

Attributes	Description
id (Table Field Id)	The unique identifier of the XML properties for the cell. This should always be set to the value of 1 since this id is always meant to be for a single cell xml table. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
uniqueName (Unique Table Name)	An optional string representing the unique name of the table column. By default this is the same as the name of the column. This should hold the name of the element or attribute that this cell is referring to in the XML. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_XmlCellPr) is located in §A.2. *end note*]

18.5.2.4 xmlPr (Column XML Properties)

This element represents the column properties for single cell XML tables.

Attributes	Description
mapId (XML Map Id)	An integer representing the ID of the XML map this table field is associated with. The XML map is defined in the xml maps part, and the Map element should have the corresponding id. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

Attributes	Description
xmlDataType (XML Data Type)	An enumeration indicating which XML data type is used by this column. The possible values for this attribute are defined by the ST_XmlDataType simple type (§18.18.93).
xpath (XPath)	A string representing the XML path to the element this column is associated with. The spreadsheet application should support XPath limited to the following: • The XPath is an absolute path to a simple-content element or attribute [*Example*: "/ns1:root/ns1:row/ns1:column1" is supported if 'column1' is a child-most node, but not "/ns1:root/ns1:row" for the same document since 'row' is not a child. *end example*] • The XPath does not express axes, but uses the default child axes [*Example*: "/ns1:root/ns1:row" is supported but not "/ns1:root/child::ns1:row *end example*] • An optional filter can be expressed at the end of the xpath [*Example*: "/ns1:root/ns1:row/ns1:column1[@foo='abc']" is supported but not "/ns1:root/ns1:row[@foo='abc']/ns1:column1" *end example*] • The filter can only contain a single expression comparing a named attribute to a specific value • Filters are only supported on XPaths that resolve to a simple-content element (not attributes) • The named attribute shall be defined as an attribute of the simple-content element • The attribute name shall be preceded by the short-hand (@) symbol representing the axes 'attribute' [*Example*: "/ns1:root/ns1:row/ns1:column1[@foo='abc']" is supported not "/ns1:root/ns1:row/ns1:column1[attribute::foo='abc']" *end example*] • An arbitrary amount of white-space can be embedded between filter tokens [*Example*:

Attributes	Description
	"/ns1:root/ns1:row/ns1:column1[@ foo='abc']" is permitted *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_XmlPr) is located in §A.2. *end note*]

18.6 Calculation Chain

The cells in a workbook can be calculated in different orders depending on various optimizations and dependencies. The calculation chain specifies the order in which the cells in a workbook were last calculated.

The calculation chain only deals with cells that require calculation - i.e., it only deals with cells that contain formulas. It does not track or express dependencies amongst the formulas, but rather only records the order in which the cells were last calculated.

The calculation chain order can change over time. One obvious way this can happen is that new formulas can be added, formulas can be removed or updated. The spreadsheet application can also optionally implement partial calculation as an optimization. Partial calculation is when the spreadsheet only recalculates cells that have had their dependencies or values changed. This way, when a number in a cell is changed, requiring an update to a dependent formula, only the cells that are affected by the update is recalculated, as opposed to recalculating the entire workbook.

The calculation chain described in this section is not required by the spreadsheet application, but can be used if the spreadsheet application finds it useful. It can be loaded by a spreadsheet application, or the application can optionally construct it at run time in memory based on formula dependencies. Since the xml data described in this section is not strictly required, the spreadsheet application is free to ignore the order in which the calculation chain specifies calculations - i.e., even if the calculation chain is loaded, the spreadsheet application is free to perform calculations in a different order at run time.

[*Example*:

Consider the following workbook (the formulas shown instead of cell values):

	A	B	C	D	E
1	1	=A1	=B1+A1	=C1+B1+A1	=D1+C1+B1+A1
2					
3					
4					
5	1	=A5	=B5+A5	=C5+B5+A5	=D5+C5+B5+A5
6					
7					

There is a constant entered in A1 and A5, and next to each of those cells are a series of cells which contain formulas that depend on those cells.

After entering the cells on the first row, and then the cells on the 5th row, the calc chain xml looks like this:

```
<calcChain xmlns="http://purl.oclc.org/ooxml/spreadsheetml/main">
  <c r="E5" i="1"/>
  <c r="D5"/>
  <c r="C5"/>
  <c r="B5"/>
  <c r="E1"/>
  <c r="D1"/>
  <c r="C1"/>
  <c r="B1"/>
</calcChain>
```

It is in this order because B1 was calced first (it was the first formula enterd in the workbook), followed by C1, D1, and so on. Then B5 was entered in the 5th row, followed by the other cells in the 5th row, ending with E5.

But, after a full recalculation, the spreadsheet application has realized that cells B5:E5 are on the same child chain, and cells B1:E1 are likewise on their own child chain. The xml now looks like this:

```
<calcChain xmlns="http://purl.oclc.org/ooxml/spreadsheetml/main">
  <c r="B1" i="1"/>
  <c r="C1" s="1"/>
  <c r="D1" s="1"/>
  <c r="E1" s="1"/>
  <c r="B5"/>
  <c r="C5" s="1"/>
  <c r="D5" s="1"/>
  <c r="E5" s="1"/>
</calcChain>
```

end example]

18.6.1 c (Cell)

This element represents a single cell, which shall contain a formula, in the calc chain. Cells are calculated in the same order as the c elements appear in the Calculation Chain part.

Attributes	Description
a (Array)	A Boolean flag indicating whether the cell's formula is an array formula. True if this cell's formula is an array formula, false otherwise. If there is a conflict between this attribute and the t attribute of the f element (§18.3.1.40), the t attribute takes precedence. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
	datatype.
i (Sheet Id)	A sheet Id of a sheet the cell belongs to. If this is omitted, it is assumed to be the same as the i value of the previous cell. The possible values for this attribute are defined by the W3C XML Schema int datatype.
l (New Dependency Level)	A Boolean flag indicating that the cell's formula starts a new dependency level. True if the formula starts a new dependency level, false otherwise. Starting a new dependency level means that all concurrent calculations, and child calculations, shall be completed - and the cells have new values - before the calc chain can continue. In other words, this dependency level might depend on levels that came before it, and any later dependency levels might depend on this level; but not later dependency levels can have any calculations started until this dependency level completes. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
r (Cell Reference)	An A-1 style reference to a cell. The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
s (Child Chain)	A Boolean flag indicating whether the cell's formula is on a child chain. True if this cell is part of a child chain, false otherwise. If this is omitted, it is assumed to be the same as the s value of the previous cell . A child chain is a list of calculations that occur which depend on the parent to the chain. There shall not be cross dependencies between child chains. Child chains are not the same as dependency levels - a child chain and its parent are all on the same dependency level. Child chains are series of calculations that can be independently farmed out to other threads or processors. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
t (New Thread)	A Boolean flag indicating whether the cell's formula starts a new thread. True if the cell's formula starts a new thread, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CalcCell) is located in §A.2. *end note*]

18.6.2 calcChain (Calculation Chain Info)

This element represents the root of the calculation chain.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CalcChain) is located in §A.2. *end note*]

18.7 Comments

A comment is a rich text note that is attached to and associated with a cell, separate from other cell content. Comment content is stored separate from the cell, and is displayed in a drawing object (like a text box) that is separate from, but associated with, a cell. Comments are used as reminders, such as noting how a complex formula works, or to provide feedback to other users. Comments can also be used to explain assumptions made in a formula or to call out something special about the cell.

[*Example*:

```
<comments>
  <authors>
    <author>Bob</author>
    <author>CBR</author>
  </authors>
  <commentList>
    <comment ref="D4" authorId="0">
      <text>
        <r>
          <rPr>
            <b/>
            <sz val="8"/>
            <color indexed="81"/>
            <rFont val="Calibri"/>
            <charset val="1"/>
            <scheme val="minor"/>
          </rPr>
          <t>Bob:</t>
        </r>
        <r>
          <rPr>
            <sz val="8"/>
           <color indexed="81"/>
           <rFont val="Calibri"/>
           <charset val="1"/>
           <scheme val="minor"/>
          </rPr>
```



```
          <t xml:space="preserve">Why such high expense?</t>
        </r>
      </text>
    </comment>
  </commentList>
</comments>
```

end example]

This xml sample displays a comment by "Bob" (bolded) that says, "Why such high expense?" (non bolded).

18.7.1 author (Author)

This element holds a string representing the name of a single author of comments. Every comment shall have an author. The maximum length of the author string is an implementation detail, but a good guideline is 255 chars.

The possible values for this element are defined by the ST_Xstring simple type (§22.9.2.19).

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. 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 (ST_Xstring) is located in §A.6.9. *end note*]

18.7.2 authors (Authors)

This element is a container that holds a list of comment author names. There can be many comment authors per sheet, but each author name shall be unique per sheet. The information for each author is stored only once for that sheet, and comments refer to the author by zero based index.

Note that there can be multiple lists of authors per workbook since each sheet contains its own comments part, and each comments part defines a list of authors for comments on that sheet.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Authors) is located in §A.2. *end note*]

18.7.3 comment (Comment)

This element represents a single user entered comment. Each comment shall have an author and can optionally contain richly formatted text.

Attributes	Description
authorId (Author Id)	Required. An unsigned integer which is used as the zero based index into the list of authors for this set of comments. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
guid (Unique Identifier for Comment)	Unique identifier for this comment. The attribute is required and shall be unique across all comments in shared workbooks. The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).
ref (Cell Reference)	Required. A string that serves as the A1 style reference to the cell that the comment is associated with. Shall only reference a single cell, not a range of cells, since comments are on a per cell basis. The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
shapeId (Shape ID)	Specifies the ID of the DrawingML shape that provides the visual representation of the comment. [*Example*: `<comment shapeId="10 " … >` *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Comment) is located in §A.2. *end note*]

18.7.4 commentList (List of Comments)

This element is a container that holds a list of comments for the sheet.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CommentList) is located in §A.2. *end note*]

18.7.5 commentPr (Comment Properties)

This element specifies the visual and positional properties of a comment.

[*Example*: In the following example, the comment's visual representation resizes with the

```
<comment … >
  <text> … </text>
  <commentPr autoFill="false">
    <anchor moveWithCells="true" sizeWithCells="true">
      <from> … </from>
      <to> … </to>
    <anchor>
  </commentPr>
</comment>
```

end example]

Attributes	Description
altText (Alternative Text)	Specifies alternative text for the object, for use by assistive technologies or applications. [*Example*: `<commentPr altText="Alternate text" … />` *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
autoFill (Automatic Fill Flag)	Specifies whether the object's fill formatting is provided automatically by the application. [*Example*: `<commentPr autoFill="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoLine (Automatic Line Flag)	Specifies whether the object's line formatting is provided automatically by the application. [*Example*: `<commentPr autoLine="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
autoScale (Automatic Text Scaling Flag)	Specifies whether the object's font is automatically scaled by the application when the object is resized. [*Example*: `<commentPr autoScale="true" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
colHidden (Hidden Column Flag)	Specifies that the column of the cell to which this comment points is hidden. [*Example*: `<commentPr colHidden="true" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
defaultSize (Default Size Flag)	Specifies whether the object is at its default size. [*Example*: `<commentPr defaultSize="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
disabled (Disabled Flag)	Specifies whether the object is allowed to run an attached macro. [*Example*: `<commentPr disabled="true" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
justLastX (Far East Alignment Flag)	Specifies that Far East alignment is set for the last line in the comment's text. Typically, justified text in Far East environments leaves the last line unjustified. Specifying this element also justifies the last line.

Attributes	Description
	[*Example*: `<commentPr justLastX="true" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
locked (Locked Flag)	Specifies that the object is locked when the sheet is protected. [*Example*: `<commentPr locked="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
lockText (Text Lock Flag)	Specifies that the object's text is locked. [*Example*: `<commentPr lockText="true" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
print (Print Flag)	Specifies whether the object is printed when the document is printed. [*Example*: `<commentPr print="false" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rowHidden (Hidden Row Flag)	Specifies that the row of the cell to which this comment points is hidden. [*Example*: `<commentPr rowHidden="true" … />` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
textHAlign (Text Horizontal Alignment)	Specifies the horizontal alignment of the comment's text field. [*Example*: `<commentPr textHAlign="center" … />` *end example*] The possible values for this attribute are defined by the ST_TextHAlign simple type (§18.18.80).
textVAlign (ext Vertical Alignment)	Specifies the vertical alignment of the comment's text field. [*Example*: `<commentPr textVAlign="center" … />` *end example*] The possible values for this attribute are defined by the ST_TextVAlign simple type (§18.18.81).
uiObject (UI Object Flag)	Specifies whether the object is a UI Object. [*Example*: `<commentPr uiObject="true" … />` *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element’s content model (CT_CommentPr) is located in §A.2. *end note*]

18.7.6 comments (Comments)

This element is the root container of a set of comments and comment authors for a particular sheet. Each set of comments for a sheet is stored in a separate xml part. The relationship part for a sheet defines a link to the correct comment part for that sheet.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Comments) is located in §A.2. *end note*]

18.7.7 text (Comment Text)

This element contains rich text which represents the text of a comment. The maximum length for this text is a spreadsheet application implementation detail. A recommended guideline is 32767 chars.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Rst) is located in §A.2. *end note*]

18.8 Styles

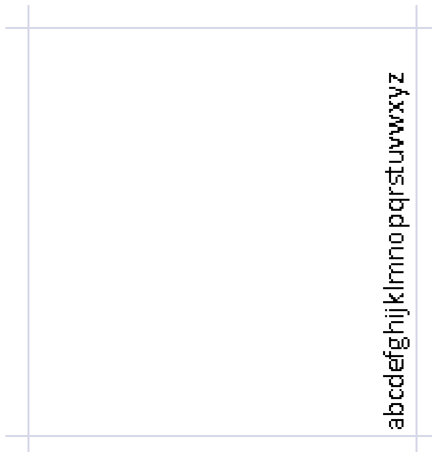
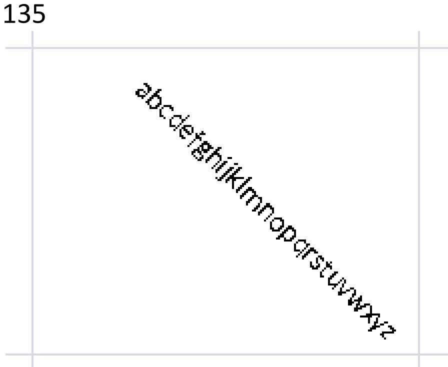
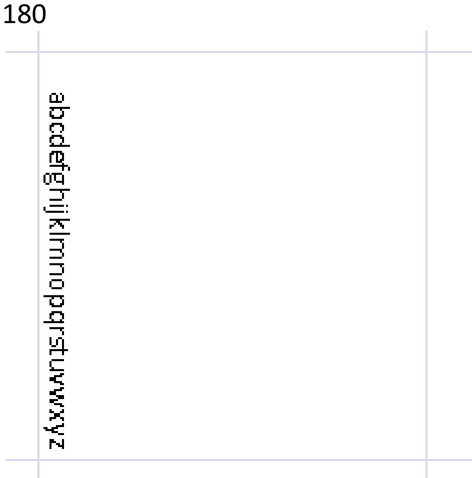
This subclause specifies the possible formatting information for the contents of the cells on a sheet in a SpreadsheetML document.

18.8.1 alignment (Alignment)

Formatting information pertaining to text alignment in cells. There are a variety of choices for how text is aligned both horizontally and vertically, as well as indentation settings, and so on.

Attributes	Description
horizontal (Horizontal Alignment)	Specifies the type of horizontal alignment in cells. The possible values for this attribute are defined by the ST_HorizontalAlignment simple type (§18.18.40).
indent (Indent)	An integer value, where an increment of 1 represents 3 spaces. Indicates the number of spaces (of the normal style font) of indentation for text in a cell. The number of spaces to indent is calculated as following: Number of spaces to indent = indent value * 3 [*Example*:For example, an indent value of 1 means that the text begins 3 space widths (of the normal style font) from the edge of the cell. *end example*] [*Note*: The width of one space character is defined by the font. *end note*] Only `left`, `right`, and `distributed` horizontal alignments are supported. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
justifyLastLine (Justify Last Line)	A boolean value indicating if the cells justified or distributed alignment should be used on the last line of text. (This is typical for East Asian alignments but not typical in other contexts.)

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
readingOrder (Reading Order)	An integer value indicating whether the reading order (bidirectionality) of the cell is left-to-right, right-to-left, or context dependent. 0 - Context Dependent - reading order is determined by scanning the text for the first non-whitespace character: if it is a strong right-to-left character, the reading order is right-to-left; otherwise, the reading order left-to-right. 1 - Left-to-Right- reading order is left-to-right in the cell, as in English. 2 - Right-to-Left - reading order is right-to-left in the cell, as in Hebrew. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
relativeIndent (Relative Indent)	An integer value (used only in a dxf element) to indicate the additional number of spaces of indentation to adjust for text in a cell. The possible values for this attribute are defined by the W3C XML Schema int datatype.
shrinkToFit (Shrink To Fit)	A boolean value indicating if the displayed text in the cell should be shrunk to fit the cell width. Not applicable when a cell contains multiple lines of text. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
textRotation (Text Rotation)	Text rotation in cells. Expressed in degrees. Values are in the range 0 to 180. The first letter of the text is considered the center-point of the arc. For 0 - 90, the value represents degrees above horizon. For 91-180 the degrees below the horizon is calculated as: [degrees below horizon] = 90 - textRotation. 0  45 

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>90

135

180

The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.</td></tr>
<tr><td>vertical (Vertical Alignment)</td><td>Vertical alignment in cells.</td></tr>
</table>

Attributes	Description
	The possible values for this attribute are defined by the ST_VerticalAlignment simple type (§18.18.88).
wrapText (Wrap Text)	A boolean value indicating if the text in a cell should be line-wrapped within the cell. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellAlignment) is located in §A.2. *end note*]

18.8.2 b (Bold)

Displays characters in bold face font style.

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_BooleanProperty) is located in §A.2. *end note*]

18.8.3 bgColor (Background Color)

Background color of the cell fill pattern. Cell fill patterns operate with two colors: a background color and a foreground color. These combine together to make a patterned cell fill.

Attributes	Description
auto (Automatic)	A boolean value indicating the color is automatic and system color dependent. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
indexed (Index)	Indexed color value. Only used for backwards compatibility. References a color in indexedColors. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rgb (Alpha Red Green Blue Color	Standard Alpha Red Green Blue color value (ARGB).

Attributes	Description
Value)	The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
theme (Theme Color)	A zero-based index into the `<clrScheme>` collection (§20.1.6.2), referencing a particular `<sysClr>` or `<srgbClr>` value expressed in the Theme part. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
tint (Tint)	Specifies the tint value applied to the color. If tint is supplied, then it is applied to the RGB value of the color to determine the final color applied. The tint value is stored as a double from -1.0 .. 1.0, where -1.0 means 100% darken and 1.0 means 100% lighten. Also, 0.0 means no change. In loading the RGB value, it is converted to HLS where HLS values are (0..HLSMAX), where HLSMAX is currently 255. [*Example*: Here are some examples of how to apply tint to color: **If (tint < 0)** Lum’ = Lum * (1.0 + tint) For example: Lum = 200; tint = -0.5; Darken 50% Lum‘ = 200 * (0.5) => 100 For example: Lum = 200; tint = -1.0; Darken 100% (make black) Lum‘ = 200 * (1.0-1.0) => 0 **If (tint > 0)** Lum‘ = Lum * (1.0-tint) + (HLSMAX – HLSMAX * (1.0-tint)) For example: Lum = 100; tint = 0.75; Lighten 75% Lum‘ = 100 * (1-.75) + (HLSMAX – HLSMAX*(1-.75)) = 100 * .25 + (255 – 255 * .25) = 25 + (255 – 63) = 25 + 192 = 217 For example: Lum = 100; tint = 1.0; Lighten 100% (make white) Lum‘ = 100 * (1-1) + (HLSMAX – HLSMAX*(1-1)) = 100 * 0 + (255 – 255 * 0) = 0 + (255 – 0) = 255 *end example*]

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Color) is located in §A.2. *end note*]

18.8.4 border (Border)

Expresses a single set of cell border formats (left, right, top, bottom, diagonal). Color is optional. When missing, 'automatic' is implied.

Attributes	Description
diagonalDown (Diagonal Down)	A boolean value indicating if the cell's diagonal border includes a diagonal line, starting at the top left corner of the cell and moving down to the bottom right corner of the cell. [*Example*: This example shows a thin diagonal down line:  *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
diagonalUp (Diagonal Up)	A boolean value indicating if the cell's diagonal border includes a diagonal line, starting at the bottom left corner of the cell and moving up to the top right corner of the cell. [*Example*: This example shows a thin diagonal up line:  *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
outline (Outline)	A boolean value indicating if left, right, top, and bottom borders should be applied only to outside borders of a cell range. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Border) is located in §A.2. *end note*]

18.8.5 borders (Borders)

This element contains borders formatting information, specifying all border definitions for all cells in the workbook.

[*Example*: In this example the first border definition specifies that there are no borders, the second definition specifies that a there is a thin bottom border and medium right border, and the third definition specifies that there is a double top border.

```
<borders count="3">
  <border>
    <begin/>
    <end/>
    <top/>
    <bottom/>
    <diagonal/>
  </border>
  <border>
    <begin/>
    <end style="medium">
      <color indexed="64"/>
    </end>
    <top/>
    <bottom style="thin">
      <color indexed="64"/>
    </bottom>
    <diagonal/>
  </border>
  <border>
    <begin/>
    <end/>
    <top style="double">
      <color auto="1"/>
    </top>
    <bottom/>
    <diagonal/>
  </border>
</borders>
```

end example]

Attributes	Description
count (Border Count)	Count of border elements. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Borders) is located in §A.2. *end note*]

18.8.6 bottom (Bottom Border)

This element specifies the color and line style for the bottom border of a cell.

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§18.18.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_BorderPr) is located in §A.2. *end note*]

18.8.7 cellStyle (Cell Style)

This element represents the name and related formatting records for a named cell style in this workbook.

Annex G contains a listing of cellStyles whose corresponding formatting records are implied rather than explicitly saved in the file. In this case, a builtinId attribute is written on the cellStyle record, but no corresponding formatting records are written.

For all built-in cell styles, the builtinId determines the style, not the name. For all cell styles, Normal is applied by default.

Attributes	Description
builtinId (Built-In Style Id)	The index of a built-in cell style: The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
customBuiltin (Custom Built In)	True indicates that this built-in cell style has been customized. By default built-in styles are not persisted when not in use. This flag indicates that a built-in style has been modified, and therefore should be saved with the workbook, even if not currently in use.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
hidden (Hidden Style)	If 'true' do not show this style in the application UI. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
iLevel (Outline Style)	Indicates that this formatting is for an outline style . When styles are applied to outline levels (using the outline feature), this value is set and the formatting specified on this cell style is applied to the corresponding level of the outline. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (User Defined Cell Style)	The name of the cell style. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
xfId (Format Id)	Zero-based index referencing an xf record in the cellStyleXfs collection. This is used to determine the formatting defined for this named cell style. The possible values for this attribute are defined by the ST_CellStyleXfId simple type (§18.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellStyle) is located in §A.2. *end note*]

18.8.8 cellStyles (Cell Styles)

This element contains the named cell styles, consisting of a sequence of named style records. A named cell style is a collection of direct or themed formatting (e.g., cell border, cell fill, and font type/size/style) grouped together into a single named style, and can be applied to a cell.

[*Example*: For example, "Normal", "Heading 1", "Title", and "20% Accent1" are named cell styles expressed below. They have builtInId's associated with them, and use xfId to reference the specific formatting elements pertaining to the particular style. The xfId is a zero-based index, referencing an xf record in the cellStyleXfs collection.

```
<cellStyles count="4">
  <cellStyle name="20% - Accent1" xfId="3" builtinId="30"/>
  <cellStyle name="Heading 1" xfId="2" builtinId="16"/>
  <cellStyle name="Normal" xfId="0" builtinId="0"/>
  <cellStyle name="Title" xfId="1" builtinId="15"/>
</cellStyles>
```

end example]

Attributes	Description
count (Style Count)	Count of style elements. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellStyles) is located in §A.2. *end note*]

18.8.9 cellStyleXfs (Formatting Records)

This element contains the master formatting records (xf's) which define the formatting for all named cell styles in this workbook. Master formatting records reference individual elements of formatting (e.g., number format, font definitions, cell fills, etc) by specifying a zero-based index into those collections. Master formatting records also specify whether to apply or ignore particular aspects of formatting. [*Example*: Whether to apply a border or not. *end example*]

A cell can have both direct formatting (e.g., bold) and a cell style (e.g., Explanatory) applied to it. Therefore, both the cell style xf records and cell xf records shall be read to understand the full set of formatting applied to a cell.

[*Example*: This example shows 4 master formatting records, each defining formatting for a named cell style (expressed in the cellStyles collection). Note that 0th record does not express any "apply" attributes, while the other records do express "apply" attribute values. For example, the last record specifies that number format, alignment, and protection formatting will not be applied to the cell, even when that information is specified in related formatting records.

```
<cellStyleXfs count="4">
  <xf numFmtId="0" fontId="0" fillId="0" borderId="0"/>
  <xf numFmtId="0" fontId="2" fillId="0" borderId="0" applyNumberFormat="0"
    applyFill="0" applyBorder="0" applyAlignment="0" applyProtection="0"/>
  <xf numFmtId="0" fontId="3" fillId="0" borderId="1" applyNumberFormat="0"
    applyFill="0" applyAlignment="0" applyProtection="0"/>
  <xf numFmtId="0" fontId="4" fillId="2" borderId="2" applyNumberFormat="0"
    applyAlignment="0" applyProtection="0"/>
</cellStyleXfs>
```

end example]

Attributes	Description
count (Style Count)	Count of cell style (xf) elements. The possible values for this attribute are defined by the W3C XML Schema unsignedInt

Attributes	Description
	datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellStyleXfs) is located in §A.2. *end note*]

18.8.10 cellXfs (Cell Formats)

This element contains the master formatting records (xf) which define the formatting applied to cells in this workbook. These records are the starting point for determining the formatting for a cell. Cells in the Sheet Part reference the xf records by zero-based index.

A cell can have both direct formatting (e.g., bold) and a cell style (e.g., Explanatory) applied to it. Therefore, both the cell style xf records and cell xf records shall be read to understand the full set of formatting applied to a cell.

Attributes	Description
count (Format Count)	Count of xf elements. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellXfs) is located in §A.2. *end note*]

18.8.11 colors (Colors)

Color information associated with this stylesheet. This collection is written whenever the legacy color palette has been modified (backwards compatibility settings) or a custom color has been selected while using this workbook.

When the color palette is modified, the indexedColors collection is written. When a custom color has been selected, the mruColors collection is written.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Colors) is located in §A.2. *end note*]

18.8.12 condense (Condense)

Macintosh compatibility setting. Represents special word/character rendering on Macintosh, when this flag is set. The effect is to condense the text (squeeze it together). SpreadsheetML applications are not required to render according to this flag.

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_BooleanProperty) is located in §A.2. *end note*]

18.8.13 diagonal (Diagonal)

This element specifies the color and line style for the diagonal border(s) of a cell, possibly including diagonally up and diagonally down. The line style for diagonal up and diagonal down lines shall be the same.

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§18.18.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_BorderPr) is located in §A.2. *end note*]

18.8.14 dxf (Formatting)

A single dxf record, expressing incremental formatting to be applied.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Dxf) is located in §A.2. *end note*]

18.8.15 dxfs (Formats)

This element contains the master differential formatting records (dxf's) which define formatting for all non-cell formatting in this workbook. Whereas xf records fully specify a particular aspect of formatting (e.g., cell borders) by referencing those formatting definitions elsewhere in the Styles part, dxf records specify incremental (or differential) aspects of formatting directly inline within the dxf element. The dxf formatting is to be applied on top of or in addition to any formatting already present on the object using the dxf record.

Attributes	Description
count (Format Count)	Count of dxf elements. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Dxfs) is located in §A.2. *end note*]

18.8.16 end (Trailing Edge Border)

This element specifies the color and line style for the trailing edge border of a cell (i.e., the right border for left-to-right cells and the left border for right-to-left cells)..

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§18.18.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_BorderPr) is located in §A.2. *end note*]

18.8.17 extend (Extend)

This element specifies a compatibility setting used for previous spreadsheet applications, resulting in special word/character rendering on those legacy applications, when this flag is set. The effect extends or stretches out the text. SpreadsheetML applications are not required to render according to this flag.

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_BooleanProperty) is located in §A.2. *end note*]

18.8.18 family (Font Family)

The font family this font belongs to. The font name overrides when there are conflicting values.

Attributes	Description
val (Value)	The font family this font belongs to. The possible values for this attribute are defined by the ST_FontFamily simple type (§18.18.94).

[*Note*: The W3C XML Schema definition of this element's content model (ST_FontFamily) is located in §A.2. *end note*]

18.8.19 fgColor (Foreground Color)

Foreground color of the cell fill pattern. Cell fill patterns operate with two colors: a background color and a foreground color. These combine together to make a patterned cell fill.

Attributes	Description
auto (Automatic)	A boolean value indicating the color is automatic and system color dependent. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
indexed (Index)	Indexed color value. Only used for backwards compatibility. References a color in indexedColors. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rgb (Alpha Red Green Blue Color Value)	Standard Alpha Red Green Blue color value (ARGB). The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
theme (Theme Color)	A zero-based index into the `<clrScheme>` collection (§20.1.6.2), referencing a particular `<sysClr>` or `<srgbClr>` value expressed in the Theme part. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
tint (Tint)	Specifies the tint value applied to the color. If tint is supplied, then it is applied to the RGB value of the color to determine the final color applied. The tint value is stored as a double from -1.0 .. 1.0, where -1.0 means 100% darken and 1.0 means 100% lighten. Also, 0.0 means no change. In loading the RGB value, it is converted to HLS where HLS values are (0..HLSMAX), where HLSMAX is currently 255. [*Example*: Here are some examples of how to apply tint to color: **If (tint < 0)** Lum' = Lum * (1.0 + tint) For example: Lum = 200; tint = -0.5; Darken 50% Lum' = 200 * (0.5) => 100 For example: Lum = 200; tint = -1.0; Darken 100% (make black)

Attributes	Description
	Lum‘ = 200 * (1.0-1.0) => 0 **If (tint > 0)** Lum‘ = Lum * (1.0-tint) + (HLSMAX – HLSMAX * (1.0-tint)) For example: Lum = 100; tint = 0.75; Lighten 75% Lum‘ = 100 * (1-.75) + (HLSMAX – HLSMAX*(1-.75)) = 100 * .25 + (255 – 255 * .25) = 25 + (255 – 63) = 25 + 192 = 217 For example: Lum = 100; tint = 1.0; Lighten 100% (make white) Lum‘ = 100 * (1-1) + (HLSMAX – HLSMAX*(1-1)) = 100 * 0 + (255 – 255 * 0) = 0 + (255 – 0) = 255 *end example*] The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Color) is located in §A.2. *end note*]

18.8.20 fill (Fill)

This element specifies fill formatting.

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Fill) is located in §A.2. *end note*]

18.8.21 fills (Fills)

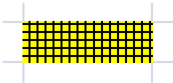
This element defines the cell fills portion of the Styles part, consisting of a sequence of fill records. A cell fill consists of a background color, foreground color, and pattern to be applied across the cell.

[*Example*: This cell has a yellow fill:

This is the corresponding XML:

```
<fill>
  <patternFill patternType="solid">
    <fgColor rgb="FFFFFF00"/>
    <bgColor indexed="64"/>
  </patternFill>
</fill>
```

This cell has a yellow fill with a thin horizontal crosshatch pattern applied (patternType = lightGrid):



This is the corresponding XML:

```
<fill>
  <patternFill patternType="lightGrid">
  <bgColor rgb="FFFFFF00"/>
  </patternFill>
</fill>
```

end example]

Attributes	Description
count (Fill Count)	Count of fill elements. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Fills) is located in §A.2. *end note*]

18.8.22 font (Font)

This element defines the properties for one of the fonts used in this workbook.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Font) is located in §A.2. *end note*]

18.8.23 fonts (Fonts)

This element contains all font definitions for this workbook.

[*Example*: This example expresses two fonts in the workbook. A Calibri family font, with font size of 11, and an Arial family font, with font size 12. The second font has strikethrough applied.

```
<fonts count="2">
  <font>
    <sz val="11"/>
    <color theme="1"/>
    <name val="Calibri"/>
    <family val="2"/>
    <scheme val="minor"/>
  </font>
```



```
    <font>
      <strike/>
      <sz val="12"/>
      <color theme="1"/>
      <name val="Arial"/>
      <family val="2"/>
    </font>
  </fonts>
```

end example]

Attributes	Description
count (Font Count)	Count of font elements. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Fonts) is located in §A.2. *end note*]

18.8.24 gradientFill (Gradient)

This element defines a gradient-style cell fill. Gradient cell fills can use one or two colors as the end points of color interpolation.

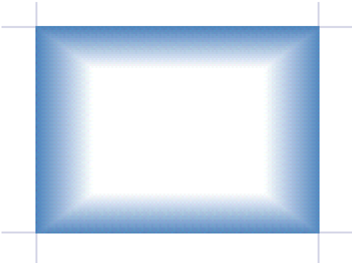
[*Example*:

This example shows a gradient cell fill, with color green at the top transitioning into blue at the bottom.

This is the XML:

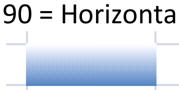
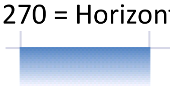
```
  <fill>
    <gradientFill degree="90">
      <stop position="0">
        <color rgb="FF92D050"/>
      </stop>
      <stop position="1">
        <color rgb="FF0070C0"/>
      </stop>
    </gradientFill>
  </fill>
```


This example shows a gradient cell fill, from the center. Note the left, right, top, and bottom values (and see explanation in the attribute section):



```
<fill>
  <gradientFill type="path" left="0.2" right="0.8" top="0.2" bottom="0.8">
    <stop position="0">
      <color theme="0"/>
    </stop>
    <stop position="1">
      <color theme="4"/>
    </stop>
  </gradientFill>
</fill>
```

end example]

Attributes	Description
bottom (Bottom Convergence)	This attribute is restricted to values ranging from 0 to 1. Specifies in percentage format (from the top to the bottom) the position of the bottom edge of the inner rectangle (color 1). For bottom, 0 means the bottom edge of the inner rectangle is on the top edge of the cell, and 1 means it is on the bottom edge of the cell. The possible values for this attribute are defined by the W3C XML Schema double datatype.
degree (Linear Gradient Degree)	Angle of the linear gradient - vertical, horizontal, diagonal. [*Example*: In these examples, color 1 is white and color 2 is blue. 90 = Horizontal & color 1 to color 2  270 = Horizontal & color 1 to color 2  0 = Vertical & color 1 to color 2

Attributes	Description
	 180 = Vertical & color 1 to color 2  45 = Diagonal Up & top to bottom (color 1 to color 2) 225 = Diagonal Up & bottom to top (color 1 to color 2) 135 = Diagonal Down & top to bottom (color 1 to color 2) 315 = Diagonal Down & bottom to top (color 1 to color 2) *end example*] The possible values for this attribute are defined by the W3C XML Schema double datatype.
left (Left Convergence)	This attribute is restricted to values ranging from 0 to 1. Specifies in percentage format (from the left to the right) the position of the left edge of the inner rectangle (color 1). For left, 0 means the left edge of the inner rectangle is on the left edge of the cell, and 1 means it is on the right edge of the cell. (applies to From Corner and From Center gradients). The possible values for this attribute are defined by the W3C XML Schema double datatype.
right (Right Convergence)	This attribute is restricted to values ranging from 0 to 1. Specifies in percentage format (from the left to the right) the position of the right edge of the inner rectangle (color 1). For right, 0 means the right edge of the inner rectangle is on the left edge of the cell, and 1 means it is on the right edge of the cell. (applies to From Corner and From Center gradients). The possible values for this attribute are defined by the W3C XML Schema double datatype.
top (Top Gradient Convergence)	This attribute is restricted to values ranging from 0 to 1. Specifies in percentage format (from the top to the bottom) the position of the top edge of the inner rectangle (color 1). For top, 0 means the top edge of the inner rectangle is on the top edge of the cell, and 1 means it is on the bottom edge of the cell. (applies to From Corner and From Center gradients). The possible values for this attribute are defined by the W3C XML Schema double datatype.
type (Gradient Fill	Type of this gradient fill.

Attributes	Description
Type)	The possible values for this attribute are defined by the ST_GradientType simple type (§18.18.37).

[*Note*: The W3C XML Schema definition of this element's content model (CT_GradientFill) is located in §A.2. *end note*]

18.8.25 horizontal (Horizontal Inner Borders)

This element specifies the color and line style for the horizontal inner border(s) of a range of cells. Used in the context of dxf elements only. [*Example*: see the borders definitions for **TableStyleMedium28**. *end example*]

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§18.18.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_BorderPr) is located in §A.2. *end note*]

18.8.26 i (Italic)

Displays characters in italic font style. The italic style is defined by the font at a system level and is not specified by ISO/IEC 29500.

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_BooleanProperty) is located in §A.2. *end note*]

18.8.27 indexedColors (Color Indexes)

A legacy indexing scheme for colors that is still required for some records, and for backwards compatibility with legacy formats.

This element contains a sequence of RGB color values that correspond to color indexes (zero-based). When using the default indexed color palette, the values are not written out, but instead are implied. When the color palette has been modified from default, then the entire color palette is written out.

Here is the table of default mappings from indexed color value to ARGB value. Note that 0-7 are redundant of 8-15 to preserve backwards compatibility.

Color Index	**ARGB Value**	[*Example*:
indexed="0"	00000000	
indexed="1"	00FFFFFF	
indexed="2"	00FF0000	
indexed="3"	0000FF00	
indexed="4"	000000FF	
indexed="5"	00FFFF00	
indexed="6"	00FF00FF	
indexed="7"	0000FFFF	
(none)	(none)	
indexed="8"	00000000	
indexed="9"	00FFFFFF	
indexed="10"	00FF0000	
indexed="11"	0000FF00	
indexed="12"	000000FF	
indexed="13"	00FFFF00	
indexed="14"	00FF00FF	
indexed="15"	0000FFFF	
indexed="16"	00800000	
indexed="17"	00008000	
indexed="18"	00000080	

indexed="19"	00808000
indexed="20"	00800080
indexed="21"	00008080
indexed="22"	00C0C0C0
indexed="23"	00808080
indexed="24"	009999FF
indexed="25"	00993366
indexed="26"	00FFFFCC
indexed="27"	00CCFFFF
indexed="28"	00660066
indexed="29"	00FF8080
indexed="30"	000066CC
indexed="31"	00CCCCFF
indexed="32"	00000080
indexed="33"	00FF00FF
indexed="34"	00FFFF00
indexed="35"	0000FFFF
indexed="36"	00800080
indexed="37"	00800000
indexed="38"	00008080
indexed="39"	000000FF
indexed="40"	0000CCFF
indexed="41"	00CCFFFF
indexed="42"	00CCFFCC
indexed="43"	00FFFF99

indexed="44"	0099CCFF	
indexed="45"	00FF99CC	
indexed="46"	00CC99FF	
indexed="47"	00FFCC99	
indexed="48"	003366FF	
indexed="49"	0033CCCC	
indexed="50"	0099CC00	
indexed="51"	00FFCC00	
indexed="52"	00FF9900	
indexed="53"	00FF6600	
indexed="54"	00666699	
indexed="55"	00969696	
indexed="56"	00003366	
indexed="57"	00339966	
indexed="58"	00003300	
indexed="59"	00333300	
indexed="60"	00993300	
indexed="61"	00993366	
indexed="62"	00333399	
indexed="63"	00333333	
indexed="64"	System Foreground	n/a
indexed="65"	System Background	n/a

[*Note*: The W3C XML Schema definition of this element's content model (CT_IndexedColors) is located in §A.2. *end note*]

18.8.28 mruColors (MRU Colors)

This element contains sequence of RGB values that correspond to custom colors selected by the user for this workbook.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MRUColors) is located in §A.2. *end note*]

18.8.29 name (Font Name)

This element specifies the face name of this font.

Attributes	Description
val (String Value)	A string representing the name of the font. If the font doesn't exist (because it isn't installed on the system), or the charset not supported by that font, then another font should be substituted. The string length for this attribute shall be 0 to 31 Unicode scalar values. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FontName) is located in §A.2. *end note*]

18.8.30 numFmt (Number Format)

This element specifies number format properties which indicate how to format and render the numeric value of a cell.

Following is a listing of number formats whose formatCode value is implied rather than explicitly saved in the file. In this case a numFmtId value is written on the xf record, but no corresponding numFmt element is written. Some of these Ids can be interpreted differently, depending on the UI language of the implementing application.

Ids not specified in the listing, such as 5, 6, 7, and 8, shall follow the number format specified by the formatCode attribute.

All Languages

ID	formatCode
0	General
1	0
2	0.00
3	#,##0

ID	formatCode
4	#,##0.00
9	0%
10	0.00%
11	0.00E+00
12	# ?/?
13	# ??/??
14	mm-dd-yy
15	d-mmm-yy
16	d-mmm
17	mmm-yy
18	h:mm AM/PM
19	h:mm:ss AM/PM
20	h:mm
21	h:mm:ss
22	m/d/yy h:mm
37	#,##0 ;(#,##0)
38	#,##0 ;[Red](#,##0)
39	#,##0.00;(#,##0.00)
40	#,##0.00;[Red](#,##0.00)
45	mm:ss
46	[h]:mm:ss
47	mmss.0
48	##0.0E+0
49	@

"General" Format

Some additional comments about the "General" number format are appropriate.

The primary goal when a cell is using "General" formatting is to render the cell content without user-specified guidance to the best ability of the application.

Alignment

(Specified for Left-to-Right mode)

- Strings: left aligned
- Boolean/error values: centered
- Numbers: right aligned
- Dates: do not follow the "General" format, instead automatically convert to date formatting.

Numbers

The application shall attempt to display the full number up to 11 digits (inc. decimal point). If the number is too large, the application shall attempt to show exponential format. If the number has too many significant digits, the display shall be truncated. The optimal method of display is based on the available cell width. If the number cannot be displayed using any of these formats in the available width, the application shall show "#" across the width of the cell.

Conditions for switching to exponential format:

3. The cell value shall have at least five digits for xE-xx
4. If the exponent is bigger than the size allowed, a floating point number cannot fit, so try exponential notation.
5. Similarly, for negative exponents, check if there is space for even one (non-zero) digit in floating point format.
6. Finally, if there isn't room for all of the significant digits in floating point format (for a negative exponent), exponential format shall display more digits if the exponent is less than -3. (The 3 is because E-xx takes 4 characters, and the leading 0 in floating point takes only 1 character. Thus, for an exponent less than -3, there is more than 3 additional leading 0's, more than enough to compensate for the size of the E-xx.)

Floating point rule:

For general formatting in cells, max overall length for cell display is 11, not including negative sign, but includes leading zeros and decimal separator.

zh-tw and zh-cn

ID	zh-tw formatCode	zh-cn formatCode
27	[$-404]e/m/d	yyyy"年"m"月"
28	[$-404]e"年"m"月"d"日"	m"月"d"日"
29	[$-404]e"年"m"月"d"日"	m"月"d"日"
30	m/d/yy	m-d-yy
31	yyyy"年"m"月"d"日"	yyyy"年"m"月"d"日"
32	hh"時"mm"分"	h"时"mm"分"

ID	zh-tw formatCode	zh-cn formatCode
33	hh"時"mm"分"ss"秒"	h"时"mm"分"ss"秒"
34	上午/下午 hh"時"mm"分"	上午/下午 h"时"mm"分"
35	上午/下午 hh"時"mm"分"ss"秒"	上午/下午 h"时"mm"分"ss"秒"
36	[$-404]e/m/d	yyyy"年"m"月"
50	[$-404]e/m/d	yyyy"年"m"月"
51	[$-404]e"年"m"月"d"日"	m"月"d"日"
52	上午/下午 hh"時"mm"分"	yyyy"年"m"月"
53	上午/下午 hh"時"mm"分"ss"秒"	m"月"d"日"
54	[$-404]e"年"m"月"d"日"	m"月"d"日"
55	上午/下午 hh"時"mm"分"	上午/下午 h"时"mm"分"
56	上午/下午 hh"時"mm"分"ss"秒"	上午/下午 h"时"mm"分"ss"秒"
57	[$-404]e/m/d	yyyy"年"m"月"
58	[$-404]e"年"m"月"d"日"	m"月"d"日"

zh-tw and zh-cn (with unicode values provided for language glyphs where they occur)

ID	zh-tw formatCode	zh-cn formatCode
27	[$-404]e/m/d	yyyy"5E74"m"6708"
28	[$-404]e"5E74"m"6708"d"65E5"	m"6708"d"65E5"
29	[$-404]e"5E74"m"6708"d"65E5"	m"6708"d"65E5"
30	m/d/yy	m-d-yy
31	yyyy"5E74"m"6708"d"65E5"	yyyy"5E74"m"6708"d"65E5"
32	hh"6642"mm"5206"	h"65F6"mm"5206"
33	hh"6642"mm"5206"ss"79D2"	h"65F6"mm"5206"ss"79D2"
34	4E0A5348/4E0B5348hh"6642"mm"5206"	4E0A5348/4E0B5348h"65F6"mm"5206"
35	4E0A5348/4E0B5348hh"6642"mm"5206"ss"79D2"	4E0A5348/4E0B5348h"65F6"mm"5206"ss"79D2"
36	[$-404]e/m/d	yyyy"5E74"m"6708"
50	[$-404]e/m/d	yyyy"5E74"m"6708"
51	[$-404]e"5E74"m"6708"d"65E5"	m"6708"d"65E5"

ID	zh-tw formatCode	zh-cn formatCode
52	4E0A5348/4E0B5348hh"6642"mm"5206"	yyyy"5E74"m"6708"
53	4E0A5348/4E0B5348hh"6642"mm"5206"ss"79D2"	m"6708"d"65E5"
54	[$-404]e"5E74"m"6708"d"65E5"	m"6708"d"65E5"
55	4E0A5348/4E0B5348hh"6642"mm"5206"	4E0A5348/4E0B5348h"65F6"mm"5206"
56	4E0A5348/4E0B5348hh"6642"mm"5206"ss"79D2"	4E0A5348/4E0B5348h"65F6"mm"5206"ss"79D2"
57	[$-404]e/m/d	yyyy"5E74"m"6708"
58	[$-404]e"5E74"m"6708"d"65E5"	m"6708"d"65E5"

ja-jp and ko-kr

ID	ja-jp formatCode	ko-kr formatCode
27	[$-411]ge.m.d	yyyy"年" mm"月" dd"日"
28	[$-411]ggge"年"m"月"d"日"	mm-dd
29	[$-411]ggge"年"m"月"d"日"	mm-dd
30	m/d/yy	mm-dd-yy
31	yyyy"年"m"月"d"日"	yyyy"년" mm"월" dd"일"
32	h"時"mm"分"	h"시" mm"분"
33	h"時"mm"分"ss"秒"	h"시" mm"분" ss"초"
34	yyyy"年"m"月"	yyyy-mm-dd
35	m"月"d"日"	yyyy-mm-dd
36	[$-411]ge.m.d	yyyy"年" mm"月" dd"日"
50	[$-411]ge.m.d	yyyy"年" mm"月" dd"日"
51	[$-411]ggge"年"m"月"d"日"	mm-dd
52	yyyy"年"m"月"	yyyy-mm-dd
53	m"月"d"日"	yyyy-mm-dd
54	[$-411]ggge"年"m"月"d"日"	mm-dd
55	yyyy"年"m"月"	yyyy-mm-dd
56	m"月"d"日"	yyyy-mm-dd
57	[$-411]ge.m.d	yyyy"年" mm"月" dd"日"
58	[$-411]ggge"年"m"月"d"日"	mm-dd

ja-jp and ko-kr (with unicode values provided for language glyphs where they occur)

ID	ja-jp formatCode	ko-kr formatCode
27	[$-411]ge.m.d	yyyy"5E74" mm"6708" dd"65E5"
28	[$-411]ggge"5E74"m"6708"d"65E5"	mm-dd
29	[$-411]ggge"5E74"m"6708"d"65E5"	mm-dd
30	m/d/yy	mm-dd-yy
31	yyyy"5E74"m"6708"d"65E5"	yyyy"B144" mm"C6D4" dd"C77C"
32	h"6642"mm"5206"	h"C2DC" mm"BD84"
33	h"6642"mm"5206"ss"79D2"	h"C2DC" mm"BD84" ss"CD08"
34	yyyy"5E74"m"6708"	yyyy-mm-dd
35	m"6708"d"65E5"	yyyy-mm-dd
36	[$-411]ge.m.d	yyyy"5E74" mm"6708" dd"65E5"
50	[$-411]ge.m.d	yyyy"5E74" mm"6708" dd"65E5"
51	[$-411]ggge"5E74"m"6708"d"65E5"	mm-dd
52	yyyy"5E74"m"6708"	yyyy-mm-dd
53	m"6708"d"65E5"	yyyy-mm-dd
54	[$-411]ggge"5E74"m"6708"d"65E5"	mm-dd
55	yyyy"5E74"m"6708"	yyyy-mm-dd
56	m"6708"d"65E5"	yyyy-mm-dd
57	[$-411]ge.m.d	yyyy"5E74" mm"6708" dd"65E5"
58	[$-411]ggge"5E74"m"6708"d"65E5"	mm-dd

th-th

ID	th-th formatCode
59	t0
60	t0.00
61	t#,##0
62	t#,##0.00
67	t0%
68	t0.00%
69	t# ?/?

ID	th-th formatCode
70	t# ??/??
71	ว/ด/ปปปป
72	ว-ดดด-ปป
73	ว-ดดด
74	ดดด-ปป
75	ช:นน
76	ช:นน:ทท
77	ว/ด/ปปปป ช:นน
78	นน:ทท
79	[ช]:นน:ทท
80	นน:ทท.0
81	d/m/bb

th-th (with unicode values provided for language glyphs where they occur)

ID	th-th formatCode
59	t0
60	t0.00
61	t#,##0
62	t#,##0.00
67	t0%
68	t0.00%
69	t# ?/?
70	t# ??/??
71	0E27/0E14/0E1B0E1B0E1B0E1B
72	0E27-0E140E140E14-0E1B0E1B
73	0E27-0E140E140E14
74	0E140E140E14-0E1B0E1B
75	0E0A:0E190E19
76	0E0A:0E190E19:0E170E17
77	0E27/0E14/0E1B0E1B0E1B0E1B 0E0A:0E190E19

ID	th-th formatCode
78	0E190E19:0E170E17
79	[0E0A]:0E190E19:0E170E17
80	0E190E19:0E170E17.0
81	d/m/bb

Attributes	Description
formatCode (Number Format Code)	The number format code for this number format. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
numFmtId (Number Format Id)	Id used by the master style records (xf's) to reference this number format. The possible values for this attribute are defined by the ST_NumFmtId simple type (§18.18.47).

[*Note*: The W3C XML Schema definition of this element's content model (CT_NumFmt) is located in §A.2. *end note*]

18.8.31 numFmts (Number Formats)

This element defines the number formats in this workbook, consisting of a sequence of numFmt records, where each numFmt record defines a particular number format, indicating how to format and render the numeric value of a cell.

[*Example*:

This cell is formatting as US currency:

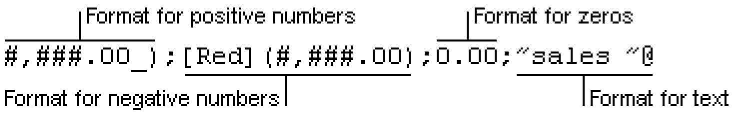
The XML expressing this format shows that the formatId is "166" and the decoded formatCode is `$#,##0.00`

```
<numFmts count="1">
  <numFmt numFmtId="166" formatCode="&quot;$&quot;#,##0.00"/>
</numFmts>
```

end example]

Number Format Codes

Up to four sections of format codes can be specified. The format codes, separated by semicolons, define the formats for positive numbers, negative numbers, zero values, and text, in that order. If only two sections are specified, the first is used for positive numbers and zeros, and the second is used for negative numbers. If only one section is specified, it is used for all numbers. To skip a section, the ending semicolon for that section shall be written.



The first section, "Format for positive numbers", is the format code that applies to the cell when the cell value contains a positive number.

The second section, "Format for negative numbers", is the format code that applies to the cell when the cell value contains a negative number.

The third section, "Format for zeros", is the format code that applies to the cell when the cell value is zero.

The fourth, and last, section, "Format for text", is the format code that applies to the cell when the cell value is text.

The & (ampersand) text operator is used to join, or concatenate, two values.

The following table describes the different symbols that are available for use in custom number formats.

Format symbol	Description and result
0	Digit placeholder. [*Example*: If the value 8.9 is to be displayed as 8.90, use the format #.00 *end example*]
#	Digit placeholder. This symbol follows the same rules as the 0 symbol. However, the application shall not display extra zeros when the number typed has fewer digits on either side of the decimal than there are # symbols in the format. [*Example*: If the custom format is #.##, and 8.9 is in the cell, the number 8.9 is displayed. *end example*]
?	Digit placeholder. This symbol follows the same rules as the 0 symbol. However, the application shall put a space for insignificant zeros on either side of the decimal point so that decimal points are aligned in the column. [*Example*: The custom format 0.0? aligns the decimal points for the numbers 8.9 and 88.99 in a column. *end example*]
. (period)	Decimal point.
%	Percentage. If the cell contains a number between 0 and 1, and the custom format 0% is used, the application shall multiply the number by 100 and add the percentage symbol in the cell.
, (comma)	Thousands separator. The application shall separate thousands by commas if the format contains a comma that is enclosed by number signs (#) or by zeros. A comma that follows a placeholder scales the number by one thousand. [*Example*: If the format is #.0,, and the cell value is 12,200,000 then the number 12.2 is displayed. *end example*]

Format symbol	Description and result
E- E+ e- e+	Scientific format. The application shall display a number to the right of the "E" symbol that corresponds to the number of places that the decimal point was moved. [*Example*: If the format is 0.00E+00, and the value 12,200,000 is in the cell, the number 1.22E+07 is displayed. If the number format is #0.0E+0, then the number 12.2E+6 is displayed. *end example*]
$-+():space	Displays the symbol. If it is desired to display a character that differs from one of these symbols, precede the character with a backslash (\). Alternatively, enclose the character in quotation marks. [*Example*: If the number format is (000), and the value 12 is in the cell, the number (012) is displayed. *end example*]
/	If this symbol is preceded and followed by a number symbol (0, #, and ?), it is interpreted as the fraction format symbol and will display the number in the format of a fraction. Otherwise, it is interpreted as the forward slash character and is displayed as such.
\	Displays the next character in the format. The application shall not display the backslash. [*Example*: If the number format is 0\!, and the value 3 is in the cell, the value 3! is displayed. *end example*]
*	Repeats the next character in the format enough times to fill the column to its current width. There shall not be more than one asterisk in one section of the format. If more than one asterisk appears in one section of the format, all but the last asterisk shall be ignored. [*Example*: if the number format is 0*x, and the value 3 is in the cell, the value 3xxxxxx is displayed. The number of x characters that are displayed in the cell varies based on the width of the column. *end example*]
_ (underline)	Skips the width of the next character. This is useful for lining up negative and positive values in different cells of the same column. [*Example*: The number format _(0.0_);(0.0) aligns the numbers 2.3 and -4.5 in the column even though the negative number is enclosed by parentheses. *end example*]
"text"	Displays whatever text is inside the quotation marks. [*Example*: The format 0.00 "dollars" displays 1.23 dollars when the value 1.23 is in the cell. *end example*]
@	Text placeholder. If text is typed in the cell, the text from the cell is placed in the format where the at symbol (@) appears. [*Example*: If the number format is "Bob "@" Smith" (including quotation marks), and the value "John" is in the cell, the value Bob John Smith is displayed. *end example*]

Text and spacing

Display both text and numbers

To display both text and numbers in a cell, enclose the text characters in double quotation marks (" ") or precede a single character with a backslash (\). Single quotation marks shall not be used to denote text. Characters inside double quotes, or immediately following backslash shall never be interpreted as part of the format code lexicon; instead they shall always be treated as literal strings. Remember to include the characters in the appropriate section of the format codes. [*Example*: Use the format $0.00" Surplus";$-0.00" Shortage" to display a positive amount as "$125.74 Surplus" and a negative amount as "$-125.74 Shortage." *end example*]

The following characters are displayed without the use of quotation marks.

$	Dollar sign				-	Minus sign
+	Plus sign				/	Slash mark
(	Left parenthesis				)	Right parenthesis
:	Colon				!	Exclamation point
^	Circumflex accent (caret)				&	Ampersand
'	Apostrophe				~	Tilde
{	Left curly bracket				}	Right curly bracket
<	Less-than sign				>	Greater-than sign
=	Equal sign					Space character

Include a section for text entry

If included, a text section shall be the last section in the number format. Include an "at" sign (@) in the section, precisely where the cell's text value should be displayed. If the @ character is omitted from the text section, text typed in the cell will not be displayed. To always display specific text characters with the typed text, enclose the additional text in double quotation marks (" "). [*Example*: If "June" is typed into the cell, and the text format is "gross receipts for "@ , then the cell will display "gross receipts for June". *end example*]

If the format does not include a text section, text entered in a cell is not affected by the format code.

Add spaces

To create a space that is the width of a character in a number format, include an underscore, followed by the character. [*Example*: When an underscore is followed with a right parenthesis, such as _), positive numbers line up correctly with negative numbers that are enclosed in parentheses because positive numbers are displayed with a blank space after them exactly the width of the right parenthesis character. *end example*]

Repeat characters

To repeat the next character in the format to fill the column width, include an asterisk (*) in the number format. [*Example*: Use 0*- to include enough dashes after a number to fill the cell, or use *0 before any format to include leading zeros. *end example*]

Decimal places, spaces, colors, and conditions

Include decimal places and significant digits

To format fractions or numbers with decimal points, include the following digit placeholders in a section. If a number has more digits to the right of the decimal point than there are placeholders in the format, the number rounds to as many decimal places as there are placeholders. If there are more digits to the left of the decimal point than there are placeholders, the extra digits are displayed. If the format contains only number signs (#) to the left of the decimal point, numbers less than 1 begin with a decimal point.

(number sign) displays only significant digits and does not display insignificant zeros.

0 (zero) displays insignificant zeros if a number has fewer digits than there are zeros in the format.

? (question mark) adds spaces for insignificant zeros on either side of the decimal point so that decimal points align when they are formatted with a fixed-width font, such as Courier New. ? can also be used for fractions that have varying numbers of digits.

To display	As	Use this code
1234.59	1234.6	####.#
8.9	8.900	#.000
.631	0.6	0.#
12 1234.568	12.0 1234.57	#.0#
44.398 102.65 2.8	44.398 102.65 2.8 (with aligned decimals)	???.???
5.25 5.3	5 1/4 5 3/10 (with aligned fractions)	# ???/???

Display a thousands separator

To display a comma as a thousands separator or to scale a number by a multiple of 1,000, include a comma in the number format.

To display	As	Use this code
12000	12,000	#,###
12000	12	#,
12200000	12.2	0.0,,

Specify colors

To set the text color for a section of the format, use the name of one of the following eight colors in square brackets in the section. The color code shall be the first item in the section.

[Black]			[Blue]			[Cyan]

[Green]			[Magenta]			[Red]
[White]			[Yellow]			

Instead of using the name of the color, the color index can be used, like this [Color3] for Red. Numeric indexes for color are restircted to the range from 1 to 56, which reference by index to the legacy color palette.

[*Note*: the default legacy color palette values are listed in §18.8.27. In the format codes, [Color1] refers to the color associated with indexed="8", or black (by default), [Color2] refers to the color associated with indexed="9", or white (by default), and so on up to [Color56] referring to the color associated with indexed="63". If the color palette has been customized from default values, then the colors associated with these indexes will reflect those customizations.

Specify conditions

To set number formats that are applied only if a number meets a specified condition, enclose the condition in square brackets. The condition consists of a comparison operator and a value. Comparison operators include: = Equal to; > Greater than; < Less than; >= Greater than or equal to, <= Less than or equal to, and <> Not equal to. [*Example*: The following format displays numbers that are less than or equal to 100 in a red font and numbers that are greater than 100 in a blue font.

```
[Red][<=100];[Blue][>100]
```

end example]

If the cell value does not meet any of the criteria, then pound signs ("#") are displayed across the width of the cell.

Currency, percentages, and scientific notation

Include currency symbols

To include currency symbols, place the currency symbol in the location it should when displayed.

Display percentages

To display numbers as a percentage of 100 — [*Example*: To display .08 as 8% or 2.8 as 280% *end example*]— include the percent sign (%) in the number format.

Display scientific notations

To display numbers in scientific format, use exponent codes in a section — [*Example*: E-, E+, e-, or e+. *end example*]

If a format contains a zero (0) or number sign (#) to the right of an exponent code, the application displays the number in scientific format and inserts an "E" or "e". The number of zeros or number signs to the right of a code determines the number of digits in the exponent. "E-" or "e-" places a minus sign by negative exponents. "E+" or "e+" places a minus sign by negative exponents and a plus sign by positive exponents.

Dates and times

Display days, months, and years

To display	As	Use this code
Months	1–12	m
Months	01–12	mm
Months	Jan–Dec	mmm
Months	January–December	mmmm
Months	J–D	mmmmm
Days	1–31	d
Days	01–31	dd
Days	Sun–Sat	ddd
Days	Sunday–Saturday	dddd
Years	00–99	yy
Years	date-base minimum value –9999	yyyy

See §18.17.4.1 for details on possible date systems.

Month versus minutes

If "m" or "mm" code is used immediately after the "h" or "hh" code (for hours) or immediately before the "ss" code (for seconds), the application shall display minutes instead of the month.

Display hours, minutes, and seconds

To display	As	Use this code
Hours	0–23	h

Hours	00–23	hh
Minutes	0–59	m
Minutes	00–59	mm
Seconds	0–59	s
Seconds	00–59	ss
Time	4 AM	h AM/PM
Time	4:36 PM	h:mm AM/PM
Time	4:36:03 P	h:mm:ss A/P
Time	4:36:03.75	h:mm:ss.00
Elapsed time (hours and minutes)	1:02	[h]:mm
Elapsed time (minutes and seconds)	62:16	[mm]:ss
Elapsed time (seconds and hundredths)	3735.80	[ss].00

Minutes versus month

The "m" or "mm" code shall appear immediately after the "h" or "hh" code or immediately before the "ss" code; otherwise, these will display as the month instead of minutes.

AM and PM

If the format contains AM or PM, the hour is based on the 12-hour clock, where "AM" or "A" indicates times from midnight until noon and "PM" or "P" indicates times from noon until midnight. Otherwise, the hour is based on the 24-hour clock.

Illegal date and time values

Cells formatted with a date or time format and which contain date or time values which do not meet the requirements specified shall show the pound sign ("#") across the width of the cell.

International Considerations

Format Code	Description
r	ja-jp/zh-tw only. When loading in ja-jp locale, code becomes "ee". When loading in zh-tw locale, code becomes "e".
rr	ja-jp/zh-tw only. When loading in ja-jp locale, code becomes "gggee". When loading in zh-tw locale, code becomes "e".
g	When loading in ja-jp locale: Single Roman character emperor reign When loading in zh-tw (Taiwan only) locale: treat same as "gg".
gg	When loading in ja-jp locale: Single Kanji character emperor reign When loading in zh-tw locale: Last era short name (since 1911)
ggg	When loading in ja-jp locale: Tow Kanji character emperor reign When loading in zh-tw locale: Last era long name (since 1911)
e	When loading in ja-jp locale: Era year When lading in zh-tw (Taiwan only) locale: Era year since 1912. If preceded by “g”, “gg”, or “ggg” then year of 1912, and years before 1912 are special, otherwise years before 1912 are Gregorian. OTHER locales: becomes "yyyy"
ee	When loading in ja-jp locale: Era year w/ leading zero When loading in zh-tw (Taiwan only) locale: Era year since 1911 OTHER locales: becomes "yy"
b2	Hijri calander
b1	Gregorian calendar
[$USD-409]	Specifies currency and locale/date system/number system information. Syntax is [$<Currency String>-<language info>]. Currency string is a string to use as a currency symbol. Language info is a 32-bit value entered in hexidecimal format. Language info format (byte 3 is most significant byte): Bytes 0,1: 16-bit Language ID (LID). Byte 2: Calendar type. High bit indicates that input is parsed using specified calendar. Byte 3: Number system type. High bit indicates that input is parsed using specified number system.

Format Code	Description
	Special language info values: 0xf800: System long date format 0xf400: System time format

Attributes	Description
count (Number Format Count)	Count of number format elements. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NumFmts) is located in §A.2. *end note*]

18.8.32 patternFill (Pattern)

This element is used to specify cell fill information for pattern and solid color cell fills. For solid cell fills (no pattern), fgColor is used. For cell fills with patterns specified, then the cell fill color is specified by the bgColor element.

Attributes	Description
patternType (Pattern Type)	Specifies the fill pattern type (including `solid` and none) Default is none, when missing. The possible values for this attribute are defined by the ST_PatternType simple type (§18.18.55).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PatternFill) is located in §A.2. *end note*]

18.8.33 protection (Protection Properties)

Contains protection properties associated with the cell. Each cell has protection properties that can be set. The cell protection properties do not take effect unless the sheet has been protected.

Attributes	Description
hidden (Hidden Cell)	A boolean value indicating if the cell is hidden. When the cell is hidden and the sheet on which the cell resides is protected, then the cell value is displayed in the cell grid location, but the contents of the cell will not be displayed in the formula bar. This is true for all types of cell content, including formula, text, or numbers. Therefore the cell A4 can contain a formula "=SUM(A1:A3)", but if the cell protection

Attributes	Description
	property of A4 is marked as hidden, and the sheet is protected, then the cell should display the calculated result [*Example*: "6" *end example*], but will not display the formula used to calculate the result. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
locked (Cell Locked)	A boolean value indicating if the cell is locked. When cells are marked as "locked" and the sheet is protected, then the options specified in the Sheet Part's sheetProtection element (§18.3.1.85) are prohibited for these cells. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CellProtection) is located in §A.2. *end note*]

18.8.34 rgbColor (RGB Color)

A single ARGB entry for the corresponding color index.

Attributes	Description
rgb (Alpha Red Green Blue)	Color value expressed in Alpha Red Green Blue format (ARGB). The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).

[*Note*: The W3C XML Schema definition of this element's content model (CT_RgbColor) is located in §A.2. *end note*]

18.8.35 scheme (Scheme)

Defines the font scheme, if any, to which this font belongs. When a font definition is part of a theme definition, then the font is categorized as either a major or minor font scheme component. When a new theme is chosen, every font that is part of a theme definition is updated to use the new major or minor font definition for that theme. Usually major fonts are used for styles like headings, and minor fonts are used for body and paragraph text.

Attributes	Description
val (Font Scheme)	Sets font scheme property. The possible values for this attribute are defined by the ST_FontScheme simple type (§18.18.33).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FontScheme) is located in §A.2. *end note*]

18.8.36 shadow (Shadow)

Macintosh compatibility setting. Represents special word/character rendering on Macintosh, when this flag is set. The effect is to render a shadow behind, beneath and to the right of the text. SpreadsheetML applications are not required to render according to this flag.

Attributes	Description
val (Value)	A boolean value for the property specified by the parent XML element. If omitted, the default value is `true`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_BooleanProperty) is located in §A.2. *end note*]

18.8.37 start (Leading Edge Border)

This element specifies the color and line style for the leading edge border of a cell (i.e., the left border for left-to-right cells and the right border for right-to-left cells).

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§18.18.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_BorderPr) is located in §A.2. *end note*]

18.8.38 stop (Gradient Stop)

One of a sequence of two or more gradient stops, constituting this gradient fill.

Attributes	Description
position (Gradient Stop Position)	Position information for this gradient stop. Interpreted exactly like gradientFill left, right, bottom, top. The position indicated here indicates the point where the color is pure. Before and and after this position the color can be in transition (or pure, depending on if this is the last stop or not). The possible values for this attribute are defined by the W3C XML Schema double

Attributes	Description
	datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_GradientStop) is located in §A.2. *end note*]

18.8.39 styleSheet (Style Sheet)

This is the root element of the Styles part.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Stylesheet) is located in §A.2. *end note*]

18.8.40 tableStyle (Table Style)

This element represents a single table style definition that indicates how a spreadsheet application should format and display a table.

Each of the tableStyle elements contains a collection of tableStyleElement elements that define formatting for a particular region of the table.

Annex G contains a listing of table styles whose tableStyleElement elements are implied rather than explicitly saved in the file. In this case, a name attribute is written on the tableStyle record, but no corresponding tableStyleElement elements are written.

All of the built-in, named table styles defined in Annex D shall be supported by applications that implement table styles.

[*Note*: Each of the table styles is made up of a collection of formatting definitions, each of which corresponds to a particular structured region of the table. An application can decide to support these built-in types, and can also decide to define more styles, each with their own definitions. An application can also decide whether the user is allowed to customize or further define additional table styles. *end note*]

Attributes	Description
count (Table Style Count)	Count of table style elements defined for this table style. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (Table Style Name)	Name of this table style. The possible values for this attribute are defined by the W3C XML Schema string datatype.
pivot (Pivot Style)	'True' if this table style should be shown as an available pivot table style. Not mutually exclusive with table - both can be `true`.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
table (Table)	True if this table style should be shown as an available table style. Not mutually exclusive with pivot - both can be true. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TableStyle) is located in §A.2. *end note*]

18.8.41 tableStyleElement (Table Style)

This element specifies formatting for one area of a table or PivotTable. Together the sequence of these elements makes up one entire Table style or PivotTable style definition.

The order in which table style element formatting is applied is as follows:

Table Style Element Order

- Whole Table
- First Column Stripe
- Second Column Stripe
- First Row Stripe
- Second Row Stripe
- Last Column
- First Column
- Header Row
- Total Row
- First Header Cell
- Last Header Cell
- First Total Cell
- Last Total Cell

1. For instance, row stripe formatting 'wins' over column stripe formatting, and both 'win' over whole table formatting.
1.
1. **PivotTable Style Element Order**

- Whole Table

- Page Field Labels
- Page Field Values
- First Column Stripe
- Second Column Stripe
- First Row Stripe
- Second Row Stripe
- First Column
- Header Row
- First Header Cell
- Subtotal Column 1
- Subtotal Column 2
- Subtotal Column 3
- Blank Row
- Subtotal Row 1
- Subtotal Row 2
- Subtotal Row 3
- Column Subheading 1
- Column Subheading 2
- Column Subheading 3
- Row Subheading 1
- Row Subheading 2
- Row Subheading 3
- Grand Total Column
- Grand Total Row

Attributes	Description
dxfId (Formatting Id)	Zero-based index to a dxf record in the dxfs collection, specifying differential formatting to use with this Table or PivotTable style element. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).
size (Band Size)	Number of rows or columns in a single band of striping. Applies only when type is `firstRowStripe`, `secondRowStripe`, `firstColumnStripe`, or `secondColumnStripe`. [*Example*: In this example, the `firstRowStripe` size is set to `2,` and the `secondRowStripe` size is set to `1`:

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><th>Column1</th><th>Column2</th><th>Column3</th><th>Column4</th></tr>
<tr><td>0.67</td><td>0.5</td><td>0.93</td><td>0.1</td></tr>
<tr><td>0.36</td><td>1</td><td>0.3</td><td>0.3</td></tr>
<tr><td>0.84</td><td>0.73</td><td>0.31</td><td>0.19</td></tr>
<tr><td>0.64</td><td>0.06</td><td>0.11</td><td>0.92</td></tr>
<tr><td>0.26</td><td>0.23</td><td>0.51</td><td>0.95</td></tr>
<tr><td>0.83</td><td>0.56</td><td>0.26</td><td>0.88</td></tr>
<tr><td>0.94</td><td>0.11</td><td>0.59</td><td>0.18</td></tr>
<tr><td>0.06</td><td>0.74</td><td>0.22</td><td>0.13</td></tr>
</table>
end example]
The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
</td></tr>
<tr><td>type (Table Style Type)</td><td> Identifies this table style element's type.
The possible values for this attribute are defined by the ST_TableStyleType simple type (§18.18.77). </td></tr>
</table>

[*Note*: The W3C XML Schema definition of this element's content model (CT_TableStyleElement) is located in §A.2. *end note*]

18.8.42 tableStyles (Table Styles)

This element represents a collection of Table style definitions for Table styles and PivotTable styles used in this workbook. It consists of a sequence of tableStyle records, each defining a single Table style.

A Table style is a collection of formatting that applies to structured regions of a Table or PivotTable [*Example*: make the header row & totals bold face, and apply light gray fill to alternating rows in the data portion of the table to achieve striped or banded rows. *end example*]

See the enumeration values in ST_TableStyleType for a listing of structured Table regions to which formatting can be applied, and which together make up a single Table style definition.

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>count (Table Style Count)</td><td> Count of table styles defined in this collection.
The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype. </td></tr>
<tr><td>defaultPivotStyle (Default Pivot Style)</td><td>Name of the default table style to apply to new PivotTables. This can be set by the user interface.</td></tr>
</table>

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema string datatype.
defaultTableStyle (Default Table Style)	Name of default table style to apply to new Tables. This can be set by the user interface. The possible values for this attribute are defined by the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TableStyles) is located in §A.2. *end note*]

18.8.43 top (Top Border)

This element specifies the color and line style for the top border of a cell.

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§18.18.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_BorderPr) is located in §A.2. *end note*]

18.8.44 vertical (Vertical Inner Border)

This element specifies the color and line style for the vertical inner border(s) of a range of cells. Used in the context of dxf elements only. [*Example*: see the borders definitions for **TableStyleMedium28**. *end example*]

Attributes	Description
style (Line Style)	The line style for this border. The possible values for this attribute are defined by the ST_BorderStyle simple type (§18.18.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_BorderPr) is located in §A.2. *end note*]

18.8.45 xf (Format)

A single xf element describes all of the formatting for a cell.

Attributes	Description
applyAlignment (Apply Alignment)	A boolean value indicating whether the alignment formatting specified for this xf should be applied. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyBorder (Apply Border)	A boolean value indicating whether the border formatting specified for this xf should be applied. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyFill (Apply Fill)	A boolean value indicating whether the fill formatting specified for this xf should be applied. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyFont (Apply Font)	A boolean value indicating whether the font formatting specified for this xf should be applied. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyNumberFormat (Apply Number Format)	A boolean value indicating whether the number formatting specified for this xf should be applied. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyProtection (Apply Protection)	A boolean value indicating whether the protection formatting specified for this xf should be applied. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
borderId (Border Id)	Zero-based index of the border record used by this cell format. The possible values for this attribute are defined by the ST_BorderId simple type (§18.18.2).
fillId (Fill Id)	Zero-based index of the fill record used by this cell format. The possible values for this attribute are defined by the ST_FillId simple type (§18.18.30).
fontId (Font Id)	Zero-based index of the font record used by this cell format. The possible values for this attribute are defined by the ST_FontId simple type (§18.18.32).
numFmtId	Id of the number format (numFmt) record used by this cell format.

Attributes	Description
(Number Format Id)	The possible values for this attribute are defined by the ST_NumFmtId simple type (§18.18.47).
pivotButton (Pivot Button)	A boolean value indicating whether the cell rendering includes a pivot table dropdown button. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
quotePrefix (Quote Prefix)	A boolean value indicating whether the text string in a cell should be prefixed by a single quote mark (e.g., 'text). In these cases, the quote is not stored in the Shared Strings Part. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
xfId (Format Id)	For xf records contained in cellXfs this is the zero-based index of an xf record contained in cellStyleXfs corresponding to the cell style applied to the cell. Not present for xf records contained in cellStyleXfs. The possible values for this attribute are defined by the ST_CellStyleXfId simple type (§18.18.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Xf) is located in §A.2. *end note*]

18.9 Metadata

A cell in a spreadsheet application can have metadata associated with it. Metadata is just a set of additional properties about the particular cell, and this metadata is stored in the metadata xml part.

There are two types of metadata: cell metadata and value metadata. Cell metadata contains information about the cell itself, and this metadata can be carried along with the cell as it moves (insert, shift, copy/paste, merge, unmerge, etc). Value metadata is information about the value of a particular cell. Value metadata properties can be propagated along with the value as it is referenced in formulas.

The file format is architected such that it supports both value and cell metadata, as well as even allowing for future extensions. Formulas, such as CUBEMEMBER() or CUBE*, shall make use of value metadata as part of the SpreadsheetML standard. So, only value metadata shall be implemented as it is used by MDX cube functions for retrieving data from OLAP data sources. The other parts are allowed for future extensibility.

See the informative material for background information on OLAP and the various CUBE* functions.

[*Example*: The CUBEMEMBER() function is used to return a specific member from an OLAP cube. The metadata will express the connection name (used as a friendly identifier for the external data connection to the OLAP server and cube), the MDX statement retrieving that member, and a set of operational attributes of the metadata that sepcify how it behaves in the spreadsheet application (i.e., whether it propagates through formula assignment, is able to be copy/pasted, etc).

```
<metadata xmlns="http://purl.oclc.org/ooxml/spreadsheetml/main">
  <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="My Connection"/>
    <s v="[Measures].[Internet Sales Amount]"/>
  </metadataStrings>
  <mdxMetadata count="1">
    <mdx n="0" f="m">
      <t c="1">
        <n x="1"/>
      </t>
    </mdx>
  </mdxMetadata>
  <valueMetadata count="1">
    <bk>
      <rc t="1" v="0"/>
    </bk>
  </valueMetadata>
</metadata>
```

As seen above, the metadata string table contains two entries: the name of the connection (My Connection), and the expression that returns the Internet Sales Amount member from the cube. The metadataType specifies that the metadata persists with assignment, cell merging, copy/pasting, shifting rows/columns, when the formatting or comments are deleted from the cell, and is assigned to the upper left most cell if a merged cell is split. In the valueMetadata collection, the metadata block specifies that the first `metadataType` is used, and indexes the first (0th) entry in the mdxMetadata collection. This MDX element in the mdxMetadata collection in turn specifies the cube function type (m= cube member) and an index into the string table that specifies the connection name. It also contains a tuple (t) element which specifies, via index into the string table, which tuple is returned. *end example*]

[*Note*: When copying a cell with metadata, and the cell contains an array formula, each pasted cell must contain the value from the corresponding position in the array and should contain the metadata corresponding to that cell. *end note*]

18.9.1 bk (Metadata Block)

This element represents a block of metadata records.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MetadataBlock) is located in §A.2. *end note*]

18.9.2 bk (Future Metadata Block)

This element represents a block of future metadata information. This is a location for storing feature extension information.

[*Note*: The W3C XML Schema definition of this element's content model (CT_FutureMetadataBlock) is located in §A.2. *end note*]

18.9.3 cellMetadata (Cell Metadata)

This element represents cell metadata information. Cell metadata is information metadata about a specific cell, and it stays tied to that cell position.

[*Note*: Applications should not use this for storing metadata, but instead us `valueMetadata`. Cell metadata is included for storing information from future application. *end note*]

Attributes	Description
count (Metadata Block Count)	Number of blocks of metadata records. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MetadataBlocks) is located in §A.2. *end note*]

18.9.4 futureMetadata (Future Metadata)

This element represents future metadata information.

Future data storage areas are xml storage areas that a later version of the spreadsheet application can store data into. So a V2 spreadsheet application can store data for new features that don't exist in the V1 version in a future storage area when saving to a format that the V1 version can open. The V1 version might be able to open the file, but won't necessarily be able to understand data that is stored in a future storage area. So the V1 version might ignore this data, but still round trip it in the file format so that V2 and V1 users can collaborate on the same spreadsheet.

Attributes	Description
count (Future Metadata Block Count)	Number of future metadata blocks. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (Metadata	Metadata type name.

Attributes	Description
Type Name)	The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_FutureMetadata) is located in §A.2. *end note*]

18.9.5 k (KPI MDX Metadata)

This element represents key performance indicator (KPI) MDX metadata. A KPI is typically an image that represents the state of some specific business measure at a given point in time. For instance, an image of a green traffic light indicating that customer satisfaction is good.

Attributes	Description
n (Member Unique Name Index)	Index of member unique name in string store. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
np (KPI Index)	Index of key performance indicator name in string store. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
p (KPI Property)	Key performance indicator property. The possible values for this attribute are defined by the ST_MdxKPIProperty simple type (§18.18.45).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MdxKPI) is located in §A.2. *end note*]

18.9.6 mdx (MDX Metadata Record)

This element represents a single record of MDX metadata information which can express a tuple, KPI, set, or member property.

Attributes	Description
f (Cube Function Tag)	This is an enumeration representing the function type of the calling cube function from the spreadsheet. The possible values for this attribute are defined by the ST_MdxFunctionType simple type (§18.18.44).
n (Connection	The zero based index of connection name in metadata string store, `metadataStrings`.

Attributes	Description
Name Index)	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Mdx) is located in §A.2. *end note*]

18.9.7 mdxMetadata (MDX Metadata Information)

This element represents a collection of specific MDX metadata records for the spreadsheet. This is used to build up the members, sets, tuples, KPIs, and member properties for the spreadsheet.

Attributes	Description
count (MDX Metadata Record Count)	Number of MDX metadata metadata records. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MdxMetadata) is located in §A.2. *end note*]

18.9.8 metadata (Metadata)

This element represents the root node for all metadata information in the spreadsheet.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Metadata) is located in §A.2. *end note*]

18.9.9 metadataStrings (Metadata String Store)

This element represents the metadata string store. This is a collection of strings that are used as a resource for the rest of the metadata part. It contains all the required OLAP strings used in the spreadsheet including the connection name, as well as MDX expressions identifying specific members and sets. It is indexed from individual metadata records so that the records can use these strings to build up the necessary MDX statements to retrieve the correct data from the OLAP cube.

Attributes	Description
count (MDX Metadata String Count)	Number of records in the string store. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MetadataStrings) is located in §A.2. *end note*]

18.9.10 metadataType (Metadata Type Information)

This element represents information about metadata on cells - it defines a specific set of behaviors that the metadata shall adhere to when subject to other spreadsheet operations.

In general, many of these attributes represent operations that can be performed on a cell that allow the metadata to remain associated with the cell. Operations that are set to 0 or false, will cause the metadata to be disassociated from the cell when that operation is performed.

Attributes	Description
adjust (Adjust Metadata)	A Boolean flag indicating that metadata corresponding to a particular cell needs to be notified when that cell's location is changed. [*Note*: This is included in the file format for future extensibility.*end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
assign (Metadata Formula Assignment)	A Boolean flag indicating whether metadata is propagated by formula assignment operation. True when metadata should be propagated by assignment, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
cellMeta (Cell Metadata)	A Boolean flag indicating whether metadata is cell metadata. True when the metadata is cell metadata, false otherwise - in the false case it is considered to be value metadata. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
clearAll (Metadata Clear All)	A Boolean flag indicating whether metadata survives a "Clear: All" operation. True if the metadata persists after a clear all, false otherwise. The Clear operations can be implemented by the run time application to provide an easy way to allow users to delete everything from a cell (Clear: All), remove only comments (Clear: Comments), only remove formats (Clear: Formats), or only remove the contents but leave the comments and formatting (Clear: Contents). Note these operations can also be performed by the user manually deleting each item. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
clearComments (Metadata Clear Comments)	A Boolean flag indicating whether metadata remains after comments have been cleared from the cell. True if the metadata persists after Clear:Comments, false otherwise. The Clear operations can be implemented by the run time application to provide an easy way to allow users to delete everything from a cell (Clear: All), remove only comments

Attributes	Description
	(Clear: Comments), only remove formats (Clear: Formats), or only remove the contents but leave the comments and formatting (Clear: Contents). Note these operations can also be performed by the user manually deleting each item. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
clearContents (Metadata Clear Contents)	A Boolean flag indicating whether metadata remains after the contents of a cell are removed. True if metadata persists after a "Clear: Contents" action, false otherwise. The Clear operations can be implemented by the run time application to provide an easy way to allow users to delete everything from a cell (Clear: All), remove only comments (Clear: Comments), only remove formats (Clear: Formats), or only remove the contents but leave the comments and formatting (Clear: Contents). Note these operations can also be performed by the user manually deleting each item. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
clearFormats (Metadata Clear Formats)	A Boolean flag indicating whether metadata remains after formatting is removed from a cell. True if metadata persists after a "Clear: Formats", false otherwise. The Clear operations can be implemented by the run time application to provide an easy way to allow users to delete everything from a cell (Clear: All), remove only comments (Clear: Comments), only remove formats (Clear: Formats), or only remove the contents but leave the comments and formatting (Clear: Contents). Note these operations can also be performed by the user manually deleting each item. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
coerce (Metadata Coercion)	A Boolean flag indicating whether value metadata can be removed when this metadata data type is coerced to another data type. True if the value metadata is removed upon coercion, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
copy (Metadata Copy)	A Boolean flag indicating whether metadata is copied with a cell. True if the metadata is copied to other cells when this cell is copied, false otherwise. This shall be set to `true` if the paste attributes for the `metadataType` are going to be used. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
delete (Metadata Cell Value Delete)	A Boolean flag indicating whether metadata survives deletion of a cell value. True when the metadata persists after the deletion of a cell value, false otherwise.

Attributes	Description
	This attribute is equivalent to the `clearContents` attribute. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
edit (Metadata Edit)	A Boolean flag indicating whether metadata survives the editing of the cell's value. True if the metadata remains unchanged after the cell's value edit, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ghostCol (Metadata Ghost Column)	A Boolean flag indicating whether metadata is copied to/from a ghost column. True when the metadata is copied to/from a ghost column, false otherwise. A ghost column is a single column that exists for the row header. It is not displayed to the end user. It is used to store default formatting for an entire row (i.e. the row gets the formatting for the corresponding cell in the ghost column). For instance, when an entire row is selected and a cell color is applied, this is stored once for the cell in the ghost column instead of for each cell in the row. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ghostRow (Metadata Ghost Row)	A Boolean flag indicating whether metadata is copied to/from a ghost row. True when the metadata is copied to/from a ghost row, false otherwise. A ghost row is a single row that exists for the column header. It is not displayed to the end user. It is used to store default formatting for an entire column (i.e. the column gets the formatting for the corresponding cell in the ghost row). For instance, when an entire column is selected and a cell color is applied, this is stored once for the cell in the ghost row instead of for each cell in the column. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
merge (Metadata Merge)	A Boolean flag indicating whether metadata survives cell merge. True if the metadata persists after a cell merge, false otherwise. It is up to the spreadsheet application on how to deal with conflicts when two cells that each have metadata are merged. The guidance here is to treat it the same as a 'regular' cell merge with the default behavior being that the data in the upper left cell wins. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
minSupportedVersion (Minimum Supported Version)	The earliest version of the spreadsheet application that supports this metadata type. The possible values for this attribute are defined by the W3C XML Schema unsignedInt

Attributes	Description
	datatype.
name (Metadata Type Name)	Represents the name of this particular metadata type. This name shall be unique amongst all other metadataTypes. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
pasteAll (Metadata Paste All)	A Boolean flag indicating whether metadata is populated to a new cell by "Paste: All". True when the metadata is populated on a Paste:All, false otherwise. Paste:All and regular paste should be implemented so that they are equivalent by the spreadsheet application. The copy flag shall be set to true for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
pasteBorders (Metadata Paste Borders)	A Boolean flag indicating whether metadata is populated with Paste: Borders. True when the metadata is populated when only borders are pasted, false otherwise. The copy flag shall be set to true for this paste behavior to be respected. [*Note*: The spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
pasteColWidths (Metadata Paste Column Widths)	A Boolean flag indicating whether metadata is populated by Paste: Column Widths. True if the metadata is populated when only column widths are pasted, false otherwise. The copy flag shall be set to true for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
pasteComments (Metadata Paste Comments)	A Boolean flag indicating whether metadata is populated by Paste: Comments. True when metadata is populated when only comments are pasted, false otherwise.

Attributes	Description
	The copy flag shall be set to true for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
pasteDataValidation (Metadata Paste Data Validation)	A Boolean flag indicating whether metadata is populated by Paste: Validation. True when metadata is populated when only data validation is pasted, false otherwise. The copy flag shall be set to true for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
pasteFormats (Metadata Paste Formats)	A Boolean flag indicating whether metadata is populated by Paste Special: Formats. True when metadata is populated when only formatting is pasted, false otherwise. The copy flag shall be set to true for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
pasteFormulas (Metadata Paste Formulas)	A Boolean flag indicating whether metadata is populated by Paste: Formulas. True when the metadata is populated when only formulas are pasted, false otherwise. The copy flag shall be set to true for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
pasteNumberFormats (Metadata Paste Number Formats)	A Boolean flag indicating whether metadata is populated with Paste: Number Formats. True when metadata is populated when only number formatting is pasted, false otherwise.

Attributes	Description
	The `copy` flag shall be set to `true` for this paste behavior to be respected. [*Note*: the spreadsheet application can implement special pasting behavior, such as pasting everything from a cell (paste all/normal paste), pasting only borders, pasting only comments, or pasting only any other specific cell property. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
pasteValues (Metadata Paste Special Values)	A Boolean flag indicating whether metadata is populated by Paste: Values. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rowColShift (Metadata Insert Delete)	A Boolean flag indicating whether metadata survives shifting due to row/column insertion/deletion. True if the metadata persists after the cell has been shifted, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
splitAll (Metadata Split All)	A Boolean flag indicating whether a merged cell split action has its metadata copied to all of the resulting cells. True if the metadata is copied to all new cells resulting from a split, false otherwise. If `splitFirst` is also set to `true`, `splitAll` wins - that is all the cells shall have the metadata copied to them. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
splitFirst (Meatadata Split First)	A Boolean flag indicating whether when a merged cell is split its metadata is copied to only the first resulting cell. True when the metadata from a split cell is only copied to the first resulting cell, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MetadataType) is located in §A.2. *end note*]

18.9.11 metadataTypes (Metadata Types Collection)

This element is a collection of metadata types.

Attributes	Description
count (Metadata	Number of metadata types.

Attributes	Description
Type Count)	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MetadataTypes) is located in §A.2. *end note*]

18.9.12 ms (Set MDX Metadata)

This element represents an MDX set.

Attributes	Description
c (Sort By Member Index Count)	Number of sort-by member indices. This is essentially the number of coordinates in the cube that this member is defined by. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ns (Set Definition Index)	Zero based index of the set definition in the metadata string store. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
o (Set Sort Order)	An enumeration specifying what sort order is used to sort the set. The possible values for this attribute are defined by the ST_MdxSetOrder simple type (§18.18.46).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MdxSet) is located in §A.2. *end note*]

18.9.13 n (Member Unique Name Index)

This element represents an index of a member unique name in metadata string store that is used to define the sort-by set.

Attributes	Description
s (String is a Set)	A Boolean flag indicating whether this string represents a set. True if the string represents a set, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
x (Index Value)	Value of the zero based index.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MetadataStringIndex) is located in §A.2. *end note*]

18.9.14 p (Member Property MDX Metadata)

This element represents an MDX member property.

Attributes	Description
n (Member Unique Name Index)	The zero based index of member unique name in the metadata string store. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
np (Property Name Index)	The zero based index of the property name in metadata string store. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MdxMemeberProp) is located in §A.2. *end note*]

18.9.15 rc (Metadata Record)

This element represents a reference to a specific metadata record.

Attributes	Description
t (Metadata Record Type Index)	A 1-based index to the metadata record type in `metadataTypes`. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
v (Metadata Record Value Index)	A zero based index to a specific metadata record. If the corresponding metadataType has `name="XLMDX"`, then this is an index to a record in mdxMetadata, otherwise this is an index to a record in the futureMetadata section whose `name` matches the name of the metadataType. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MetadataRecord) is located in §A.2. *end note*]

18.9.16 t (Tuple MDX Metadata)

This element represents an MDX tuple. A tuple is the intersection of two or more members of distinct dimensions in the cube. For instance, the three members (product, City, month) that are used to show the data point for how many products were sold.

The spreadsheet application should allow the values for the attributes of this element to be specified by the OLAP server.

Attributes	Description
b (Server Formatting Bold Font)	A Boolean flag indicating whether the bold style is applied. True if bold shall be applied, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
bc (Server Formatting Background Color)	Specifies the background color in RGB values. It is in hex and is read in the form of 0x00RRGGBB. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
c (Member Index Count)	The number of member expressions in the tuple. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ct (Server Formatting Culture Currency)	The culture tag to use for currency number format. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
fc (Server Formatting Foreground Color)	Represents the foreground color in RGB. It is in hex and is read in the form of 0x00RRGGBB. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
fi (Server Formatting Built-In Number Format Index)	Server formatting built-in number format index. This is an index into the spreadsheet application's built in number formats that is used to specify formatting. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
i (Server Formatting Italic Font)	A Boolean flag indicating that the italic formatting shall be applied. True if italic formatting is applied, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
si (Server Formatting String	Server formatting string index in the metadata string store, used to index to a string that contains information on how to format the number.

Attributes	Description
Index)	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
st (Server Formatting Strikethrough Font)	A Boolean flag indicating whether the strikethrough font style is applied. True if strikethrough shall be applied, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
u (Server Formatting Underline Font)	A Boolean flag indicating whether the underline font style is applied. True if underline shall be applied, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MdxTuple) is located in §A.2. *end note*]

18.9.17 valueMetadata (Value Metadata)

This element represents the value metadata information for the spreadsheet. It is essentially a collection of block elements that each define the value metadata for a particular cell. Cells in the workbook index into this collection, and each block element in this collection in turn references the `mdxMetadata` records.

Attributes	Description
count (Metadata Block Count)	Number of blocks of metadata records. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

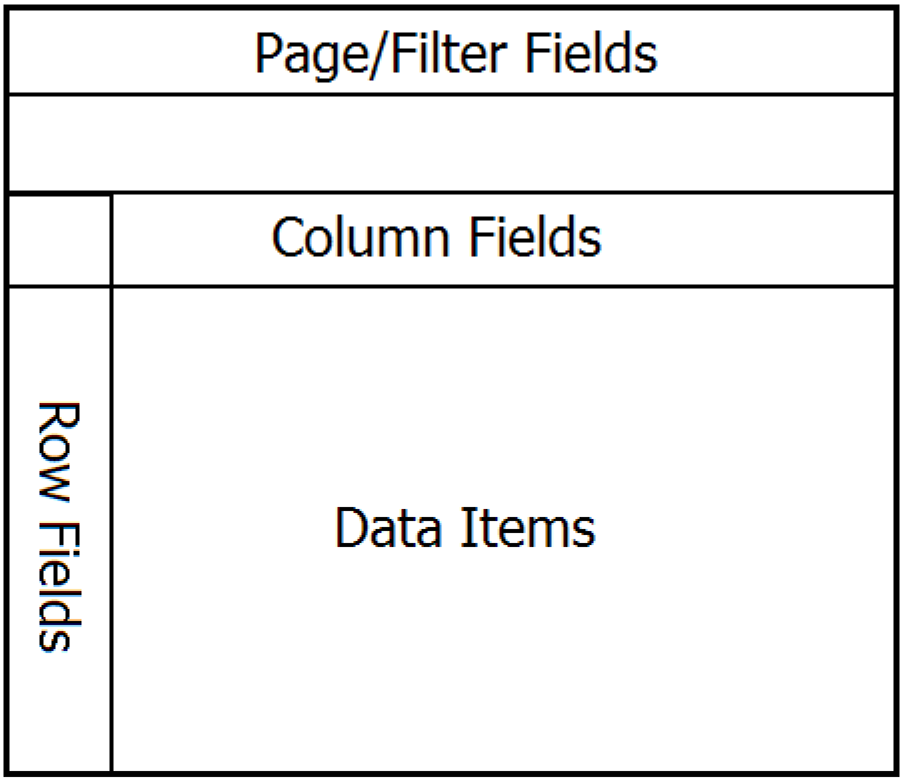
[*Note*: The W3C XML Schema definition of this element's content model (CT_MetadataBlocks) is located in §A.2. *end note*]

18.10 Pivot Tables

PivotTables display aggregated views of data easily and in an understandable layout. Hundreds or thousands of pieces of underlying information can be aggregated on row and column axes, revealing the meanings behind the data. PivotTable reports are used to organize and summarize your data in different ways. Creating a PivotTable report is about moving pieces of information around to see how they fit together. In a few gestures the pivot rows and columns can be moved into different arrangements and layouts.

A PivotTable object has a row axis area, a column axis area, a data area, and a page/report filter area. Additionally, PivotTables have a corresponding field list pane, or similar user interface, that displays all the fields of data that can be placed on one of the PivotTable areas. In SpreadsheetML, each PivotTable area maps to a collection of fields in the PivotTableDefinition that correspond to each area.

The following image shows the layout for the PivotTable areas.



[*Example*:

The following image shows a table of data in a worksheet.

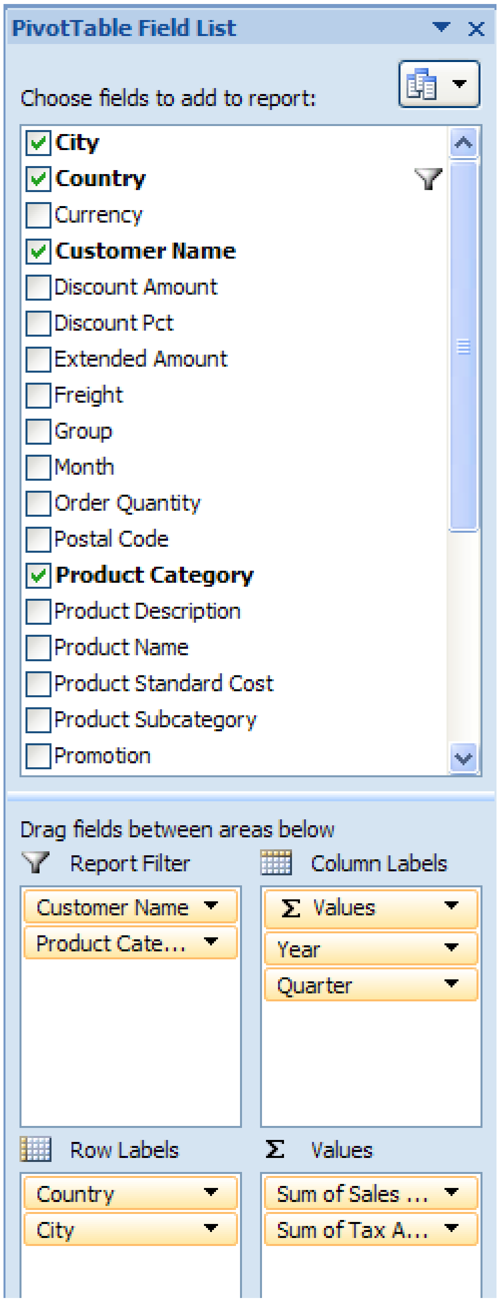
	A	C	F	H	I	O	P	Q	Z	AA	AB
1	Customer Name	Country	City	Product Category	Product Subcategory	Year	Quarter	Month	Sales Amount	Tax Amount	Freight
2	Michele Raman	Australia	Bendigo	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
3	Misty Raji	Australia	Bendigo	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
4	Tabitha E Arthur	Australia	Bendigo	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
5	Clarence D Rai	Australia	Bendigo	Bikes	Mountain Bikes	2001	3	July	3399.99	271.9992	84.9998
6	Jimmy L Moreno	Australia	Bendigo	Bikes	Mountain Bikes	2001	3	July	3399.99	271.9992	84.9998
7	Rob Verhoff	Australia	Bendigo	Bikes	Mountain Bikes	2001	3	July	3374.99	269.9992	84.3748
8	Levi Sai	Australia	Bendigo	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
9	Logan Gonzales	Australia	Brisbane	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
10	Dalton J Lee	Australia	Brisbane	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
11	Jessie J Ortega	Australia	Brisbane	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
12	Paul J. Shakespear	Australia	Caloundra	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
13	Joan R Martin	Australia	Caloundra	Bikes	Road Bikes	2001	3	September	699.0982	55.9279	17.4775
14	Casey Pal	Australia	Caloundra	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
15	Ethan G Coleman	Australia	Caloundra	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
16	Kendra Rubio	Australia	Caloundra	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
17	Bethany G Yuan	Australia	Cloverdal	Bikes	Mountain Bikes	2001	3	August	3399.99	271.9992	84.9998
18	Jasmine Wilson	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
19	Micah Wu	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
20	Warren L Zhang	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	July	699.0982	55.9279	17.4775
21	Ariana Stewart	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
22	Suzanne K Lu	Australia	Coffs Hart	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
23	Randall M Rubio	Australia	Cranbourr	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
24	Deborah K Kumar	Australia	Cranbourr	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
25	Krystal Holt	Australia	Cranbourr	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568
26	Patricia T Raman	Australia	Cranbourr	Bikes	Road Bikes	2001	3	August	3578.27	286.2616	89.4568
27	Wendy Dominguez	Australia	Cranbourr	Bikes	Mountain Bikes	2001	3	August	3374.99	269.9992	84.3748
28	Willie She	Australia	Darlinghu	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
29	Alan Zhu	Australia	Darlinghu	Bikes	Road Bikes	2001	3	September	3578.27	286.2616	89.4568
30	Dawn R Tang	Australia	Darlinghu	Bikes	Road Bikes	2001	3	July	3578.27	286.2616	89.4568

The following image shows a PivotTable summary of the worksheet table data.

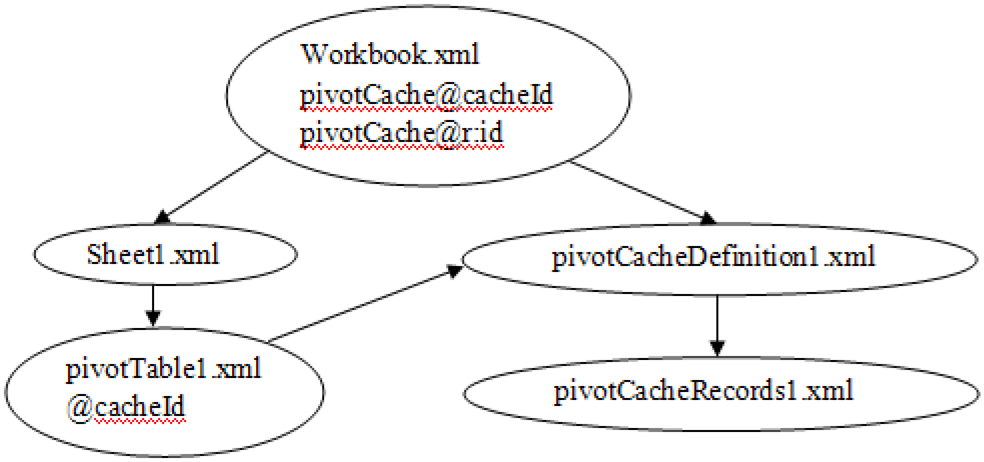
	A	B	C	D	E	F	G
1							
2		Country	(All)				
3		State	(All)				
4		City	(All)				
5							
6		Sum of Sales Amount	Column Labels				
7			2001				2001 Total
8				3		3 Total	
9		Row Labels	July	August	September		
10		Bikes	209652.9046	222538.2892	173993.5128	606184.7066	606184.7066
11		Mountain Bikes	64424.81	60899.82	10174.97	135499.6	135499.6
12		Road Bikes	145228.0946	161638.4692	163818.5428	470685.1066	470685.1066
13		Grand Total	209652.9046	222538.2892	173993.5128	606184.7066	606184.7066

The filter area consists of the "Country", "State", and "City" fields. The row area consists of the "Product Category" and "Product Subcategory" fields. "Bikes" belongs to the "Product Category" field and both "Mountain Bikes" and "Road Bikes" belong to the "Product Subcategory" field. The column consists of the "Year" ("2001"), "Quarter" ("3"), and "Month" ("July", "August", and "September") fields.

The following image shows the field list for the PivotTable in the previous image.



File Structure



The workbook points to (and owns the longevity of) the *pivotCacheDefinition* part, which in turn points to and owns the *pivotCacheRecords* part. The workbook also points to and owns the sheet part, which in turn points to and owns a *pivotTable* part definition, when a PivotTable is on the sheet. There can be multiple PivotTables on a sheet. The *pivotTable* part points to the appropriate *pivotCacheDefinition* which it is using. Since multiple PivotTables can use the same cache, the *pivotTable* part does not own the longevity of the *pivotCacheDefinition*.

The *pivotTable* part describes the particulars of the layout of the PivotTable on the sheet. It indicates what fields are on the row axis, the column axis, report filter, and values areas of the PivotTable. It also indicates formatting information about the PivotTable. If conditional formatting has been applied to the PivotTable, that is also expressed in the *pivotTable* part.

Outline of XML for pivotTableDefinition

```
<pivotTableDefinition>
  <location/>
  <pivotFields/>
  <rowFields/>
  <rowItems/>
  <colFields/>
  <colItems/>
  <pageFields/>
  <dataFields/>
  <conditionalFormats/>
  <pivotTableStyleInfo/>
</pivotTableDefinition>
```

The *pivotCacheRecords* part contains the underlying data to be aggregated. It is a cache of the source data.

Outline of XML for pivotCacheRecords


```
<pivotCacheRecords/>
  <r/>
</pivotCacheRecords>
```

The *pivotCacheDefinition* part defines each field in the *pivotCacheRecords* part, including field name and information about the data contained in the field. The *pivotCacheDefinition* part also defines pivot items that are shared among the *pivotTableDefinition* and *pivotCacheRecords* parts.

Outline of XML for pivotCacheDefinition

```
<pivotCacheDefinition>
  <cacheSource/>
  <cacheFields>
    <cacheField>
      <sharedItems>
        <d/>
      </sharedItems>
      <fieldGroup/>
    </cacheField>
  </cacheFields>
</pivotCacheDefinition>
```

18.10.1 Pivot Tables

This section describes the definition of PivotTables in SpreadsheetML.

18.10.1.1 autoSortScope (AutoSort Scope)

Represents the sorting scope for the PivotTable.

[*Note*: The W3C XML Schema definition of this element's content model (CT_AutoSortScope) is located in §A.2. *end note*]

18.10.1.2 b (Boolean)

Represents a boolean value for an item in the PivotTable.

Attributes	Description
c (Caption)	Specifies the caption for the item. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
cp (Member Property Count)	Specifies the number of property values for this item. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
f (Calculated Item)	Specifies a boolean value that indicates whether this item has a calculated value.

Attributes	Description
	A value of `1` or `true` indicates the item has a calculated value. A value of `0` or `false` indicates the item does not have a calculated value. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of `1` or `true` indicates this item is unused. A value of `0` or `false` indicates this item is used. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
v (Value)	Specifies the value of the item. This attribute is required. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Boolean) is located in §A.2. *end note*]

18.10.1.3 cacheField (PivotCache Field)

Represent a single field in the PivotCache. This definition contains information about the field, such as its source, data type, and location within a level or hierarchy. The sharedItems element stores additional information about the data in this field. If there are no shared items, then values are stored directly in the pivotCacheRecords part.

[*Example*:

```
<cacheField name="Group" numFmtId="0">
  <sharedItems count="3">
    <s v="Pacific"/>
    <s v="North America"/>
    <s v="Europe"/>
  </sharedItems>
</cacheField>
```

end example]

Attributes	Description
caption (PivotCache Field Caption)	Specifies the caption of the cache field. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
databaseField (Database Field)	Specifies a boolean value that indicates whether this field came from the source database rather having been created by the application. A value of `1` or `true` indicates the field is from the source database. A value of `0` or `false` indicates the field was created by the application. [*Note*: This attribute could be used for a defined grouped or calculated field. In this case, source database fields should precede defined grouped or calculated fields. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
formula (Calculated Field Formula)	Specifies the formula for the calculated field. This formula is specified by the end-user. Calculated fields can perform calculations by using the contents of other fields in the PivotTable. In formulas you create for calculated fields or calculated items, you can use operators and expressions as you do in other worksheet formulas. You can use constants and refer to data from the PivotTable, but you cannot use cell references or defined names. You cannot use worksheet functions that require cell references or defined names as arguments, and you cannot use array functions. Further behaviors and restrictions apply to formulas for calculated fields: • Formulas for calculated fields operate on the sum of the underlying data for any fields in the formula. [*Example*: The formula =Sales * 1.2 multiplies the sum of the sales for each type and region by 1.2; it does not multiply each individual sale by 1.2 and then sum the multiplied amounts. *end example*] • Formulas cannot refer to totals. For more information about formulas see §18.17 in Formulas. For more information about defined names see §18.2.6 in Workbook. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
hierarchy (Hierarchy)	Specifies the hierarchy that this field is part of. The possible values for this attribute are defined by the W3C XML Schema int datatype.
level (Hierarchy Level)	Specifies the hierarchy level that this field is part of. The possible values for this attribute are defined by the W3C XML Schema unsignedInt

Attributes	Description
	datatype.
mappingCount (Member Property Count)	Specifies the number of property mappings for this field. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
memberPropertyField (Member Property Field)	Specifies a boolean value that indicates whether the field contains OLAP member property information. A value of `1` or `true` indicates this field contains OLAP member property information. A value of `0` or `false` indicates this field does not contain OLAP member property information. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
name (PivotCache Field Name)	Specifies the name of the cache field. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
numFmtId (Number Format Id)	Specifies the number format that is applied to all items in the field. Number formats are written to the styles part. For more information see §18.8.31 in Styles. [*Note*: Formatting information provided by cell table and by PivotTable need not agree. If the two formats differ, the cell-level formatting takes precedence. If you change the layout of the PivotTable, the PivotTable formatting will then take precedence. *end note*] The possible values for this attribute are defined by the ST_NumFmtId simple type (§18.18.47).
propertyName (Property Name)	Specifies the name of the property if this field is an OLAP property field. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
serverField (Server-based Field)	Specifies a boolean value that indicates whether the field is a server-based page field. A value of `1` or `true` indicates this field is a server-based page field. A value of `0` or `false` indicates this field is not a server-based page field. This attribute applies to ODBC sources only. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sqlType (SQL Data Type)	Specifies the SQL data type of the field. This attribute stores an ODBC data type and applies to ODBC data sources only. A value is supplied for this

Attributes	Description
	attribute only if it is provided to the application. The following are data types supported by ODBC. For a more information, see the ODBC specification. • 0 SQL_UNKNOWN_TYPE • 1 SQL_CHAR • 2 SQL_VARCHAR • -1 SQL_LONGVARCHAR • -8 SQL_WCHAR • -9 SQL_WVARCHAR • -10 SQL_WLONGVARCHAR • 3 SQL_DECIMAL • 2 SQL_NUMERIC • 5 SQL_SMALLINT • 4 SQL_INTEGER • 7 SQL_REAL • 6 SQL_FLOAT • 8 SQL_DOUBLE • -7 SQL_BIT • -6 SQL_TINYINT • -5 SQL_BIGINT • -2 SQL_BINARY • -3 SQL_VARBINARY • -4 SQL_LONGVARBINARY • 9 SQL_TYPE_DATE or SQL_DATE • 10 SQL_TYPE_TIME or SQL_TIME • 11 SQL_TYPE_TIMESTAMP or SQL_TIMESTAMP • 102 SQL_INTERVAL_MONTH • 101 SQL_INTERVAL_YEAR • 107 SQL_INTERVAL_YEAR_TO_MONTH • 103 SQL_INTERVAL_DAY • 104 SQL_INTERVAL_HOUR • 105 SQL_INTERVAL_MINUTE • 106 SQL_INTERVAL_SECOND • 108 SQL_INTERVAL_DAY_TO_HOUR • 109 SQL_INTERVAL_DAY_TO_MINUTE • 110 SQL_INTERVAL_DAY_TO_SECOND • 111 SQL_INTERVAL_HOUR_TO_MINUTE • 112 SQL_INTERVAL_HOUR_TO_SECOND • 113 SQL_INTERVAL_MINUTE_TO_SECOND • -11 SQL_GUID • -20 SQL_SIGNED_OFFSET • -22 SQL_UNSIGNED_OFFSET

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema int datatype.
uniqueList (Unique List Retrieved)	Specifies a boolean value that indicates whether the application was able to get a list of unique items for the field. The attribute only applies to PivotTables that use ODBC and is intended to be used in conjunction with optimization features in the application. [*Example*: the application can optimize memory usage when populating PivotCache records if it has a list of unique items for a field before all the records are retrieved from ODBC. *end example*] A value of `1` or `true` indicates the application was able to get a list of unique values for the field. A value of `0` or `false` indicates the application was unable to get a list of unique values for the field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CacheField) is located in §A.2. *end note*]

18.10.1.4 cacheFields (PivotCache Fields)

Represents the collection of field definitions in the source data.

[*Example*:

```
<cacheFields count="1">
  <cacheField name="Group" numFmtId="0">
    <sharedItems count="3">
      <s v="One"/>
      <s v="Two"/>
      <s v="Three"/>
    </sharedItems>
  </cacheField>
</cacheFields>
```

end example]

Attributes	Description
count (Field Count)	Specifies the number of fields in the cache. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CacheFields) is located in §A.2. *end note*]

18.10.1.5 cacheHierarchies (PivotCache Hierarchies)

Represents the collection of OLAP hierarchies in the PivotCache.

[*Example*:

```
<cacheHierarchies count="2">
  <cacheHierarchy uniqueName="[Account].[Account]" caption="Account"
    attribute="1" keyAttribute="1"
    defaultMemberUniqueName="[Account].[Account].[All Accounts]"
    allUniqueName="[Account].[Account].[All Accounts]"
    dimensionUniqueName="[Account]" count="0"/>
  <cacheHierarchy uniqueName="[Account].[Account Number]" caption="Account
    Number" attribute="1" defaultMemberUniqueName="[Account].[Account
    Number].[All Accounts]" allUniqueName="[Account].[Account Number].[All
    Accounts]" dimensionUniqueName="[Account]" count="0"/>
</cacheHierarchies>
```

end example]

Attributes	Description
count (Hierarchy Count)	Specifies the number of OLAP hierarchies in the cache. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CacheHierarchies) is located in §A.2. *end note*]

18.10.1.6 cacheHierarchy (PivotCache Hierarchy)

Represents an OLAP hierarchy in the PivotCache.

[*Example*:

```
<cacheHierarchy uniqueName="[Account].[Account Number]" caption="Account Number"
  attribute="1" defaultMemberUniqueName="[Account].[Account Number].[All
  Accounts]" allUniqueName="[Account].[Account Number].[All Accounts]"
  dimensionUniqueName="[Account]" count="0"/>
```

end example]

Attributes	Description
allCaption (Display Name of 'All')	Specifies the display name of the "all" member of this hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
allUniqueName (Unique Name of 'All')	Specifies the unique name of the "all" member of this hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
attribute (Attribute Hierarchy)	Specifies a boolean value that indicates whether this hierarchy is an attribute hierarchy. An attribute hierarchy is an OLAP member that is exposed as a flat, single-level hierarchy on the OLAP server. A value of `1` or `true` indicates this hierarchy is an attribute hierarchy. A value of `0` or `false` indicates this hierarchy is not an attribute hierarchy. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
caption (Hierarchy Display Name)	Specifies the display name of the hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
count (Levels Count)	Specifies the number of levels in this hierarchy. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
defaultMemberUniqueName (Default Member Unique Name)	Specifies the unique name of the default member of this hierarchy The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
dimensionUniqueName (Dimension Unique Name)	Specifies the unique name of the dimension to which this hierarchy belongs. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
displayFolder (Display Folder)	Specifies the display folder in which this hierarchy should be displayed. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
hidden (Hidden)	Specifies a boolean value that indicates whether the hierarchy is hidden. A value of `1` or `true` indicates this hierarchy is hidden. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
	datatype.
iconSet (KPI Icon Set)	Specifies the icon set to use to visualize a KPI trend or status expression. PivotTables use the icon sets available for conditional formatting in SpreadsheetML. See associated simple type definition for details. The following values are used by PivotTables: • no value: default iconset. For status KPI this corresponds to 3 traffic lights. For trend KPI this corresponds to 3-arrows. • 1: Variance Arrow - 3 arrow. • 2: 3 arrows • 3: Status Arrow Ascending - 5 arrows. • 4: Status Arrow Descending - 5 arrows • 5: Standard Arrow - 5 arrows gray. • 6: Traffic Light Single - 3 traffic lights 1. • 7: Traffic Light, Traffic Light Multiple - 3 traffic lights 2. • 8: Gauge Ascending - 5 quarters. • 9: Gauge Descending - 5 quarters. • 10: Thermometer, Cylinder, Smiley Face - 3 signs. • 11: Road Signs - 3 symbols. The possible values for this attribute are defined by the W3C XML Schema int datatype.
keyAttribute (Key Attribute Hierarchy)	Specifies a boolean value that indicates whether this hierarchy is the key attribute hierarchy in an OLAP dimension. A value of 1 or true indicates this hierarchy is the key attribute hierarchy in an OLAP dimension. A value of 0 or false indicates this hierarchy is not a key attribute hierarchy. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
measure (Measure Hierarchy)	Specifies a boolean value that indicates whether this hierarchy is a measure. A value of 1 or true indicates this hierarchy is a measure. A value of 0 or false indicates this hierarchy is not a measure. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
measureGroup (Measure Group Name)	Specifies the name of the measure group to which this hierarchy belongs. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
measures (Measures)	Specifies a boolean value that indicates whether this hierarchy contains all the measures. A value of 1 or true indicates this hierarchy contains all the measures.

Attributes	Description
	A value of 0 or false indicates this hierarchy does not contain all the measures. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
memberValueDatatype (Member Value Data Type)	Specifies the data type of the member value. This attribute stores an OLEDB data type. [*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/oledb/htm/oledbtype_indicators.asp *end note*] memberValueDataType is stored for key attribute hierarchies in order to tell when the application will offer date filtering instead of label filtering in OLAP PivotTables. Date filtering is only offered when the data type is Date/Time. memberValueDatatype="7" indicates a date/time data type. The possible values for this attribute are defined by the W3C XML Schema unsignedShort datatype.
oneField (One Field)	Specifies a boolean value that indicates whether this hierarchy is associated with only one field due to its position in the view. A value of 1 or true indicates this hierarchy is associated with only one field. A value of 0 or false indicates this field is not restricted to only one association due to its position in the user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
parentSet (Parent Set)	Specifies the parent hierarchy of the set. If the attribute is missing it means that the parent hierarchy is unknown or doesn't exist in the cache. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
set (Set)	Specifies a boolean value that indicates whether this hierarchy is a set. A value of 1 or true indicates this hierarchy is a set. A value of 0 or false indicates this hierarchy is not a set. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
time (Time)	Specifies a boolean value that indicates whether this hierarchy is of type time.

Attributes	Description
	A value of 1 or true indicates this hierarchy is of type time. A value of 0 or false indicates is of a different type. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
unbalanced (Unbalanced)	Specifies a boolean value that indicates whether this hierarchy is an unbalanced hierarchy. If value is not written, then this attribute either cannot be determined or does not apply to the current hierarchy. A value of 1 or true indicates this hierarchy is unbalanced. A value of 0 or false indicates is balanced. For more information on balanced hierarchies, see the documentation provided for your OLAP server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
unbalancedGroup (Unbalanced Group)	Specifies a boolean value that indicated whether the grouped version of this hierarchy is an unbalanced hierarchy. If value is not written, then this attribute either cannot be determined or does not apply to the current hierarchy. A value of 1 or true indicates this hierarchy is unbalanced when grouped. A value of 0 or false indicates is balanced when grouped. For more information on balanced hierarchies, see the documentation provided for your OLAP server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
uniqueName (Hierarchy Unique Name)	Specifies the unique name of the hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CacheHierarchy) is located in §A.2. *end note*]

18.10.1.7 cacheSource (PivotCache Source Description)

Represents the description of data source whose data is stored in the pivot cache. The data source refers to the underlying rows or database records that provide the data for a PivotTable. You can create a PivotTable report from a SpreadsheetML table, an external database (including OLAP cubes), multiple SpreadsheetML worksheets, or another PivotTable.

Quarter	Region	Sport	Sales
Qtr1	East	Golf	$5,000
Qtr1	East	Safari	$9,000
Qtr1	East	Tennis	$1,500
Qtr2	East	Golf	$2,000
Qtr2	East	Safari	$6,000
Qtr2	East	Tennis	$500
Qtr1	West	Golf	$3,500
Qtr1	West	Tennis	$6,000
Qtr2	West	Golf	$2,500
Qtr2	West	Tennis	$3,200

Information about the data source is stored in the connection element and is retrieved using the connectionId attribute.

[*Example*:

```
<cacheSource type="external" connectionId="1"/>
```

end example]

OLAP data sources are distinguished from other data sources in SpreadsheetML. OLAP records are not stored in the *pivotCacheRecords* part, whereas all records for non-OLAP data sources are stored in the cache.

Attributes	Description
connectionId (Connection Index)	Specifies the index to the workbook connection. This attribute is used when the cache type is 'External.' See §18.13.1 for more information about the connection element. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
type (Cache Type)	Specifies the cache type. The possible values for this attribute are defined by the ST_SourceType simple type (§18.18.75).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CacheSource) is located in §A.2. *end note*]

18.10.1.8 calculatedItem (Calculated Item)

Represents an item within a PivotTable field that uses a formula . The formula is specified in the formula attribute.

Calculations and options available for a PivotTable depend on whether the source data came from an OLAP database or another type of database. This complex type applies to non-OLAP external data or on worksheet data. See calculatedMember for information on calculations on OLAP data sources.

Attributes	Description
field (Field Index)	Specifies the index of the pivotField with which this calculated item is associated. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
formula (Calculated Item Formula)	Specifies the formula of the calculated item. In formulas you create for calculated items, you can use operators and expressions as you do in other worksheet formulas. You can use constants and refer to data from the PivotTable, but you cannot use cell references or defined names. You cannot use worksheet functions that require cell references or defined names as arguments, and you cannot use array functions. Further behaviors and restrictions apply to formulas for `calculatedItems`: • Formulas for calculated items operate on the individual records; the calculated item formula =Dairy *115% multiplies each individual sale of Dairy times 115%, after which the multiplied amounts are summarized together in the data area. • Formulas cannot refer to totals. • You can include the field name in a reference to an item. The item name shall be in square brackets. Use this format to avoid #NAME? errors when two items in two different fields in a report have the same name. • You can refer to an item by its position in the PivotTable as currently sorted and displayed. The item referred to in this way can change whenever the positions of items change or different items are displayed or hidden. Hidden items are not counted in this index. • You can use relative positions to refer to items. The positions are determined relative to the calculated item that contains the formula. If the position you give is before the first item or after the last item in the field, the formula results in a #REF! error. For more information about formulas see §18.17 in Formulas. For more information about defined names see §18.2.6 in Workbook. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CalculatedItem) is located in §A.2. *end note*]

18.10.1.9 calculatedItems (Calculated Items)

Represents the collection of calculated items.

Attributes	Description
count (Calculated Item Formula Count)	Specifies the number of calculated item formulas in the cache. The possible values for this attribute are defined by the W3C XML Schema unsignedInt

Attributes	Description
	datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CalculatedItems) is located in §A.2. *end note*]

18.10.1.10 calculatedMember (Calculated Member)

A calculated member is a member in an OLAP hierarchy for which the value is calculated by an OLAP server using a Multidimensional Expressions (MDX) expression. For PivotTables that are created from OLAP cubes the summarized values are calculated by an OLAP server before the SpreadsheetML application displays the results. In OLAP PivotTables, the consuming application cannot change the summary function used to calculate totals and subtotals.
Calculated members are defined by the Multidimensional Expressions (MDX) expression in the mdx attribute.

[*Example*:

```
<calculatedMembers count="1">
  <calculatedMember name="[Product].[Product Categories].[All
    Products].[Calculated Member]" mdx="'[Product].[Product Categories].[All
    Products].[Accessories]'" memberName="Calculated Member"
    hierarchy="[Product].[Product Categories]" parent="[Product].[Product
    Categories].[All Products]"/>
</calculatedMembers>
```

end example]

Attributes	Description
hierarchy (Hierarchy Name)	Specifies the name of the hierarchy to which the calculated member belongs. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
mdx (Calculated Member MDX Formula)	Specifies the MDX formula for the calculated member. [*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn2.microsoft.com/en-us/library/ms145595.aspx *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
memberName (OLAP Calculated Member Name)	Specifies the OLAP member name for the calculated member. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

Attributes	Description
name (Calculated Member Name)	Specifies the name of the calculated member. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
parent (Parent Name)	Specifies the name of the parent of the calculated member. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
set (Set)	Specifies a boolean value that indicates whether this calculated member describes a calculated set rather than a calculated member. A value of `1` or `true` indicates this is a calculated set. A value of `0` or `false` indicates this is a calculated member. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
solveOrder (Calculated Members Solve Order)	Specifies the order in which this calculated member is calculated in relation to other calculated members. The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CalculatedMember) is located in §A.2. *end note*]

18.10.1.11 calculatedMembers (Calculated Members)

Represents the collection of calculated members in an OLAP PivotTable.

[*Example*:

```
<calculatedMembers count="1">
  <calculatedMember name="[Product].[Product Categories].[All
    Products].[Calculated Member]" mdx="'[Product].[Product Categories].[All
    Products].[Accessories]'" memberName="Calculated Member"
    hierarchy="[Product].[Product Categories]" parent="[Product].[Product
    Categories].[All Products]"/>
</calculatedMembers>
```

end example]

Attributes	Description
count (Calculated Members Count)	Specifies the number of calculated members.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CalculatedMembers) is located in §A.2. *end note*]

18.10.1.12 chartFormat (PivotChart Format)

Represents the format defined in the PivotChart that is associated with this PivotTable.

[*Example*:

```
<sh:pivotTableDefinition xmlns:sh="…" name="PivotTable1" cacheId="0"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="0"
  applyPatternFormats="0" applyAlignmentFormats="0" applyWidthHeightFormats="1"
  dataCaption="Values" updatedVersion="3" minRefreshableVersion="3"
  showCalcMbrs="0" useAutoFormatting="1" colGrandTotals="0" itemPrintTitles="1"
  createdVersion="3" indent="0" outline="1" outlineData="1"
  multipleFieldFilters="0" chartFormat="1" fieldListSortAscending="1">
```

end example]

Attributes	Description
chart (Chart Index)	Specifies the index of the chart part to which the formatting applies. For more information see the DrawingML specification for more information on the chart part. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
format (Pivot Format Id)	Specifies the index of the pivot format that is currently in use. This index corresponds to a dxf element in the Styles part. For more information see the Styles section (§18.8). The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
series (Series Format)	Specifies a boolean value that indicates whether format applies to a series. A value of `1` or `true` indicates this format applies to a series. A value of `0` or `false` indicates this format applies to a data point. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ChartFormat) is located in §A.2. *end note*]

18.10.1.13 chartFormats (PivotChart Formats)

Represents the collection of formats applied to PivotChart.

[*Example*:

```
<sh:chartFormats count="4">
  <sh:chartFormat chart="0" format="0" series="1">
    <sh:pivotArea type="data" outline="0">
      <sh:references count="3">
        <sh:reference field="4294967294" count="1" selected="0">
          <sh:x v="0"/>
        </sh:reference>
        <sh:reference field="14" count="1" selected="0">
          <sh:x v="0"/>
        </sh:reference>
        <sh:reference field="15" count="1" selected="0">
          <sh:x v="2"/>
        </sh:reference>
      </sh:references>
    </sh:pivotArea>
  </sh:chartFormat>
  <sh:chartFormat chart="0" format="1" series="1">
    <sh:pivotArea type="data" outline="0">
      <sh:references count="3">
        <sh:reference field="4294967294" count="1" selected="0">
          <sh:x v="0"/>
        </sh:reference>
        <sh:reference field="14" count="1" selected="0">
          <sh:x v="0"/>
        </sh:reference>
        <sh:reference field="15" count="1" selected="0">
          <sh:x v="3"/>
        </sh:reference>
      </sh:references>
    </sh:pivotArea>
  </sh:chartFormat>
```



```
    <sh:chartFormat chart="0" format="2" series="1">
      <sh:pivotArea type="data" outline="0">
        <sh:references count="3">
          <sh:reference field="4294967294" count="1" selected="0">
            <sh:x v="1"/>
          </sh:reference>
          <sh:reference field="14" count="1" selected="0">
            <sh:x v="0"/>
          </sh:reference>
          <sh:reference field="15" count="1" selected="0">
            <sh:x v="2"/>
          </sh:reference>
        </sh:references>
      </sh:pivotArea>
    </sh:chartFormat>
    <sh:chartFormat chart="0" format="3" series="1">
      <sh:pivotArea type="data" outline="0">
        <sh:references count="3">
          <sh:reference field="4294967294" count="1" selected="0">
            <sh:x v="1"/>
          </sh:reference>
          <sh:reference field="14" count="1" selected="0">
            <sh:x v="0"/>
          </sh:reference>
          <sh:reference field="15" count="1" selected="0">
            <sh:x v="3"/>
          </sh:reference>
        </sh:references>
      </sh:pivotArea>
    </sh:chartFormat>
  </sh:chartFormats>
```

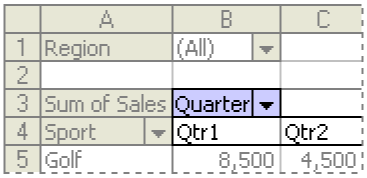
end example]

Attributes	Description
count (Format Count)	Specifies the number of formats in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ChartFormats) is located in §A.2. *end note*]

18.10.1.14 colFields (Column Fields)

Represents the collection of fields that are on the column axis of the PivotTable.



	A	B	C
1	Region	(All)	
2			
3	Sum of Sales	Quarter	
4	Sport	Qtr1	Qtr2
5	Golf	8,500	4,500

In the image above, the blue field is a column field.

[*Example*: In the following SpreadsheetML example, "Year", "Quarter" and "Month" are on the column axis of the PivotTable, in that order.

```
<colFields count="3">
  <field x="14"/>
  <field x="15"/>
  <field x="16"/>
</colFields>
```

end example]

Attributes	Description
count (Repeated Items Count)	Specifies the number of items in this collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ColFields) is located in §A.2. *end note*]

18.10.1.15 colHierarchiesUsage (Column OLAP Hierarchy References)

Represents the collection of references to OLAP hierarchies on the column axis of a PivotTable.

[*Example*:

```
<sh:colHierarchiesUsage count="2">
  <sh:colHierarchyUsage hierarchyUsage="33"/>
  <sh:colHierarchyUsage hierarchyUsage="-2"/>
</sh:colHierarchiesUsage>
```

end example]

Attributes	Description
count (Items Count)	Specifies the number of items in the collection.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ColHierarchiesUsage) is located in §A.2. *end note*]

18.10.1.16 colHierarchyUsage (Column OLAP Hierarchies)

Represents the collection of references to OLAP Hierarchies on the column axis of a PivotTable.

[*Example*:

<sh:colHierarchyUsage hierarchyUsage="33"/>

end example]

Attributes	Description
hierarchyUsage (Hierarchy Usage)	Specifies the reference to an OLAP hierarchy in a PivotTable. The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_HierarchyUsage) is located in §A.2. *end note*]

18.10.1.17 colItems (Column Items)

Represents the collection of column items of the PivotTable.

[*Example*: In the following SpreadsheetML example the item values are found in cells C6:H8. For example "2001" / "3" / "July" values are in C7:C9. Those are the first column item values and are referenced by the first <i> element below.

```
<colItems count="5">
  <i>
    <x/>
    <x/>
    <x/>
  </i>
  <i r="2">
    <x v="1"/>
  </i>
```



```
    <i r="2">
      <x v="2"/>
    </i>
    <i t="default" r="1">
      <x/>
    </i>
    <i t="default">
      <x/>
    </i>
  </colItems>
```

end example]

The first <i> collection represents all item values for the first column in the column axis area of the PivotTable. The first <x> in the first <i> corresponds to the first field in the columns area of the PivotTable, namely "Year". The implied index value of '0' on this <x> indicates that the item value for this first item in the column is the 0th item for this pivotField. The 0th item for this pivotField is itself an index to an item value into this field's shared items collection in the pivotCacheDefinition part, namely "2001".

The item values corresponding to the second and third <x> elements can be found in the same way, arriving at "3" for the second item value, and arriving at "July" for the third item value for this first column.

The second <i> collection expresses all 3 item values for the second column in the column axis area. The @r value of '2' indicates that the first two item values from the previous column is repeated here, which means that the first item value for this second column is "2001" again and the second item value for this second column is "3". The third item value is expressed by the only <x> element under this second <i> element, and without further explanation is understood to reference the item value "August".

Attributes	Description
count (Column Item Count)	Specifies the number of items on the column axis of the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_colItems) is located in §A.2. *end note*]

18.10.1.18 conditionalFormat (Conditional Formatting)

Represents the conditional formatting defined in the PivotTable.

Attributes	Description
priority (Priority)	Specifies the priority of PivotTable conditional formatting rule. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
scope (Conditional Formatting Scope)	Specifies the scope of PivotTable conditional formatting rule. The possible values for this attribute are defined by the ST_Scope simple type (§18.18.67).
type (Conditional Formatting Rule Type)	Specifies the type of PivotTable conditional formatting rule. See associated simple type definition for details. The possible values for this attribute are defined by the ST_Type simple type (§18.18.84).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ConditionalFormat) is located in §A.2. *end note*]

18.10.1.19 conditionalFormats (Conditional Formats)

Represents the collection of conditional formats applied to a PivotTable.

[*Example*:

```
<sh:conditionalFormats count="1">
  <sh:conditionalFormat priority="1">
    <sh:pivotAreas count="1">
      <sh:pivotArea type="data" collapsedLevelsAreSubtotals="1">
        <sh:references count="5">
          <sh:reference field="4294967294" count="1" selected="0">
            <sh:x v="0"/>
          </sh:reference>
          <sh:reference field="2" count="1" selected="0">
            <sh:x v="0"/>
          </sh:reference>
          <sh:reference field="14" count="1" selected="0">
            <sh:x v="0"/>
          </sh:reference>
```



```
          <sh:reference field="15" count="2" selected="0">
            <sh:x v="2"/>
            <sh:x v="3"/>
          </sh:reference>
        </sh:references>
      </sh:pivotArea>
    </sh:pivotAreas>
  </sh:conditionalFormat>
</sh:conditionalFormats>
```

end example]

Attributes	Description
count (Conditional Format Count)	Specifies the number of conditional formats defined for the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ConditionalFormats) is located in §A.2. *end note*]

18.10.1.20 consolidation (Consolidation Source)

Represents the description of the PivotCache source using multiple consolidation ranges. This element is used when the source of the PivotTable is a collection of ranges in the workbook. The ranges are specified in the rangeSets collection. The logic for how the application consolidates the data in the ranges is application-defined. [*Example*: the application might consolidate data based on its position in the worksheet that the end-user specifies. *end example*]

Attributes	Description
autoPage (Auto Page)	Specifies a boolean value that indicates whether the application will automatically create one additional page field to describe/qualify the source ranges. A value of `1` or `true` indicates the application will create an additional page field. A value of `0` or `false` indicates will not create an additional page field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Consolidation) is located in §A.2. *end note*]

18.10.1.21 d (Date Time)

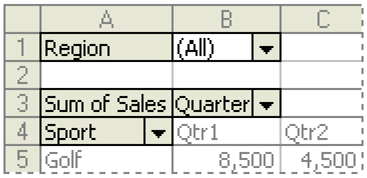
Represents a date-time value in the PivotTable.

Attributes	Description
c (Caption)	Specifies the caption for the item. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
cp (Member Property Count)	Specifies the number of member property values. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
f (Calculated Item Value)	Specifies a boolean value that indicates whether this is a calculated item value. A value of `1` or `true` indicates this is a calculated item value. A value of `0` or `false` indicates this is not a calculated item value. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of `1` or `true` indicates this is an unused item. A value of `0` or `false` indicates this item is used. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
v (Value)	Specifies the value of the item. The possible values for this attribute are defined by the W3C XML Schema dateTime datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DateTime) is located in §A.2. *end note*]

18.10.1.22 dataField (Data Field Item)

Represents a field from a source list, table, or database that contains data that is summarized in a PivotTable.



	A	B	C
1	Region	(All)	
2			
3	Sum of Sales	Quarter	
4	Sport	Qtr1	Qtr2
5	Golf	8,500	4,500

A data field represents data that is derived from a field in the source list or database. [*Example*: The Sport field, for example, might come from a column in the source list that is labeled Sport and contains the names of various sports (Golf, Tennis) for which the source list has sales figures. *end example*]

Source data can be taken from an SpreadsheetML list or range, an external database or cube, or another PivotTable. Data fields use summary functions to combine values from the underlying source data. You can also use custom calculations to compare data values, or add your own formulas that use elements of the report or other worksheet data.

[*Example*:

```
<dataFields count="1">
  <dataField name="Sum of Sales Amount" fld="25" baseField="0" baseItem="0"/>
</dataFields>
```

end example]

Attributes	Description
baseField ('Show Data As' Base Field)	Specifies the index to the base field when the ShowDataAs calculation is in use. The possible values for this attribute are defined by the W3C XML Schema int datatype.
baseItem ('Show Data As' Base Setting)	Specifies the index to the base item when the ShowDataAs calculation is in use. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
fld (Field)	Specifies the index to the field (<r>) in the pivotCacheRecords part that this data item summarizes. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (Data Field Name)	Specifies the name of the data field. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
numFmtId (Number Format Id)	Specifies the index to the number format applied to this data field. Number formats are written to the styles part. See the Styles section(§18.8) for more information on number formats. Formatting information provided by cell table and by PivotTable need not agree. If the two formats differ, the cell-level formatting takes precedence. If you change the layout

Attributes	Description
	the PivotTable, the PivotTable formatting will then take precedence. The possible values for this attribute are defined by the ST_NumFmtId simple type (§18.18.47).
showDataAs (Show Data As Display Format)	Specifies the display format for this data field. Formatting information provided by cell table and by PivotTable need not agree. If the two formats differ, the cell-level formatting takes precedence. If you change the layout the PivotTable, the PivotTable formatting will then take precedence. The possible values for this attribute are defined by the ST_ShowDataAs simple type (§18.18.70).
subtotal (Subtotal)	Specifies the aggregation function that applies to this data field. The possible values for this attribute are defined by the ST_DataConsolidateFunction simple type (§18.18.17).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DataField) is located in §A.2. *end note*]

18.10.1.23 dataFields (Data Fields)

Represents the collection of items in the data region of the PivotTable.

[*Example*:

```
<dataFields count="1">
  <dataField name="Sum of Sales Amount" fld="25" baseField="0" baseItem="0"/>
</dataFields>
```

end example]

Attributes	Description
count (Data Items Count)	Specifies the number of items in the data region of the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DataFields) is located in §A.2. *end note*]

18.10.1.24 dimension (OLAP Dimension)

Represents a PivotTable OLAP Dimension. A dimension is a field that organizes a single type of data into a hierarchy with levels of detail. [*Example*: An OLAP database could contain a Time dimension providing data for levels Year, Month, Week, and Day, allowing you to create reports that let you compare day-to-day sales results or view a summary of your sales for an entire year. *end example*]

Attributes	Description
caption (Dimension Display Name)	Specifies the display name of the dimension. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
measure (Measure)	Specifies a boolean value that indicates whether this is a measure dimension. A value of `1` or `true` indicates this dimension is a measure dimension. A value of `0` or `false` indicates this dimension is not a measure dimension. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
name (Dimension Name)	Specifies the name of the dimension. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
uniqueName (Dimension Unique Name)	Specifies the unique name of the dimension. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotDimension) is located in §A.2. *end note*]

18.10.1.25 dimensions (OLAP Dimensions)

Represents the collection of PivotTable OLAP dimensions.

[*Example*:

```
<dimensions count="22">
  <dimension name="Account" uniqueName="[Account]" caption="Account"/>
  <dimension name="Customer" uniqueName="[Customer]" caption="Customer"/>
  <dimension name="Date" uniqueName="[Date]" caption="Date"/>
  <dimension name="Delivery Date" uniqueName="[Delivery Date]"
    caption="Delivery Date"/>
```

```
    <dimension name="Department" uniqueName="[Department]" caption="Department"/>
    <dimension name="Destination Currency" uniqueName="[Destination Currency]"
      caption="Destination Currency"/>
    <dimension name="Employee" uniqueName="[Employee]" caption="Employee"/>
    <dimension name="Geography" uniqueName="[Geography]" caption="Geography"/>
    <dimension name="Internet Sales Order Details" uniqueName="[Internet Sales
      Order Details]" caption="Internet Sales Order Details"/>
    <dimension measure="1" name="Measures" uniqueName="[Measures]"
      caption="Measures"/>
    <dimension name="Organization" uniqueName="[Organization]"
      caption="Organization"/>
    <dimension name="Product" uniqueName="[Product]" caption="Product"/>
    <dimension name="Promotion" uniqueName="[Promotion]" caption="Promotion"/>
    <dimension name="Reseller" uniqueName="[Reseller]" caption="Reseller"/>
    <dimension name="Reseller Sales Order Details" uniqueName="[Reseller Sales
      Order Details]" caption="Reseller Sales Order Details"/>
    <dimension name="Sales Channel" uniqueName="[Sales Channel]" caption="Sales
      Channel"/>
    <dimension name="Sales Reason" uniqueName="[Sales Reason]" caption="Sales
      Reason"/>
    <dimension name="Sales Summary Order Details" uniqueName="[Sales Summary Order
      Details]" caption="Sales Summary Order Details"/>
    <dimension name="Sales Territory" uniqueName="[Sales Territory]"
      caption="Sales Territory"/>
    <dimension name="Scenario" uniqueName="[Scenario]" caption="Scenario"/>
    <dimension name="Ship Date" uniqueName="[Ship Date]" caption="Ship Date"/>
    <dimension name="Source Currency" uniqueName="[Source Currency]"
      caption="Source Currency"/>
  </dimensions>
```

end example]

Attributes	Description
count (OLAP Dimensions Count)	Specifies the number of OLAP dimensions in the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Dimensions) is located in §A.2. *end note*]

18.10.1.26 discretePr (Discrete Grouping Properties)

Represents the collection of discrete grouping properties for a field group.

[*Example*:

```
…
  <fieldGroup par="6" base="0">
    <rangePr groupBy="months" startDate="2002-01-01T00:00:00"
      endDate="2006-05-06T00:00:00"/>
    <groupItems count="14">
      <s v="&lt;1/1/2002"/>
      <s v="Jan"/>
      <s v="Feb"/>
      <s v="Mar"/>
      <s v="Apr"/>
      <s v="May"/>
      <s v="Jun"/>
      <s v="Jul"/>
      <s v="Aug"/>
      <s v="Sep"/>
      <s v="Oct"/>
      <s v="Nov"/>
      <s v="Dec"/>
      <s v="&gt;5/6/2006"/>
    </groupItems>
  </fieldGroup>
</cacheField>
<cacheField name="Name" numFmtId="0">
  <sharedItems count="4">
    <s v="Joe"/>
    <s v="John"/>
    <s v="Bob"/>
    <s v="Robert"/>
  </sharedItems>
  <fieldGroup par="4"/>
</cacheField>
<cacheField name="ProductID" numFmtId="0">
  <sharedItems containsSemiMixedTypes="0" containsString="0" containsNumber="1"
    containsInteger="1" minValue="1" maxValue="4" count="4">
    <n v="1"/>
    <n v="2"/>
    <n v="3"/>
    <n v="4"/>
  </sharedItems>
```



```
    <fieldGroup base="2">
      <rangePr startNum="1" endNum="4" groupInterval="2"/>
      <groupItems count="4">
        <s v="&lt;1"/>
        <s v="1-2"/>
        <s v="3-4"/>
        <s v="&gt;5"/>
      </groupItems>
    </fieldGroup>
  </cacheField>
```

end example]

Attributes	Description
count (Mapping Index Count)	Specifies the number of mapping indexes for this grouped field. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DiscretePr) is located in §A.2. *end note*]

18.10.1.27 e (Error Value)

Represents an error value. The use of this item indicates that an error value is present in the PivotTable source. The error is recorded in the value attribute.

Attributes	Description
b (Bold)	Specifies a boolean value that indicates whether the value contains bold formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of 1 or true indicates this value contains bold formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
bc (background Color)	Specifies the background color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
c (Item Caption)	Specifies the item/member caption The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§22.9.2.19).
cp (Member Property Count)	Specifies the number of member property values. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
f (Calculated Item)	Specifies a boolean value that indicates whether this is a calculated item value. A value of `1` or `true` indicates value is a calculated item value. A value of `0` or `false` indicates this value is not a calculated item value. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
fc (Foreground Color)	Specifies the foreground color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
i (Italic)	Specifies a boolean value that indicates whether the value contains italic formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of `1` or `true` indicates this value contains italic formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
in (Format Index)	Specifies the index to the OLAP serverformat element where the format string for this entry is stored. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
st (Strikethrough)	Specifies a boolean value that indicates whether the value contains strikethrough formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of `1` or `true` indicates this value contains strikethrough formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of `1` or `true` indicates this item is not used.

Attributes	Description
	A value of `0` or `false` indicates this item is used. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
un (Underline)	Specifies a boolean value that indicates whether the value contains underline formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of `1` or `true` indicates this value contains underline formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
v (Value)	Specifies the value of the item. This attribute depends on how the application records errors. [*Note*: While the error values are determined by the application, the following are some example error values that could be used: • #DIV/0! • #NAME? • #VALUE! • #NULL! • #NUM! • #REF! • #N/A • #GETTING_DATA *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Error) is located in §A.2. *end note*]

18.10.1.28 entries (Entries)

Represents the collection of OLAP sheet data entries.

Attributes	Description
count (Tuple Count)	Specifies the number of tuple entries. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PCDSDTCEntries) is located in §A.2. *end note*]

18.10.1.29 field (Field)

Represents a generic field that can appear either on the column or the row region of the PivotTable. There areas many <x> elements as there are item values in any particular column or row.

[*Example*:

```
<sh:field x="2"/>
```

end example]

Attributes	Description
x (Field Index)	Specifies the index to a pivotField item value. There are as many x elements as there are item values in any particular column. Note that these x elements sometimes are not explicitly written, but instead "inherited" from the previous column or i element, via the value of @r. The pivotField items don't list values explicitly, but instead reference a shared item value in the pivotCacheDefinition part. The first instance of x has no attribute value @v associated with it, so the default value for @v is assumed to be "0". The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Field) is located in §A.2. *end note*]

18.10.1.30 fieldGroup (Field Group Properties)

Represents the collection of properties for a field group.

[*Example*:

```
…
  <fieldGroup par="6" base="0">
    <rangePr groupBy="months" startDate="2002-01-01T00:00:00"
      endDate="2006-05-06T00:00:00"/>
    <groupItems count="14">
      <s v="&lt;1/1/2002"/>
      <s v="Jan"/>
      <s v="Feb"/>
      <s v="Mar"/>
      <s v="Apr"/>
      <s v="May"/>
      <s v="Jun"/>
      <s v="Jul"/>
      <s v="Aug"/>
      <s v="Sep"/>
```



```
        <s v="Oct"/>
        <s v="Nov"/>
        <s v="Dec"/>
        <s v="&gt;5/6/2006"/>
      </groupItems>
    </fieldGroup>
  </cacheField>
  <cacheField name="Name" numFmtId="0">
    <sharedItems count="4">
      <s v="Joe"/>
      <s v="John"/>
      <s v="Bob"/>
      <s v="Robert"/>
    </sharedItems>
    <fieldGroup par="4"/>
  </cacheField>
  <cacheField name="ProductID" numFmtId="0">
    <sharedItems containsSemiMixedTypes="0" containsString="0" containsNumber="1"
      containsInteger="1" minValue="1" maxValue="4" count="4">
      <n v="1"/>
      <n v="2"/>
      <n v="3"/>
      <n v="4"/>
    </sharedItems>
    <fieldGroup base="2">
      <rangePr startNum="1" endNum="4" groupInterval="2"/>
      <groupItems count="4">
        <s v="&lt;1"/>
        <s v="1-2"/>
        <s v="3-4"/>
        <s v="&gt;5"/>
      </groupItems>
    </fieldGroup>
  …
```

end example]

Attributes	Description
base (Field Base)	Specifies the base of this field, if any. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
par (Parent)	Specifies the parent of this field, if any.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_FieldGroup) is located in §A.2. *end note*]

18.10.1.31 fieldsUsage (Fields Usage)

Represents the fields in the cache that are being used by this hierarchy.

[*Example*:

```
<fieldsUsage count="6">
  <fieldUsage x="-1"/>
  <fieldUsage x="2"/>
  <fieldUsage x="3"/>
  <fieldUsage x="4"/>
  <fieldUsage x="5"/>
  <fieldUsage x="6"/>
</fieldsUsage>
```

end example]

Attributes	Description
count (Field Count)	Specifies the number of fields that are being used by this hierarchy. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_FieldsUsage) is located in §A.2. *end note*]

18.10.1.32 fieldUsage (PivotCache Field Id)

Represents a cache field used in this hierarchy.

[*Example*:

```
<fieldUsage x="-1"/>
```

end example]

Attributes	Description
x (Field Index)	Specifies the index of a field.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_FieldUsage) is located in §A.2. *end note*]

18.10.1.33 filter (PivotTable Advanced Filter)

Represents a PivotTable advanced filter.

[*Example*:

```
<sh:filter fld="3" type="count" id="1" iMeasureHier="187">
  <sh:autoFilter ref="A1">
    <sh:filterColumn colId="0">
      <sh:top10 val="5"/>
    </sh:filterColumn>
  </sh:autoFilter>
</sh:filter>
```

end example]

Attributes	Description
description (Pivot Filter Description)	Specifies the description of the pivot filter. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
evalOrder (Evaluation Order)	Specifies the evaluation order of the pivot filter. This attribute is zero-based. The possible values for this attribute are defined by the W3C XML Schema int datatype.
fld (Field Index)	Specifies the index of the field to which this pivot filter belongs. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
id (Pivot Filter Id)	Specifies the unique identifier of the pivot filter as assigned by the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
iMeasureFld (Measure Field Index)	Specifies the index of the measure field. This attribute is used only by filters in Relational pivots and specifies on which measure a value filter should apply. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

Attributes	Description
iMeasureHier (Measure Index)	Specifies the index of the measure cube field. This attribute is used only by filters in OLAP pivots and specifies on which measure a value filter should apply. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
mpFld (Member Property Field Id)	Specifies the index of the field representing the member property field on which this pivot filter is defined. This attribute is used only by label pivot filters. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (Pivot Filter Name)	Specifies the name of the pivot filter. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
stringValue1 (Label Pivot)	Specifies the string value "1" used by label pivot filters. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
stringValue2 (Label Pivot Filter String Value 2)	Specifies the string value "2" used by label pivot filters. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
type (Pivot Filter Type)	Specifies the type of the pivot filter. The possible values for this attribute are defined by the ST_PivotFilterType simple type (§18.18.59).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotFilter) is located in §A.2. *end note*]

18.10.1.34 filters (Filters)

Represents the collection of filters that apply to this PivotTable.

[*Example*:

```
<sh:filters count="1">
  <sh:filter fld="3" type="count" id="1" iMeasureHier="187">
    <sh:autoFilter ref="A1">
      <sh:filterColumn colId="0">
        <sh:top10 val="5"/>
      </sh:filterColumn>
    </sh:autoFilter>
  </sh:filter>
</sh:filters>
```

end example]

Attributes	Description
count (Pivot Filter Count)	Specifies the number of pivot filters in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotFilters) is located in §A.2. *end note*]

18.10.1.35 format (PivotTable Format)

Represents the format defined in the PivotTable.

Attributes	Description
action (Format Action)	Specifies the formatting behavior for the area indicated in the pivotArea element. The default value for this attribute is "formatting," which indicates that the specified cells have some formatting applied. The format is specified in the dxfId attribute. If the formatting is cleared from the cells, then the value of this attribute becomes "blank." The possible values for this attribute are defined by the ST_FormatAction simple type (§18.18.34).
dxfId (Format Id)	Specifies the identifier of the format the application is currently using for the PivotTable. Formatting information is written to the styles part. See the Styles section (§18.8) for more information on formats. Formatting information provided by cell table and by PivotTable need not agree. If the two formats differ, the cell-level formatting takes precedence. If you change the layout the PivotTable, the PivotTable formatting will then take precedence. The possible values for this attribute are defined by the ST_DxfId simple type (§18.18.25).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Format) is located in §A.2. *end note*]

18.10.1.36 formats (PivotTable Formats)

Represents the collection of formats applied to PivotTable.

Attributes	Description
count (Formats Count)	Specifies the number of formats in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Formats) is located in §A.2. *end note*]

18.10.1.37 group (OLAP Group)

Represents an OLAP level group.

[*Example*:

```
<group name="CategoryXl_Grp_1" uniqueName="[Product].[Product Categories].
  [Product Categories1].[GROUPMEMBER.[CategoryXl_Grp_1]].[Product]].
  [Product Categories]].[All Products]]]" caption="Group1"
  uniqueParent="[Product].[Product Categories].[All Products]" id="1">
  <groupMembers count="2">
    <groupMember
      uniqueName="[Product].[Product Categories].[Category].&amp;[4]"/>
    <groupMember
      uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
  </groupMembers>
</group>
```

end example]

Attributes	Description
caption (Group Caption)	Specifies the caption for this group. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
id (Group Id)	Specifies the unique number for this group within the level. The possible values for this attribute are defined by the W3C XML Schema int datatype.
name (Group	Specifies the name of this group.

Attributes	Description
Name)	The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
uniqueName (Unique Group Name)	Specifies the unique name of this group. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
uniqueParent (Parent Unique Name)	Specifies the unique name of the parent of this group. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_LevelGroup) is located in §A.2. *end note*]

18.10.1.38 groupItems (OLAP Group Items)

Represents the collection of items in a field group.

[*Example*:

```
…
  <fieldGroup par="6" base="0">
    <rangePr groupBy="months" startDate="2002-01-01T00:00:00"
      endDate="2006-05-06T00:00:00"/>
    <groupItems count="14">
      <s v="&lt;1/1/2002"/>
      <s v="Jan"/>
      <s v="Feb"/>
      <s v="Mar"/>
      <s v="Apr"/>
      <s v="May"/>
      <s v="Jun"/>
      <s v="Jul"/>
      <s v="Aug"/>
      <s v="Sep"/>
      <s v="Oct"/>
      <s v="Nov"/>
      <s v="Dec"/>
      <s v="&gt;5/6/2006"/>
    </groupItems>
  </fieldGroup>
</cacheField>
```



```
<cacheField name="Name" numFmtId="0">
  <sharedItems count="4">
    <s v="Joe"/>
    <s v="John"/>
    <s v="Bob"/>
    <s v="Robert"/>
  </sharedItems>
  <fieldGroup par="4"/>
</cacheField>
<cacheField name="ProductID" numFmtId="0">
  <sharedItems containsSemiMixedTypes="0" containsString="0" containsNumber="1"
    containsInteger="1" minValue="1" maxValue="4" count="4">
    <n v="1"/>
    <n v="2"/>
    <n v="3"/>
    <n v="4"/>
  </sharedItems>
  <fieldGroup base="2">
    <rangePr startNum="1" endNum="4" groupInterval="2"/>
    <groupItems count="4">
      <s v="&lt;1"/>
      <s v="1-2"/>
      <s v="3-4"/>
      <s v="&gt;5"/>
    </groupItems>
  </fieldGroup>
…
```

end example]

Attributes	Description
count (Items Created Count)	Specifies the number of items created for this grouped field. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_GroupItems) is located in §A.2. *end note*]

18.10.1.39 groupLevel (OLAP Grouping Levels)

Represents the collection of OLAP grouping levels.

[*Example*:


```
<groupLevel uniqueName="[Product].[Product Categories].[Category]"
  caption="Category">
  <groups count="1">
    <group name="CategoryXl_Grp_1" uniqueName="[Product].[Product
      Categories].[Product Categories1].
      [GROUPMEMBER.[CategoryXl_Grp_1]].[Product]].[Product Categories]].
      [All Products]]]" caption="Group1" uniqueParent="[Product].
      [Product Categories].[All Products]" id="1">
      <groupMembers count="2">
        <groupMember
          uniqueName="[Product].[Product Categories].[Category].&amp;[4]"/>
        <groupMember
          uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
      </groupMembers>
    </group>
  </groups>
</groupLevel>
```

end example]

Attributes	Description
caption (Grouping Level Display Name)	Specifies the display name for this grouping level. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
customRollUp (Custom Roll Up)	Specifies a boolean value that indicates whether this group level has a custom roll up. A value of `1` or `true` indicates this group level has a custom roll up. A value of `0` or `false` indicates this group level does not have a custom roll up. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
uniqueName (Unique Name)	Specifies the unique name for this grouping level. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
user (User-Defined Group Level)	Specifies a boolean value that indicates whether this is a user-defined group level. A value of `1` or `true` indicates this is a user-defined group. A value of `0` or `false` indicates this group is not user-defined. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_GroupLevel) is located in §A.2. *end note*]

18.10.1.40 groupLevels (OLAP Grouping Levels)

Represents the collection of OLAP grouping levels.

[*Example*:

```
<groupLevels count="5">
  <groupLevel uniqueName="[Product].[Product Categories].[(All)]"
    caption="(All)"/>
  <groupLevel uniqueName="[Product].[Product Categories].[Product Categories1]"
    caption="Product Categories1" user="1"/>
  <groupLevel uniqueName="[Product].[Product Categories].[Category]"
    caption="Category">
    <groups count="1">
      <group name="CategoryXl_Grp_1" uniqueName="[Product].[Product Categories].
        [Product Categories1].[GROUPMEMBER.[CategoryXl_Grp_1]].
        [Product]].[Product Categories]].[All Products]]]" caption="Group1"
        uniqueParent="[Product].[Product Categories].[All Products]" id="1">
        <groupMembers count="2">
          <groupMember
            uniqueName="[Product].[Product Categories].[Category].&amp;[4]"/>
          <groupMember
            uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
        </groupMembers>
      </group>
    </groups>
  </groupLevel>
  <groupLevel uniqueName="[Product].[Product Categories].[Subcategory]"
    caption="Subcategory"/>
  <groupLevel uniqueName="[Product].[Product Categories].[Product]"
    caption="Product"/>
</groupLevels>
```

end example]

Attributes	Description
count (Grouping Level Count)	Specifies the number of grouping levels. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_GroupLevels) is located in §A.2. *end note*]

18.10.1.41 groupMember (OLAP Group Member)

Represents an OLAP group member.

[*Example*:

```
<groupMember uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
```

end example]

Attributes	Description
group (Group)	Specifies a boolean value that indicates whether this member represents a group. A value of `1` or `true` indicates this member represents a group. A value of `0` or `false` indicates this member does not represent a group. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
uniqueName (Group Member Unique Name)	Specifies the unique name of this group member. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_GroupMember) is located in §A.2. *end note*]

18.10.1.42 groupMembers (OLAP Group Members)

Represents the collection of OLAP group members.

[*Example*:

```
<groupMembers count="2">
  <groupMember uniqueName="[Product].[Product Categories].[Category].&amp;[4]"/>
  <groupMember uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
</groupMembers>
```

end example]

Attributes	Description
count (Group Member Count)	Specifies the number of group members in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt

Attributes	Description
	datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_GroupMembers) is located in §A.2. *end note*]

18.10.1.43 groups (OLAP Level Groups)

Represents the collection of OLAP level groups.

[*Example*:

```
<groups count="1">
  <group name="CategoryXl_Grp_1" uniqueName="[Product].[Product Categories].
    [Product Categories1].[GROUPMEMBER.[CategoryXl_Grp_1]].[Product]].
    [Product Categories]].[All Products]]]" caption="Group1"
    uniqueParent="[Product].[Product Categories].[All Products]" id="1">
    <groupMembers count="2">
      <groupMember
        uniqueName="[Product].[Product Categories].[Category].&amp;[4]"/>
      <groupMember
        uniqueName="[Product].[Product Categories].[Category].&amp;[1]"/>
    </groupMembers>
  </group>
</groups>
```

end example]

Attributes	Description
count (Level Group Count)	Specifies the number of level groups in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Groups) is located in §A.2. *end note*]

18.10.1.44 i (Row Items)

Represents the collection of items in the row region of the PivotTable.

[*Example*: In this example the item values are found in cells B10:B13. For example "Bikes" is in B10, and corresponds to the first <i> element below.


```
<rowItems count="4">
  <i>
    <x/>
  </i>
  <i r="1">
    <x/>
  </i>
  <i r="1">
    <x v="1"/>
  </i>
  <i t="grand">
    <x/>
  </i>
</rowItems>
```

end example]

Attributes	Description
i (Data Field Index)	Specifies a zero-based index indicating the referenced data item it in a data field with multiple data items. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
r (Repeated Items Count)	Specifies the number of items to repeat from the previous row item. The first item has no @r explicitly written. Since a default of "0" is specified in the schema, for any item whose @r is missing, a default value of "0" is implied. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
t (Item Type)	Specifies the type of the item. Value of 'default' indicates a grand total as the last row item value The possible values for this attribute are defined by the ST_ItemType simple type (§18.18.43).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_I) is located in §A.2. *end note*]

18.10.1.45 item (PivotTable Field Item)

Represents a single item in PivotTable field.

[*Example*:

```
<sh:item x="66"/>
```


end example]

Attributes	Description
c (Child Items)	Specifies a boolean value that indicates whether the approximate number of child items for this item is greater than zero. A value of 1 or true indicates the approximate number of child items for this item is greater than zero. A value of 0 or false indicates the approximate number of child items for this item is zero. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
d (Expanded)	Specifies a boolean value that indicates whether this item has been expanded in the PivotTable view. A value of 1 or true indicates this item has been expanded. A value of 0 or false indicates this item is collapsed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
e (Drill Across Attributes)	Specifies a boolean value that indicates whether attribute hierarchies nested next to each other on a PivotTable row or column will offer drilling "across" each other or not. [*Example*: if the application offers drill across for attribute hierarchies and not for user hierarchies, this attribute would only be written when two attribute hierarchies are placed next to each other on an axis. *end example*] A value of 1 or true indicates there is a drill across attribute hierarchies positioned next to each other on a pivot axis. A value of 0 or false indicates there is not drill across attribute hierarchies. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
f (Calculated Member)	Specifies a boolean value that indicates whether this item is a calculated member. A value of 1 or true indicates this item is a calculated member. A value of 0 or false indicates this item is not calculated. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
h (Hidden)	Specifies a boolean value that indicates whether the item is hidden.

Attributes	Description
	A value of 1 or true indicates item is hidden. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
m (Missing)	Specifies a boolean value that indicate whether the item has a missing value. A value of 1 or true indicates the item value is missing. The application should still retain the item settings in case the item reappears during a later refresh. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
n (Item User Caption)	Specifies the user caption of the item. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
s (Character)	Specifies a boolean value that indicates whether the item has a character value. A value of 1 or true indicates the item has a string/character value. A value of 0 or false indicates item the item has a value of a different type. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sd (Hide Details)	Specifies a boolean value that indicates whether the details are hidden for this item. A value of 1 or true indicates item details are hidden. A value of 0 or false indicates item details are shown. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
t (Item Type)	Specifies the type of this item. A value of 'default' indicates the subtotal or total item. The possible values for this attribute are defined by the ST_ItemType simple type (§18.18.43).
x (Item Index)	Specifies the item index in pivotFields collection in the PivotCache. [*Example*: In the following example, "Product Category" and "Product Subcategory" are on the row axis of the PivotTable, in that order. `<rowFields count="2">` `        <field x="7"/>` `        <x="8"/>` `    </rowFieldsfield >`

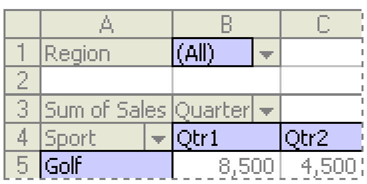
Attributes	Description
	end example] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Item) is located in §A.2. *end note*]

18.10.1.46 items (Field Items)

Represents the collection of items in a PivotTable field. The items in the collection are ordered by index. Items represent the unique entries from the field in the source data.

In the following image, the item Golf represents all rows of data in the source list for which the Sport field contains the entry Golf.



The order in which the items are listed is the order they would appear on a particular axis [*Example*: Row or column. *end example*]

[*Example*: In the following SpreadsheetML example, the first field is "Customer Name" and the first item referenced here is <item x="66"/>, which references the value "Adam L Flores" in the pivotCacheDefinition. Therefore, if you added "Customer Name" to the row axis, "Adam L Flores" would be the first row item listed.

```
<pivotFields count="28">
  <pivotField showAll="0" includeNewItemsInFilter="1">
    <items count="8">
      <item x="66"/>
      <item x="133"/>
      <item x="74"/>
      <item x="27"/>
      <item x="118"/>
      <item x="63"/>
      <item x="141"/>
      <item t="default"/>
    </items>
  </pivotField>
  <pivotField showAll="0" includeNewItemsInFilter="1"/>
```

```
    <pivotField axis="axisPage" showAll="0" includeNewItemsInFilter="1">
      <items count="2">
        <item x="0"/>
        <item t="default"/>
      </items>
    </pivotField>
  <pivotField showAll="0" includeNewItemsInFilter="1"/>
```

end example]

Attributes	Description
count (Field Count)	Specifies the number of fields in the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Items) is located in §A.2. *end note*]

18.10.1.47 kpi (OLAP KPI)

Represents the KPI defined on the OLAP server and stored in the PivotCache.

[*Example*:

```
  <kpi uniqueName="Growth in Customer Base" caption="Growth in Customer Base"
    displayFolder="Customer Perspective\Expand Customer Base"
    measureGroup="Internet Sales" value="[Measures].[Growth in Customer Base]"
    goal="[Measures].[Growth in Customer Base Goal]"
    status="[Measures].[Growth in Customer Base Status]"
    trend="[Measures].[Growth in Customer Base Trend]"/>
```

end example]

Attributes	Description
caption (KPI Display Name)	Specifies the display name of the KPI. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
displayFolder (KPI Display Folder)	Specifies the folder where this KPI is displayed in a list of fields for the PivotTable. This attribute depends on how the application exposes a list of fields in the user interface. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
goal (KPI Goal Unique Name)	Specifies the unique name of the KPI goal measure.

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
measureGroup (KPI Measure Group Name)	Specifies the name of the measure group to which this KPI belongs. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
parent (Parent KPI)	Specifies the name of the parent KPI for this KPI. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
status (KPI Status Unique Name)	Specifies the unique name of the KPI status measure. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
time (Time Member KPI Unique Name)	Specifies the unique name of the KPI current time member. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
trend (KPI Trend Unique Name)	Specifies the unique name of the KPI trend measure. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
uniqueName (KPI Unique Name)	Specifies the unique name of the KPI. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
value (KPI Value Unique Name)	Specifies the unique name of the KPI value measure. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
weight (KPI Weight Unique Name)	Specifies the unique name of the KPI weight measure. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PCDKPI) is located in §A.2. *end note*]

18.10.1.48 kpis (OLAP KPIs)

Represents the collection of Key Performance Indicators (KPIs) defined on the OLAP server and stored in the PivotCache.

[*Example*:

```
<kpis count="3">
  <kpi uniqueName="Growth in Customer Base" caption="Growth in Customer Base"
    displayFolder="Customer Perspective\Expand Customer Base"
    measureGroup="Internet Sales" value="[Measures].[Growth in Customer Base]"
    goal="[Measures].[Growth in Customer Base Goal]"
    status="[Measures].[Growth in Customer Base Status]"
    trend="[Measures].[Growth in Customer Base Trend]"/>
  <kpi uniqueName="Net Income" caption="Net Income"
    displayFolder="Financial Perspective\Maintain Overall Margins"
    measureGroup="Financial Reporting" value="[Measures].[Net Income Value]"
    goal="[Measures].[Net Income Goal]" status="[Measures].[Net Income Status]"
    trend="[Measures].[Net Income Trend]"/>
  <kpi uniqueName="Operating Profit" caption="Operating Profit"
    displayFolder="Financial Perspective\Maintain Overall Margins"
    measureGroup="Financial Reporting" parent="Net Income"
    value="[Measures].[Operating Profit Value]"
    goal="[Measures].[Operating Profit Goal]"
    status="[Measures].[Operating Profit Status]"
    trend="[Measures].[Operating Profit Trend]"/>
…
```

end example]

Attributes	Description
count (KPI Count)	Specifies the number of KPIs stored in the PivotCache. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PCDKPIs) is located in §A.2. *end note*]

18.10.1.49 location (PivotTable Location)

Represents location information for the PivotTable.

[*Example*:

```
<location ref="B6:G13" firstHeaderRow="1" firstDataRow="4" firstDataCol="1"
  rowPageCount="3" colPageCount="1"/>
```

end example]

Attributes	Description
colPageCount (Columns Per Page)	Specifies the number of columns per page for this PivotTable that the filter area will occupy. By default there is a single column of filter fields per page and the fields occupy as many rows as there are fields. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
firstDataCol (First Data Column)	Specifies the first column of the PivotTable data, relative to the top left cell in the `ref` value. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
firstDataRow (PivotTable Data First Row)	Specifies the first row of the PivotTable data, relative to the top left cell in the `ref` value. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
firstHeaderRow (First Header Row)	Specifies the first row of the PivotTable header, relative to the top left cell in the `ref` value. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ref (Reference)	Specifies the first row of the PivotTable. The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
rowPageCount (Rows Per Page Count)	Specifies the number of rows per page for this PivotTable that the filter area will occupy. By default there is a single column of filter fields per page and the fields occupy as many rows as there are fields. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Location) is located in §A.2. *end note*]

18.10.1.50 m (No Value)

Represents a value that was not specified.

[*Example*:

```
<sharedItems containsString="0" containsBlank="1" count="1">
  <m/>
</sharedItems>
```

end example]

Attributes	Description
b (Bold)	Specifies a boolean value that indicates whether the value contains bold formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of 1 or true indicates this value contains bold formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
bc (background Color)	Specifies the background color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
c (Caption)	Specifies the caption for this item. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
cp (Member Property Count)	Specifies the number of member property values for this item. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
f (Calculated Item)	Specifies a boolean value that indicates whether this is a calculated item value. A value of 1 or true indicates this item is a calculated value. A value of 0 or false indicates this item is not calculated. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
fc (Foreground Color)	Specifies the foreground color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
i (Italic)	Specifies a boolean value that indicates whether the value contains italic formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of 1 or true indicates this value contains underline formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
in (Format Index)	Specifies the index to the OLAP serverformat element where the format string for this

Attributes	Description
	entry is stored. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
st (Strikethrough)	Specifies a boolean value that indicates whether the value contains strikethrough formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of `1` or `true` indicates this value contains underline formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of `1` or `true` indicates this item is unused. A value of `0` or `false` indicates this item is used. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
un (Underline)	Specifies a boolean value that indicates whether the value contains underline formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of `1` or `true` indicates this value contains underline formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Missing) is located in §A.2. *end note*]

18.10.1.51 map (OLAP Measure Group)

Represents a PivotTable OLAP measure group - Dimension map.

[*Example*:

```
<map measureGroup="0" dimension="2"/>
```

end example]

Attributes	Description
dimension (Dimension Id)	Specifies the identifier for the dimension. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
measureGroup (Measure Group Id)	Specifies the identifier of the measure group. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MeasureDimensionMap) is located in §A.2. *end note*]

18.10.1.52 maps (OLAP Measure Group)

Represents the PivotTable OLAP measure group - Dimension maps.

[*Example*:

```
<maps count="3">
  <map measureGroup="0" dimension="2"/>
  <map measureGroup="1" dimension="19"/>
  <map measureGroup="2" dimension="8"/>
</maps>
```

end example]

Attributes	Description
count (Measure Group Count)	Specifies the number of measure groups, or dimension maps, in the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MeasureDimensionMaps) is located in §A.2. *end note*]

18.10.1.53 measureGroup (OLAP Measure Group)

Represents a PivotTable OLAP measure group.

[*Example*:

```
<measureGroup name="Sales Orders" caption="Sales Orders"/>
```

end example]

Attributes	Description
caption (Measure Group Display Name)	Specifies the display name of the measure group. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
name (Measure Group Name)	Specifies the name of the measure group. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_MeasureGroup) is located in §A.2. *end note*]

18.10.1.54 measureGroups (OLAP Measure Groups)

Represents the collection of PivotTable OLAP measure groups.

[*Example*:

```
<measureGroups count="11">
  <measureGroup name="Exchange Rates" caption="Exchange Rates"/>
  <measureGroup name="Financial Reporting" caption="Financial Reporting"/>
  <measureGroup name="Internet Customers" caption="Internet Customers"/>
  <measureGroup name="Internet Orders" caption="Internet Orders"/>
  <measureGroup name="Internet Sales" caption="Internet Sales"/>
  <measureGroup name="Reseller Orders" caption="Reseller Orders"/>
  <measureGroup name="Reseller Sales" caption="Reseller Sales"/>
  <measureGroup name="Sales Orders" caption="Sales Orders"/>
  <measureGroup name="Sales Reasons" caption="Sales Reasons"/>
  <measureGroup name="Sales Summary" caption="Sales Summary"/>
  <measureGroup name="Sales Targets" caption="Sales Targets"/>
</measureGroups>
```

end example]

Attributes	Description
count (Measure Group Count)	Specifies the number of measure groups in the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MeasureGroups) is located in §A.2. *end note*]

18.10.1.55 member (Member)

Represents an item that can be included or excluded.

Attributes	Description
name (Hidden Item Name)	Specifies the name of a hidden item. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Member) is located in §A.2. *end note*]

18.10.1.56 members (Members)

Represents the collection of items that can be included or excluded.

Attributes	Description
count (Item Count)	Specifies the number of items in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
level (Hierarchy Level)	Specifies the hierarchy level with which these items are associated. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Members) is located in §A.2. *end note*]

18.10.1.57 mp (OLAP Member Property)

Represents an OLAP member property.

[*Example*:

```
<sh:mp field="7"/>
```

end example]

Attributes	Description
field (Field Index)	Specifies the index of the field with which this member property is associated. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

Attributes	Description
level (Level Index)	Specifies the index of the level to which this member property applies. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (OLAP Member Property Unique Name)	Specifies the unique name of the OLAP member property. The following attributes depend on the name attribute: • nameLen • pLen • pPos These attributes consist of metadata about a member in an OLAP cube and are usually displayed in a tooltip or mechanism in the user interface. [*Example*: If the value for name equals "[Store].[Store Name].[Store Manager]": • nameLen will equal 20. This would refer to “[Store].[Store Name]” • pPos will equal 22. This would refer to starting character of “Store Manager” • pLen will equal 13. This would to length of “Store Manager” *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
nameLen (Name Length)	Specifies the length of the unique name portion of name. [*Example*: If the value for name equals "[Store].[Store Name].[Store Manager]", nameLen will equal 20. This would refer to “[Store].[Store Name]”. *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
pLen (Property Name Length)	Specifies the length of the property name portion of name. [*Example*: If the value for name equals "[Store].[Store Name].[Store Manager]", pLen will equal 13. This would to length of “Store Manager”. *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
pPos (Property Name Character Index)	Specifies the index of the character where the property name portion begins in name. [*Example*: If the value for name equals "[Store].[Store Name].[Store Manager]", pPos will equal 22. This would refer to starting character of “Store Manager”. *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
showAsCaption (Show As Caption)	Specifies a boolean value that indicates whether to show the property a member caption. A value of 1 or true indicates member property value is shown in as a caption.

Attributes	Description
	A value of 0 or false indicates member property value will not be shown. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showCell (Show Cell)	Specifies a boolean value that indicates whether to show the member property value in a PivotTable cell. A value of 1 or true indicates member property value is shown in a cell. A value of 0 or false indicates member property value will not be shown. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showTip (Show Tooltip)	Specifies a boolean value that indicates whether to show the member property value in a tooltip on the appropriate PivotTable view cells. A value of 1 or true indicates member property value is shown in a tooltip. A value of 0 or false indicates member property value will not be shown. This attribute depends on whether the application employs tooltips or similar mechanism in the user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MemberProperty) is located in §A.2. *end note*]

18.10.1.58 mpMap (Member Properties Map)

Represents a mapping to cached member properties.

[*Example*:

```
<mpMap v="7"/>
```

end example]

Attributes	Description
v (Shared Items Index)	Specifies the index into the shared items table in the PivotCache that identifies this item. The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_X) is located in §A.2. *end note*]

18.10.1.59 mps (OLAP Member Properties)

Represents the collection of OLAP member property. Member properties contain additional information that is available about the items in an OLAP dimension field. [*Example*: If a Geography dimension has property fields Population and Average Income available, you could create a PivotTable report that displays the sales figures for cities where your products are selling well. By displaying and analyzing the population and income figures for these cities, you could target cities with similar demographics for your marketing campaign. *end example*]

[*Example*:

```
<sh:mps count="3">
  <sh:mp field="7"/>
  <sh:mp field="8"/>
  <sh:mp field="9"/>
</sh:mps>
```

end example]

Attributes	Description
count (OLAP Member Properties Count)	Specifies the number of OLAP member properties in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_MemberProperties) is located in §A.2. *end note*]

18.10.1.60 n (Numeric)

Represents a numeric value in the PivotTable.

[*Example*:

```
<sharedItems containsSemiMixedTypes="0" containsString="0" containsNumber="1"
  containsInteger="1" minValue="3" maxValue="3" count="1">
  <n v="3"/>
</sharedItems>
```

end example]

Attributes	Description
b (Bold)	Specifies a boolean value that indicates whether this value contains bold formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of 1 or true indicates this value contains italic formatting on the server.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
bc (Background Color)	Specifies the background color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
c (Caption)	Specifies the caption for this item. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
cp (Member Property Count)	Specifies the number of member property values for this item. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
f (Calculated Item)	Specifies a boolean value that indicates whether this is a calculated item value. A value of `1` or `true` indicates this item is a calculated value. A value of `0` or `false` indicates this item is not calculated. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
fc (Foreground Color)	Specifies the foreground color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
i (Italic)	Specifies a boolean value that indicates whether the value contains italic formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of `1` or `true` indicates this value contains italic formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
in (Format Index)	Specifies the index to the OLAP serverformat element where the format string for this entry is stored. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
st (Strikethrough)	Specifies a boolean value that indicates whether the value contains strikethrough

Attributes	Description
	formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of `1` or `true` indicates this value contains strikethrough formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of `1` or `true` indicates this item is not used. A value of `0` or `false` indicates this item is used. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
un (Underline)	Specifies a boolean value that indicates whether the value contains underline formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of `1` or `true` indicates this value contains underline formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
v (Value)	Specified the value of this item. The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Number) is located in §A.2. *end note*]

18.10.1.61 page (Page Items)

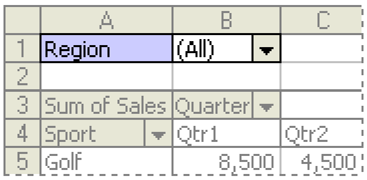
Represents the collection of page item values for a page field.

Attributes	Description
count (Page Item String Count)	Specifies the number of page item strings in the collectoin. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PCDSCPage) is located in §A.2. *end note*]

18.10.1.62 pageField (Page Field)

Represents a field on the page or report filter of the PivotTable.



In the image above, the blue field is a page or report filter field. Page/filter fields allow you to filter the entire PivotTable to display data for a single item or all the items.

[*Example*:

```
<sh:pageField fld="43" hier="103"
  name="[Product].[Product Categories].[All Products]" cap="All Products"/>
```

end example]

Attributes	Description
cap (Hierarchy Display Name)	Specifies the display name of the hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
fld (Field)	Specifies the index of the field that appears on the page or filter report area of the PivotTable. The possible values for this attribute are defined by the W3C XML Schema int datatype.
hier (OLAP Hierarchy Index)	Specifies the index of the OLAP hierarchy to which this item belongs. The possible values for this attribute are defined by the W3C XML Schema int datatype.
item (Item Index)	Specifies the index of the item in the PivotCache. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (Hierarchy Unique Name)	Specifies the unique name of the hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageField) is located in §A.2. *end note*]

18.10.1.63 pageFields (Page Field Items)

Represents the collection of items in the page or report filter region of the PivotTable.

[*Example*:

```
<sh:pageFields count="2">
  <sh:pageField fld="43" hier="103"
    name="[Product].[Product Categories].[All Products]" cap="All Products"/>
  <sh:pageField fld="66" hier="126"
    name="[Promotion].[Promotions].[All Promotions]" cap="All Promotions"/>
</sh:pageFields>
```

end example]

Attributes	Description
count (Page Item Count)	Specifies the number of items in the page region of the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageFields) is located in §A.2. *end note*]

18.10.1.64 pageItem (Page Item)

Represents an item value for a PivotTable page.

Attributes	Description
name (Page Item Name)	Specifies the name of this page item. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PageItem) is located in §A.2. *end note*]

18.10.1.65 pages (Page Item Values)

Represents the collection of page item values for each page field.

Attributes	Description
count (Page Item String Count)	Specifies the number of page item strings in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt

Attributes	Description
	datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Pages) is located in §A.2. *end note*]

18.10.1.66 pivotAreas (Pivot Areas)

Represents the collection of pivot areas that comprise the PivotTable location.

[*Example*:

```
<sh:pivotAreas count="1">
  <sh:pivotArea field="2" dataOnly="0" outline="0"/>
</sh:pivotAreas>
```

end example]

Attributes	Description
count (Pivot Area Count)	Specifies the number of PivotAreas for the PivotTable location. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotAreas) is located in §A.2. *end note*]

18.10.1.67 pivotCacheDefinition (PivotCache Definition)

Represents the pivotCacheDefinition part. This part defines each field in the source data, including the name, the string resources of the instance data (for shared items), and information about the type of data that appears in the field.

[*Example*:

```
<pivotCacheDefinition xmlns="…" xmlns:r="…" r:id="rId1" refreshedBy="AnonUser"
  refreshedDateIso="2006-05-22T10:07:16Z" createdVersion="3"
refreshedVersion="3"
  minRefreshableVersion="3" recordCount="182">
  <cacheSource type="worksheet">
    <worksheetSource name="Table1"/>
  </cacheSource>
```

```
    <cacheFields count="28">
      <cacheField name="Customer Name" numFmtId="0">
      <cacheField name="Postal Code" numFmtId="0">
        <sharedItems/>
      </cacheField>
      <cacheField name="Product Category" numFmtId="0">
        <sharedItems count="1">
          <s v="Bikes"/>
        </sharedItems>
      </cacheField>
      <cacheField name="Year" numFmtId="0">
        <sharedItems count="1">
          <s v="2001"/>
        </sharedItems>
      </cacheField>
      <cacheField name="Quarter" numFmtId="0">
        <sharedItems containsSemiMixedTypes="0" containsString="0"
          containsNumber="1" containsInteger="1" minValue="3" maxValue="3"
          count="1">
          <n v="3"/>
        </sharedItems>
      </cacheField>
    </cacheFields>
  </pivotCacheDefinition>
```

end example]

Attributes	Description
backgroundQuery (Background Query)	Specifies a boolean value that indicates whether the application should query and retrieve records asynchronously from the cache. A value of `1` or `true` indicates the application will retrieve records asynchronously from the cache. A value of `0` or `false` indicates the application will retrieve records synchronously. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
createdVersion (PivotCache Created Version)	Specifies the version of the application that created the cache. This attribute is application-dependent. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
enableRefresh	Specifies a boolean value that indicates whether the end-user can refresh the cache. This

Attributes	Description
(Enable PivotCache Refresh)	attribute depends on whether the application exposes a method for allowing end-users control over refreshing the cache via the user interface. A value of `1` or `true` indicates the end-user can refresh the cache. A value of `0` or `false` indicates the end-user cannot refresh the cache. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
id (Relationship Identifier) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the unique identifier that corresponds to the related `pivotCacheRecords` part. See (§18.10.1.68) for more information. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
invalid (Invalid Cache)	Specifies a boolean value that indicates whether the cache needs to be refreshed. A value of `1` or `true` indicates the cache needs to be refreshed. A value of `0` or `false` indicates the cache does not need to be refreshed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
minRefreshableVersion (Minimum Version Required for Refresh)	Specifies the earliest version of the application that is required to refresh the cache. This attribute is application-dependent. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
missingItemsLimit (Missing Items Limit)	Specifies the number of unused items to allow before discarding unused items. This attribute is application-dependent. The application shall specify a threshold for unused items. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
optimizeMemory (Optimize Cache for Memory)	Specifies a boolean value that indicates whether the application will apply optimizations to the cache to reduce memory usage. This attribute is application-dependent. This application shall define its own cache optimization methods. The application shall also decide whether to expose cache optimization status via the user interface or an object model. A value of `1` or `true` indicates the application will apply optimizations to the cache.

Attributes	Description
	A value of 0 or false indicates the application will not apply optimizations to the cache. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
recordCount (PivotCache Record Count)	Specifies the number of records in the cache. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
refreshedBy (Last Refreshed By)	Specifies the name of the end-user who last refreshed the cache. This attribute is application-dependent and is specified by applications that track and store the identity of the current user. This attribute also depends on whether the application exposes mechanisms via the user interface whereby the end-user can refresh the cache. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
refreshedDateIso (PivotCache Last Refreshed Date ISO)	Specifies the date when the cache was last refreshed. This attribute depends on whether the application exposes mechanisms via the user interface whereby the end-user can refresh the cache. The possible values for this attribute are defined by the W3C XML Schema dateTime datatype.
refreshedVersion (PivotCache Last Refreshed Version)	Specifies the version of the application that last refreshed the cache. This attribute depends on whether the application exposes mechanisms via the user interface whereby the end-user can refresh the cache. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
refreshOnLoad (Refresh On Load)	Specifies a boolean value that indicates whether the application will refresh the cache when the workbook has been opened. A value of 1 or true indicates that application will refresh the cache when the workbook is loaded. A value of 0 or false indicates the application will not automatically refresh cached data. The end user shall trigger refresh of the cache manually via the application user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
saveData (Save Pivot Records)	Specifies a boolean value that indicates whether the pivot records are saved with the cache. A value of 1 or true indicates pivot records are saved in the cache. A value of 0 or false indicates are not saved in the cache.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
supportAdvancedDrill (Supports Attribute Drilldown)	Specifies whether the cache's data source supports attribute drilldown. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
supportSubquery (Supports Subqueries)	Specifies whether the cache's data source supports subqueries. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
tupleCache (Stores Cache for OLAP Functions)	Specifies a boolean value that indicates whether the PivotCache is used store information for OLAP sheet data functions. A value of `1` or `true` indicates information about OLAP sheet data functions are stored in the cache. A value of `0` or `false` indicates the PivotCache does not contain information about OLAP sheet data functions. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
upgradeOnRefresh (Upgrade PivotCache on Refresh)	Specifies a boolean value that indicates whether the cache is scheduled for version upgrade. This attribute depends on whether the application exposes mechanisms via the user interface whereby the cache might be upgraded. A value of `1` or `true` indicates the cache is scheduled for upgrade. A value of `0` or `false` indicates the cache is not scheduled for upgrade. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotCacheDefinition) is located in §A.2. *end note*]

18.10.1.68 pivotCacheRecords (PivotCache Records)

Represents the collection of records in the PivotCache. This part stores the underlying source data that the PivotTable aggregates.

[*Example*:


```
<pivotCacheRecords xmlns="…" xmlns:r="…" count="2">
  <r>
    <x v="0"/>
    <s v="Pacific"/>
    <x v="0"/>
    <s v="Australia"/>
    <x v="0"/>
    <x v="0"/>
    <s v="3550"/>
    <x v="0"/>
    <x v="0"/>
    <s v="Road-150 Red, 62"/>
    <s v="This bike is ridden by race winners. Developed with the Adventure
      Works Cycles professional race team, it has a extremely light
      heat-treated aluminum frame, and steering that allows precision
      control."/>
    <s v="No Discount"/>
…
    <n v="89.456800000000001"/>
  </r>
…
```

end example]

Attributes	Description
count (PivotCache Records Count)	Specifies the number of records in the cache. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotCacheRecords) is located in §A.2. *end note*]

18.10.1.69 pivotField (PivotTable Field)

Represents a single field in the PivotTable. This element contains information about the field, including the collection of items in the field.

[*Example*:

```
<pivotField axis="axisRow" allDrilled="1" showAll="0" measureFilter="1"
  sortType="descending">
```



```
    <items count="8">
      <item s="1" c="1" x="0"/>
      <item s="1" c="1" x="1"/>
      <item c="1" x="2"/>
      <item c="1" x="3"/>
      <item c="1" x="4"/>
      <item c="1" x="5"/>
      <item c="1" x="6"/>
      <item t="default"/>
    </items>
    <autoSortScope>
      <pivotArea dataOnly="0" outline="0" fieldPosition="0">
        <references count="2">
          <reference field="4294967294" count="1" selected="0">
            <x v="0"/>
          </reference>
          <reference field="25" count="1" selected="0">
            <x v="0"/>
          </reference>
        </references>
      </pivotArea>
    </autoSortScope>
  </pivotField>
```

end example]

Attributes	Description
allDrilled (All Items Expanded)	Specifies a boolean value that indicates whether all items in the field are expanded. Applies only to OLAP PivotTables. A value of `1` or `true` indicates all items in the field are expanded. A value of `0` or `false` indicates all items are not expanded. However some items might be expanded. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoShow (Auto Show)	Specifies a boolean value that indicates whether an "AutoShow" filter is applied to this field. This attribute depends on the implementation of filtering in the application. A value of `1` or `true` indicates an "AutoShow" filter is applied to the field. A value of `0` or `false` indicates an "AutoShow" filter is not applied. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
	datatype.
avgSubtotal (Average)	Specifies a boolean value that indicates whether to apply the 'Average' aggregation function in the subtotal of this field. A value of `1` or `true` indicates the subtotal for this field is 'Average.' A value of `0` or `false` indicates a different aggregation function is applied to the subtotal for this field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
axis (Axis)	Specifies the region of the PivotTable that this field is displayed. The possible values for this attribute are defined by the ST_Axis simple type (§18.18.1).
compact (Compact)	Specifies a boolean value that indicates whether the application will display fields compactly in the sheet on which this PivotTable resides. A value of `1` or `true` indicates the next field should be displayed in the same column of the sheet. A value of `0` or `false` indicates each pivot field will display in its own column in the sheet. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
countASubtotal (CountA)	Specifies a boolean value that indicates whether to apply the 'countA' aggregation function in the subtotal of this field. A value of `1` or `true` indicates the subtotal for this field is 'countA.' A value of `0` or `false` indicates a different aggregation function is applied to the subtotal for this field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
countSubtotal (Count)	Specifies a boolean value that indicates whether to apply the 'count' aggregation function in the subtotal of this field. A value of `1` or `true` indicates the subtotal for this field is 'count.' A value of `0` or `false` indicates a different aggregation vfunction is applied to the subtotal for this field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
dataField (Data Field)	Specifies a boolean value that indicates whether this field appears in the data region of the PivotTable. A value of `1` or `true` indicates this field appears in the data region of the PivotTable. A value of `0` or `false` indicates this field appears in another region of the PivotTable. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dataSourceSort (Data Source Sort)	Specifies a boolean value that indicates whether sort is applied to this field in the data source. A value of `1` or `true` indicates this field is sorted in the data source. A value of `0` or `false` indicates this field is not sorted in the data source. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
defaultAttributeDrillState (Drill State)	Specifies a boolean value that indicates the drill state of the attribute hierarchy in an OLAP-based PivotTable. A value of `1` or `true` indicates the attribute hierarchy is expanded. A value of `0` or `false` indicates the attribute hierarchy is collapsed. This attribute is designed to allow the application to issue more optimized queries when all items of each field have the same drill state. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
defaultSubtotal (Show Default Subtotal)	Specifies a boolean value that indicates whether the default subtotal aggregation function is displayed for this field. A value of `1` or `true` indicates the default subtotal aggregation function is displayed for this field. A value of `0` or `false` indicates the default aggregation function is not displayed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dragOff (Drag Off)	Specifies a boolean value that indicates whether the field can be removed from the PivotTable. A value of `1` or `true` indicates the field can be removed from the PivotTable.

Attributes	Description
	A value of 0 or false indicates the field cannot be removed from the PivotTable. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dragToCol (Drag To Column)	Specifies a boolean value that indicates whether the field can be dragged to the column axis. A value of 1 or true indicates the field can be dragged to the column axis. A value of 0 or false indicates the field cannot be dragged to the column axis. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dragToData (Field Can Drag to Data)	Specifies a boolean value that indicates whether the field can be dragged to the data region. A value of 1 or true indicates the field can be dragged to the data region. A value of 0 or false indicates the field cannot be dragged to the data region. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dragToPage (Drag Field to Page)	Specifies a boolean value that indicates whether the field can be dragged to the page region. A value of 1 or true indicates the field can be dragged to the page region. A value of 0 or false indicates the field cannot be dragged to the page region. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dragToRow (Drag To Row)	Specifies a boolean value that indicates whether the field can be dragged to the row axis. A value of 1 or true indicates the field can be dragged to the row axis. A value of 0 or false indicates the field cannot be dragged to the row axis. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
hiddenLevel (Hidden Level)	Specifies a boolean value that indicates whether there is a hidden level in the PivotTable. This attribute applies to OLAP-based PivotTables only. A value of 1 or true indicates the OLAP PivotTable contains a hidden level.

Attributes	Description
	A value of 0 or false indicates the OLAP PivotTable does not contain any hidden levels. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
hideNewItems (Hide New Items)	Specifies a boolean value that indicates whether new items that appear after a refresh should be hidden by default. A value of 1 or true indicates that items that appear after a refresh should be hidden by default. A value of 0 or false indicates that items that appear after a refresh should be shown by default. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
includeNewItemsInFilter (Inclusive Manual Filter)	Specifies a boolean value that indicates whether manual filter is in inclusive mode. A value of 1 or true indicates the manual filter is inclusive. A value of 0 or false indicates the manual filter is not inclusive. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
insertBlankRow (Insert Blank Row)	Specifies a boolean value that indicates whether to insert a blank row after each item. A value of 1 or true indicates that a blank row is inserted after each item. A value of 0 or false indicates no additional rows are inserted after each item. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
insertPageBreak (Insert Item Page Break)	Specifies a boolean value that indicates whether to insert a page break after each item. A value of 1 or true indicates that a page break is inserted after each item. A value of 0 or false indicates no page breaks are inserted after items. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
itemPageCount (Items Per Page Count)	Specifies the number of items showed per page in the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

Attributes	Description
maxSubtotal (Max Subtotal)	Specifies a boolean value that indicates whether to apply the 'max' aggregation function in the subtotal of this field. A value of 1 or true indicates that the 'max' aggregation function is applied in the subtotal for this field. A value of 0 or false indicates another aggregation function is applied in the subtotal for this field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
measureFilter (Measure Filter)	Specifies a boolean value that indicates whether field has a measure based filter. A value of 1 or true indicates the field has a measure-based filter. A value of 0 or false indicates does not have a measure-based filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
minSubtotal (Min Subtotal)	Specifies a boolean value that indicates whether to apply the 'min' aggregation function in the subtotal of this field. A value of 1 or true indicates that the 'min' aggregation function is applied in the subtotal for this field. A value of 0 or false indicates another aggregation function is applied in the subtotal for this field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
multipleItemSelectionAllowed (Multiple Field Filters)	Specifies a boolean value that indicates whether the field can have multiple items selected in the page field. A value of 1 or true indicates the PivotTable can have multiple items selected in the page field. A value of 0 or false indicates the PivotTable cannot have multiple items selected in the page field. This attribute depends on the application support for selecting multiple items in page fields. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
name (Field Name)	Specifies the name of the field. The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§22.9.2.19).
nonAutoSortDefault (Auto Sort)	Specifies a boolean value that indicates whether sort operation that is applied to field should be AutoSort operation or simple data sort operation. A value of `1` or `true` indicates that an AutoSort operation is applied to the field. A value of `0` or `false` indicates a simple data sort operation is applied to the field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
numFmtId (Number Format Id)	Specifies the identifier of the number format to apply to this field. Number formats are written to the styles part. See the Styles section (§18.8) for more information on number formats. Formatting information provided by cell table and by PivotTable need not agree. If the two formats differ, the cell-level formatting takes precedence. If you change the layout the PivotTable, the PivotTable formatting will then take precedence. The possible values for this attribute are defined by the ST_NumFmtId simple type (§18.18.47).
outline (Outline Items)	Specifies a boolean value that indicates whether the items in this field should be shown in Outline form. A value of `1` or `true` indicates the items in this field is shown in Outline form. A value of `0` or `false` indicates the items in this field will not be shown in Outline form. This attribute depends on the application support for displaying items in Outline form. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
productSubtotal (Product Subtotal)	Specifies a boolean value that indicates whether to apply 'product' aggregation function in the subtotal of this field. A value of `1` or `true` indicates that the 'product' aggregation function is applied in the subtotal for this field. A value of `0` or `false` indicates another aggregation function is applied in the subtotal for this field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rankBy (Auto Show Rank By)	Specifies the index of the data field by which AutoShow will rank. The possible values for this attribute are defined by the W3C XML Schema unsignedInt

Attributes	Description
	datatype.
serverField (Server-based Page Field)	Specifies a boolean value that indicates whether this is a server-based page field. A value of `1` or `true` indicates this is a server-based page field. A value of `0` or `false` indicates this is a local page field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showAll (Show All Items)	Specifies a boolean value that indicates whether to show all items for this field. A value of `1` or `true` indicates that all items be shown. A value of `0` or `false` indicates items be shown according to user specified criteria. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showDropDowns (Show PivotField Header Drop Downs)	Specifies a boolean value that indicates whether to hide drop down buttons on PivotField headers. This attribute depends on the application implementation for filtering in the user interface. A value of `1` or `true` indicates the application will display some mechanism for selecting and applying filters – [*Example*: A dropdown menu *end example*] – in the user interface. A value of `0` or `false` indicates for mechanism for applying a filter is displayed in the user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showPropAsCaption (Show As Caption)	Specifies a boolean value that indicates whether to show the property as a member caption. A value of `1` or `true` indicates the property is shown as a member caption. A value of `0` or `false` indicates the property will not be shown as a member caption. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showPropCell (Show Member Property in Cell)	Specifies a boolean value that indicates whether to show the member property value in a PivotTable cell. A value of `1` or `true` indicates the property value is shown in a PivotTable cell. A value of `0` or `false` indicates the property value will not be shown in a PivotTable cell.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showPropTip (Show Member Property ToolTip)	Specifies a boolean value that indicates whether to show the member property value in a tooltip on the appropriate PivotTable cells. A value of `1` or `true` indicates the property value is shown in a tooltip in the user interface. A value of `0` or `false` indicates the property will not be shown in a tooltip. This attribute depends on whether the application employs tooltips or similar mechanism in the user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sortType (Auto Sort Type)	Specifies the type of sort that is applied to this field. The possible values for this attribute are defined by the ST_FieldSortType simple type (§18.18.28).
stdDevPSubtotal (StdDevP Subtotal)	Specifies a boolean value that indicates whether to apply the 'stdDevP' aggregation function in the subtotal of this field. A value of `1` or `true` indicates that the 'stdDevP' aggregation function is applied in the subtotal for this field. A value of `0` or `false` indicates another aggregation function is applied in the subtotal for this field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
stdDevSubtotal (StdDev Subtotal)	Specifies a boolean value that indicates whether to use 'stdDev' in the subtotal of this field. A value of `1` or `true` indicates that the 'stdDev' aggregation function is applied in the subtotal for this field. A value of `0` or `false` indicates another aggregation function is applied in the subtotal for this field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
subtotalCaption (Custom Subtotal Caption)	Specifies the custom text that is displayed for the subtotals label. The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§22.9.2.19).
subtotalTop (Subtotals At Top)	Specifies a boolean value that indicates whether to display subtotals at the top of the group. Applies only when `Outline` its true. A value of `1` or `true` indicates a subtotal is display at the top of the group. A value of `0` or `false` indicates subtotal will not be displayed at the top of the group. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sumSubtotal (Sum Subtotal)	Specifies a boolean value that indicates whether apply the 'sum' aggregation function in the subtotal of this field. A value of `1` or `true` indicates the 'sum' aggregation function is applied in the subtotal of this field. A value of `0` or `false` indicates another aggregation function is applied in the subtotal of this field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
topAutoShow (Top Auto Show)	Specifies a boolean value that indicates whether an AutoShow filter applied to this field is set to show the top ranked values. A value of `1` or `true` indicates whether an AutoShow filter will show top values for this field. A value of `0` or `false` indicates bottom ranked values are shown. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
uniqueMemberProperty (Unique Member Property)	Specifies the unique name of the member property to be used as a caption for the field and field items. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
varPSubtotal (VarP Subtotal)	Specifies a boolean value that indicates whether to apply the 'varP' aggregation function in the subtotal of this field. A value of `1` or `true` indicates the 'varP' aggregation function is applied in the subtotal of this field. A value of `0` or `false` indicates another aggregation function is applied in the subtotal of this field.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
varSubtotal (Variance Subtotal)	Specifies a boolean value that indicates whether to apply the 'variance' aggregation function in the subtotal of this field. A value of `1` or `true` indicates the 'variance' aggregation function is applied in the subtotal of this field. A value of `0` or `false` indicates another aggregation function is applied in the subtotal of this field. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotField) is located in §A.2. *end note*]

18.10.1.70 pivotFields (PivotTable Fields)

Represents the collection of fields that appear on the PivotTable.

[*Example*:

```
<pivotFields count="28">
  <pivotField showAll="0" includeNewItemsInFilter="1">
    <items count="8">
      <item x="66"/>
      <item x="133"/>
      <item x="74"/>
      <item x="27"/>
      <item x="118"/>
      <item x="63"/>
      <item x="141"/>
      <item t="default"/>
    </items>
  </pivotField>
  <pivotField showAll="0" includeNewItemsInFilter="1"/>
```



```
    <pivotField axis="axisPage" showAll="0" includeNewItemsInFilter="1">
      <items count="2">
        <item x="0"/>
        <item t="default"/>
      </items>
    </pivotField>
  <pivotField showAll="0" includeNewItemsInFilter="1"/>
```

end example]

Attributes	Description
count (Field Count)	Specifies the number of fields in the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotFields) is located in §A.2. *end note*]

18.10.1.71 pivotHierarchies (PivotTable OLAP Hierarchies)

Represents the collection of OLAP hierarchies associated with the PivotTable.

[*Example*:

```
  <sh:pivotHierarchies count="3">
    <sh:pivotHierarchy dragToRow="0" dragToCol="0" dragToPage="0" dragToData="1"/>
    <sh:pivotHierarchy dragToRow="0" dragToCol="0" dragToPage="0" dragToData="1"/>
    <sh:pivotHierarchy dragToRow="0" dragToCol="0" dragToPage="0" dragToData="1"/>
    <sh:pivotHierarchy dragToRow="0" dragToCol="0" dragToPage="0" dragToData="1"/>
  </sh:pivotHierarchies>
```

end example]

Attributes	Description
count (OLAP Hierarchy Count)	Specifies the number of OLAP hierarchies in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotHierarchies) is located in §A.2. *end note*]

18.10.1.72 pivotHierarchy (OLAP Hierarchy)

Represents a OLAP hierarchy associated with the PivotTable. A hierarchy is a hierarchical representation of related OLAP dimensions. Hierarchies are defined on the OLAP server and cannot be changed in the PivotTable. [*Example*: Hierarchy "A" might be defined as follows:

Level 1 Country/Region

Level 2 State\Provence

Level 3 City

end example]

[*Example*:

```
<sh:pivotHierarchy dragToRow="0" dragToCol="0" dragToPage="0" dragToData="1"/>
```

end example]

Attributes	Description
caption (Hierarchy Caption)	Specifies the user defined caption of the hierarchy. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
dragOff (Drag Off)	Specifies a boolean value that indicates whether the user is allowed to remove this hierarchy from the PivotTable. A value of `1` or `true` indicates the user can remove this hierarchy from the PivotTable. A value of `0` or `false` indicates the user cannot remove the hierarchy from the PivotTable. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dragToCol (Drag To Column)	Specifies a boolean value that indicates whether the user is allowed to put this hierarchy into the column area of the PivotTable. A value of `1` or `true` indicates the user can put this hierarchy into the column area of the PivotTable. A value of `0` or `false` indicates the user cannot remove this hierarchy. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dragToData (Drag To Data)	Specifies a boolean value that indicates whether the user is allowed to put this hierarchy into the data area of the view.

Attributes	Description
	A value of 1 or true indicates A value of 0 or false indicates The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dragToPage (Drag to Page)	Specifies a boolean value that indicates whether the user is allowed to put this hierarchy into the page area of the PivotTable. A value of 1 or true indicates the user can put this hierarchy into the page area of the PivotTable. A value of 0 or false indicates cannot put this hierarchy into the page area. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dragToRow (Drag To Row)	Specifies a boolean value that indicates whether the user is allowed to put this hierarchy into the row area of the PivotTable. A value of 1 or true indicates the user can put this hierarchy into the row area of the PivotTable. A value of 0 or false indicates cannot put this hierarchy into the row area. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
includeNewItemsInFilter (Inclusive Manual Filter)	Specifies a boolean value that indicates whether the application will show only the items the user has selected. A value of 1 or true indicates the application will show only items the user has selected; all other items are hidden. A value of 0 or false indicates the application will show all items. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
multipleItemSelectionAllowed (Multiple Field Filters)	Specifies a boolean value that indicates whether the user can select multiple members when the hierarchy is in the page field area of the view. A value of 1 or true indicates the user can select multiple members. A value of 0 or false indicates the user cannot select multiple members. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
	datatype.
outline (Outline New Levels)	Specifies a boolean value that indicates whether new levels added to the PivotTable are shown in Outline mode. A value of `1` or `true` indicates new levels are shown in Outline mode. A value of `0` or `false` indicates new items are not shown in Outline mode. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showInFieldList (Show In Field List)	Specifies a boolean value that indicates whether this hierarchy is omitted from the field list. This attribute depends on how the application exposes a list of fields for PivotTables in the user interface. A value of `1` or `true` indicates this hierarchy is show in the field list or similar mechanism in the user interface. A value of `0` or `false` indicates is not shown in the field list. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
subtotalTop (New Levels Subtotals At Top)	Specifies a boolean value that indicates whether new levels added to the view will show their subtotals at the top. A value of `1` or `true` indicates new levels added to the view show their subtotals at the top. A value of `0` or `false` indicates new levels added to the view show their subtotals at the bottom. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotHierarchy) is located in §A.2. *end note*]

18.10.1.73 pivotTableDefinition (PivotTable Definition)

Represents the PivotTable root element for non-null PivotTables. There exists one pivotTableDefinition for each PivotTableDefinition part. The PivotTable definition encompasses the following information:

Structure

- Top-level attributes

- Location information
- Collection of fields
- Fields on the row axis
- Items on the row axis (specific values)
- Fields on the column axis
- Items on the column axis (specific values)
- Fields on the report filter region
- Fields in the values region
- Style information

Outline of the XML for a pivotTableDefinition

```
<pivotTableDefinition>
  <location/>
  <pivotFields/>
  <rowFields/>
  <rowItems/>
  <colFields/>
  <colItems/>
  <pageFields/>
  <dataFields/>
  <conditionalFormats/>
  <pivotTableStyleInfo/>
</pivotTableDefinition>
```

Layout

The reference specified in the ref attribute on the location element specifies the location of the PivotTable body. The data area, row, column, and data fields and value items are located in this area. More specifically, the row fields begin below the A1-most cell in the reference, and the column fields begin adjacent to that cell, in the same row, extending out into the PivotTable body away from column A. *[Note:* How far below or across the field labels begin are dependent upon how many row, column, and data fields are shown in the PivotTable. More detail is provided below. *end note]*

[*Note*: All layout discussion and examples are given for outline mode layout. There two additional layout modes: compact and tabular. See Other layout modes below for a discussion of how those differ from outline mode. *end note*]

When encountering sheet boundaries, the PivotTable is truncated rather than wrapped, and as much as possible shall be shown.

The graphics given in this section are meant to illustrate layout only, and do not require implementation of any implied controls, like dropdowns or expand/collapse functionality.

Page Field Layout

	A	B	C	D
1				
2		SSN	(All)	
3				
4		**State**	**City**	**Sum of Amount**
5		**⊟CA**		**195.51**
6			San Diego	195.51
7		**⊟OR**		**54.97**
8			Portland	12.54
9			Tillamook	42.43
10		**⊟WA**		**244.12**
11			Seattle	96.72
12			Tacoma	79.83
13			Everett	67.57
14		**Grand Total**		**494.6**

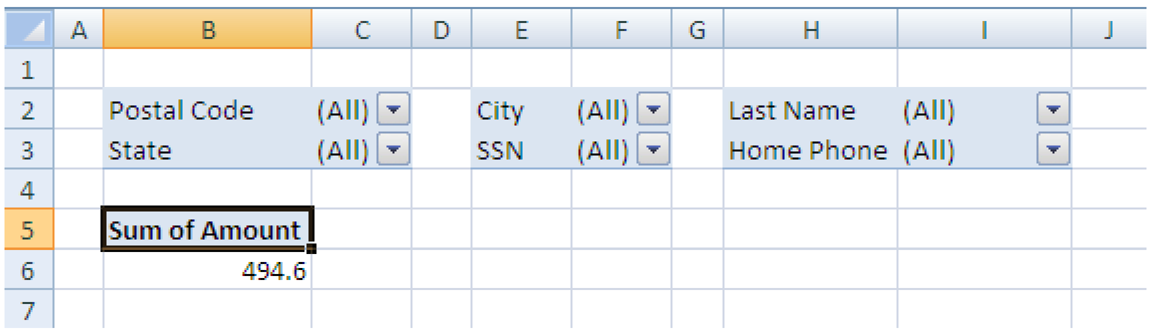
In the above picture, SSN is a page field, State and City are row fields, and Amount is a data field. There are no column fields.

Page fields allow you to filter the entire PivotTable report to display data for a single item or all items.

The page field area always ends (vertically) so that there is always 1 row of space between the page field area and the top row of the PivotTable body, and always begins (horizontally) in the same column as the A1-most column of the PivotTable body. Each page field occupies two cells: the A1-most for displaying the field name, and the next cell over for displaying the selected item values. [*Example*: (see above picture) If the top row in the PivotTable body reference is row 4, then page field layout ends (vertically) in row 2, and if the A1-most column of the PivotTable body is column B, then page field layout begins (horizontally) in column B. *end example*]

Aside from the number of fields in the page field area, there are two attributes of pivotTableDefinition that affect page field layout: pageOverThenDown and pageWrap. pageOverThenDown = 1 specifies that when there is more than 1 page field, lay them out horizontally across the sheet (extending in the direction of the PivotTable body area, away from column A) until the maximum specified in pageWrap is reached, and then begin a new row. If the pageWrap value is high and there are many page fields, then it is possible (and allowed) for page fields to extend beyond the edge of the PivotTable body. When laying out page fields in the same row (side by side), each shall be separated by a single column. However, multiple rows of page fields are not separated by single rows between them. pageOverThenDown = `0` specifies that when there is more than 1 page field, lay them out vertically down the sheet (always keeping 1 row of space between the PivotTable body and page field area) until the maximum specified in pageWrap is reached, and then begin a new column. Again, for multiple page fields, if they shall occupy more than 1 column, then each column of page fields is separated by a single column, and multiple rows of page fields are not separated by single rows between them.

[*Example:* This example shows a PivotTable body occupying B5:B6 and 6 page fields in the page field area, where pageOverThenDown = 0 and pageWrap = 2. This means that the first column of the page field area contains 2 page fields, and then, because the pageWrap value only allows 2 page fields per column, a new column of page fields is started, and so on until all 6 page fields are shown.



	A	B	C	D	E	F	G	H	I	J
1										
2		Postal Code	(All)		City	(All)		Last Name	(All)	
3		State	(All)		SSN	(All)		Home Phone	(All)	
4										
5		Sum of Amount								
6		494.6								
7										

The order of assignment of position within page field layout for this example is:

- Postal Code
- State
- City
- SSN
- Last Name
- Home Phone

Aside from the 6 page fields, the only other field in this PivotTable example is a data field called Amount.

end example]

[*Note:* When the user gestures to add a page field and there are not enough free cells above the PivotTable body area to allow for page fields to be added, the application must determine the best response. The application may decide to shift the PivotTable down some number of rows to make room, or overwrite existing data or features that might be above the PivotTable, or simply block the user gesture completely. In any result, however, the application should adhere to the layout principles given above. *end note*]

Row Field Layout

	A	B	C	D
1				
2		SSN	(All)	
3				
4		**State**	**City**	**Sum of Amount**
5		**CA**		**195.51**
6			San Diego	195.51
7		**OR**		**54.97**
8			Portland	12.54
9			Tillamook	42.43
10		**WA**		**244.12**
11			Seattle	96.72
12			Tacoma	79.83
13			Everett	67.57
14		**Grand Total**		**494.6**

The **State** and **City** fields are row fields, **SSN** is a page field, there are no column fields, and **Amount** is a data field.

Row fields provide for and specify how the data is summarized, grouped, and viewed as rows in the PivotTable.

The row field area always begins in the A1-most column of the PivotTable body area. The layout of page fields does not affect the layout of row fields.

Row Field Layout - 1 Row Field and 0 Column Fields

When there is only 1 row field and 0 column fields,

the first row field is located in the A1-most cell of the PivotTable body, and
the values for that field are expressed in the cells directly under that row field, in the same column.

[*Example:*

State	**Sum of Amount**
CA	195.51
OR	54.97
WA	244.12
Grand Total	**494.6**

In this example, there are no page fields, no column fields, **State** is a row field, and **Amount** is a data field. *end example*]

Row Field Layout - 2 or More Row Fields and 0 Column Fields

When there are 2 or more row fields and 0 column fields to be displayed,

- the row field labels are located adjacent to each other and in the same row as the first row field label
- Each corresponding set of values for the row field in question are located in the cells under that row field (same column)

- Innermost row field values (the ones closest to the data summary area) are grouped and organized by values in the next outer row field, in the following fashion: starting with the outermost row field, the first value is listed. For the next innermost row field, starting on the next row and over one column (toward the data summary area), the value list for that field begins. If that is the innermost row field, all values are listed for that row field, and then moving down a row and back to the outer column, the next value for the outermost row field is listed. If there are more inner row fields, the same layout rules apply until the innermost row field is reached.
- In this case of 0 column fields, only the top row of the PivotTable body is used for row field labels.

[*Example:*

	A	B	C	D	E
1					
2					
3					
4		Postal Co	State	City	Sum of Amount
5		⊟09999			54.97
6			⊟OR		54.97
7				Portland	12.54
8				Tillamook	42.43
9		⊟12345			195.51
10			⊟CA		195.51
11				San Diego	195.51
12		⊟456789			244.12
13			⊟WA		244.12
14				Seattle	96.72
15				Tacoma	79.83
16				Everett	67.57
17		Grand Total			494.6

In this example **Postal Code**, **State**, and **City** are row fields and **Amount** is a data field. There are no page fields and no column fields. *end example*]

Row Field Layout - 1 or More Row Fields and 1 or More Column Fields

When there are row fields and 1 or more column fields, the row fields are not located in the topmost row of the PivotTable body. Instead the row fields are located in the n+1st topmost row of the PivotTable body, where n is the number of column fields in the PivotTable.

[*Example:*

	A	B	C	D	E
1					
2					
3					
4					
5		Sum of Amount	Postal Code	State	City
6			⊟09999		
7			⊟OR		OR Total
8		Last Name	Portland	Tillamook	
9		Cencini	12.54		12.54
10		Freehafer			
11		Giussani			
12		Hellung-Larsen			
13		Kotas		42.43	42.43
14		Neipper			
15		Sergienko			
16		Thorpe			
17		Zare			
18		**Grand Total**	**12.54**	**42.43**	**54.97**

This example shows 3 column fields in the PivotTable (**Postal Code**, **State**, and **City**), a single row field **Last Name**, and a single data field **Amount**. The PivotTable body area begins at B5 and the row field label **Last Name** is located in the 4th row of the PivotTable body area, in row 8 of the spreadsheet, cell B8. Since **Last Name** is the only row field in this example, its row field values begin and are listed directly under the label. *end example*]

Column Field Layout

	A	B	C	D	E	F	G	H
1								
2								
3								
4								
5		Sum of Amount	State	City				
6			CA	OR		WA		
7		Last Name	San Diego	Portland	Tillamook	Seattle	Tacoma	Everett
8		Cencini		12.54				
9		Freehafer				53.34		
10		Giussani					79.83	
11		Hellung-Larsen						67.57
12		Kotas			42.43			
13		Neipper	63.67					
14		Sergienko	50.69					
15		Thorpe	81.15					
16		Zare				43.38		

State and **City** are column fields, **Last Name** is a row field, and **Amount** is a data field.

Column fields provide for and specify how the data is summarized, grouped, and viewed horizontally in the PivotTable.

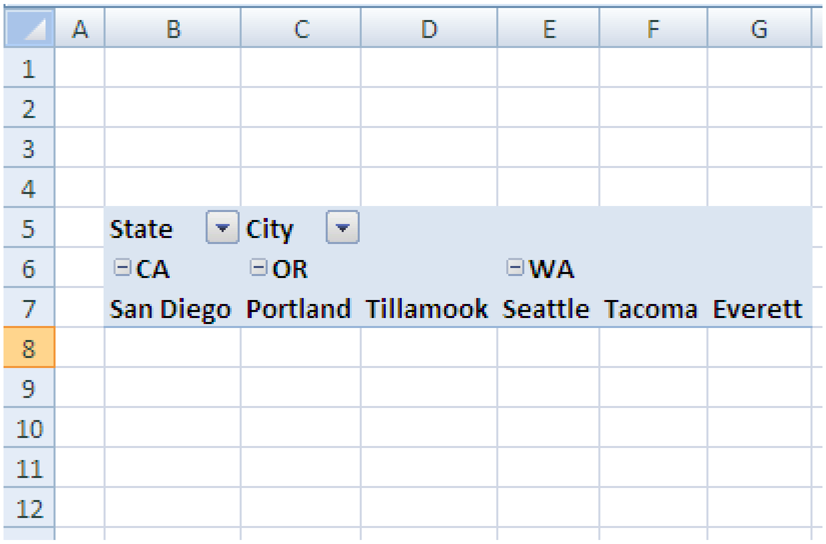
The layout of page fields does not affect the layout of column fields.

The column field label area is always located in the top row of the PivotTable body.

Column Field Layout - 0 Row Fields and 0 Data Fields

- When there are no row fields and no data fields, then the first column field is located in the A1-most column of the PivotTable body.
- When there are multiple column fields
- the labels are located adjacent to each other in the same row as the first column field label.
- Each corresponding set of values for each of the column fields are located in the rows directly below the column field label row and above the data area, one row of values for each column field.
- The first column field's values are located in the row directly under the column field row.
- Column field values are displayed starting directly underneath the first column field label's cell and filling adjacent cells in the same row. The second column field's values are located two rows under the column field label row, and values are again displayed starting directly underneath the first column field label's cell, filling adjacent cells in the same manner as the first set of values. The layout of column field values continues in this way until all column field values are displayed.

Inner column field values (the ones closer to the data summary area) are grouped and organized by values in the next outer column field, similarly to how row field values are grouped. [*Example:*



In this example, **State** and **City** are column fields, and there are no row fields, no page fields, and no data fields. *end example*]

Column Field Layout – 1 or More Column Fields and 1 or More Row Fields

When there are 1 or more column fields and 1 or more row fields in the PivotTable, then:

- First, row fields are displayed according to the row field layout described earlier
- The first column field label is located in the top row of the PivotTable body area, and adjacent to any row field labels that are displayed.
- Multiple column fields shall be displayed as described earlier

[*Example:* In this example, **State** and **City** are column fields, **Amount** is a data field, and **Last Name** is a row field.

	A	B	C	D	E	F	G	H
1								
2								
3								
4								
5		**Sum of Amount**	**State**	**City**				
6			**CA**	**OR**		**WA**		
7		**Last Name**	**San Diego**	**Portland**	**Tillamook**	**Seattle**	**Tacoma**	**Everett**
8		Cencini		12.54				
9		Freehafer				53.34		
10		Giussani					79.83	
11		Hellung-Larsen						67.57
12		Kotas			42.43			
13		Neipper	63.67					
14		Sergienko	50.69					
15		Thorpe	81.15					
16		Zare				43.38		

end example]

Data Field Layout

	A	B	C
1			
2			
3			
4			
5		**Last Name**	**Sum of Amount**
6		Cencini	12.54
7		Freehafer	53.34
8		Giussani	79.83
9		Hellung-Larsen	67.57
10		Kotas	42.43
11		Neipper	63.67
12		Sergienko	50.69
13		Thorpe	81.15
14		Zare	43.38

Last Name is a row field, Sum of Amount is a data field label, and the data underneath Sum of Amount are the summarized data values.

Data fields specify which fields are summarized in the PivotTable report.

The summarized data always appears below the column field and value area, and any row field values are closer to column A than any of the summarized data. When there are no row fields and no column fields, the summarized data is located directly under the A1-most cell of the PivotTable body. Each cell in the summarized data area represents an aggregation of a set of records. The set of records that a particular cell is summarizing is determined by looking at the row field value(s) and column field value(s) that intersect on that particular cell, and then determining which records in the source data contain all of those row and column field values.

Data Field Layout - 0 Row Fields and 0 Column Fields and 1 Data Field

When there are no row fields and no column fields and only 1 data field being summarized, the data field label is located in the A1-most cell of the PivotTable body.

[*Example:*

	A	B
1		
2		Sum of Amount
3		494.6

In this example there is only 1 field in the PivotTable, a data field **Amount**. *end example*]

Data Field Layout - More Than 1 Data Field

When there is more than 1 data field being summarized,

- An additional field (in these examples labeled “Values”, but the label can be specified by the user) is added to the field list, located as either a row field label or a column field label (depending on user choice and behaviour as specified by the dataOnRows and dataPosition attributes), and
- each data field being summarized is displayed either in the row area (when the additional field is a row field) as if it were an item value of that row field (see row field layout description above), or in the column area (when the additional field is a column field) as if it were an item value of that column field (see column field layout description above).

[*Example:*

	A	B	C
1			
2		Values	
3		Sum of Amount	Sum of Tax
4		494.6	28.71

In this example there are 2 data fields **Amount** and **Tax**. There are no page fields, no column fields, no row fields, and the additional field labeled Values is placed on the column area.

	A	B	C
1			
2		**Values**	
3		Sum of Amount	494.6
4		Sum of Tax	28.71

Above is the same PivotTable, with the Values field placed on the row area.

end example]

Data Field Layout - 0 Row Fields, 1 or More Column Fields, and 1 Data Field

When there are no row fields, 1 or more column fields, and only 1 data field being summarized, the data field label is located in the A1-most column of the PivotTable body, directly under the column field area.

[*Example:*

	A	B	C	D	E
1					
2			**State**		
3			**CA**	**OR**	**WA**
4		Sum of Amount	195.51	54.97	244.12

In this example there is 1 column field **State** and 1 data field **Amount**. There are no row fields or page fields. *end example*]

Data Field Layout - 0 Column Fields, 1 or More Row Fields, and 1 Data Field

When there are no column fields, 1 or more row fields, and only 1 data field being summarized, the data field label is located in the same row as the row field labels, above the data summary area.

[*Example:*

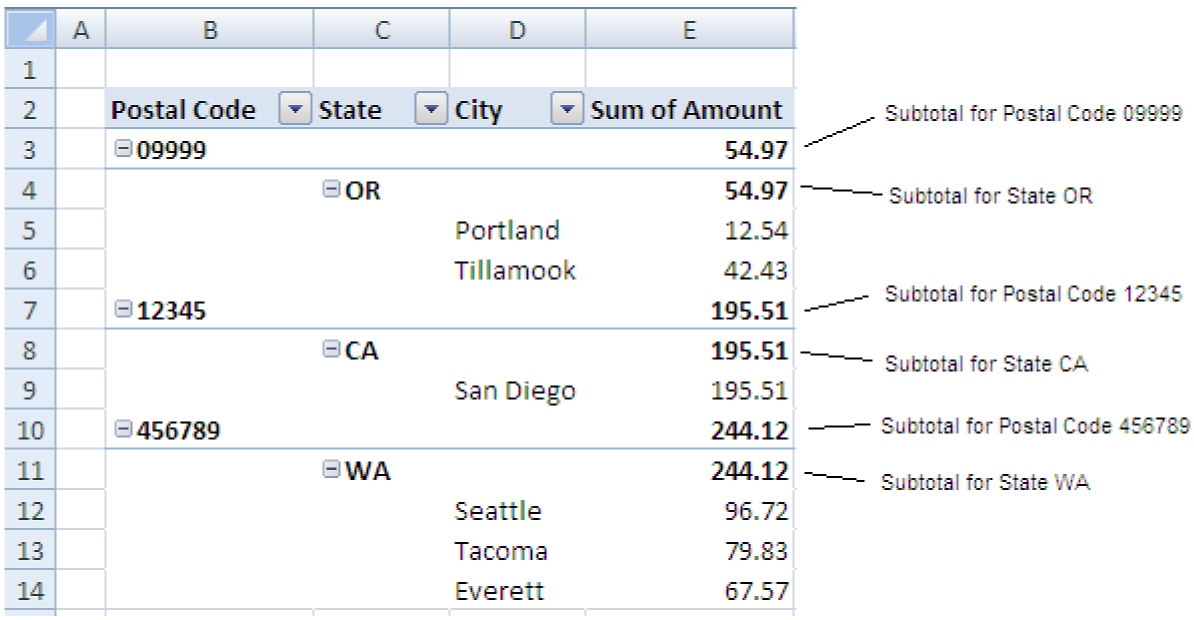
	A	B	C
1			
2		**State**	**Sum of Amount**
3		CA	195.51
4		OR	54.97
5		WA	244.12

In this example there is 1 data field **Amount** and 1 row field **State**. There are no column fields or page fields. *end example*]

Subtotal and grand total layout

If subtotals are *on*, the values for row subtotals are placed at either the top of each group of data being summarized or at the bottom of each group, as indicated by the subtotalTop attribute value on the pivotField element. Row subtotal values appear in the same column as the data being subtotalled. If placed at the top of the group, then the subtotal value for the group appears in the row above the group of values, in the same row as the group's parent row field value. When there is only a single row field, no subtotal is shown.

[*Example:*

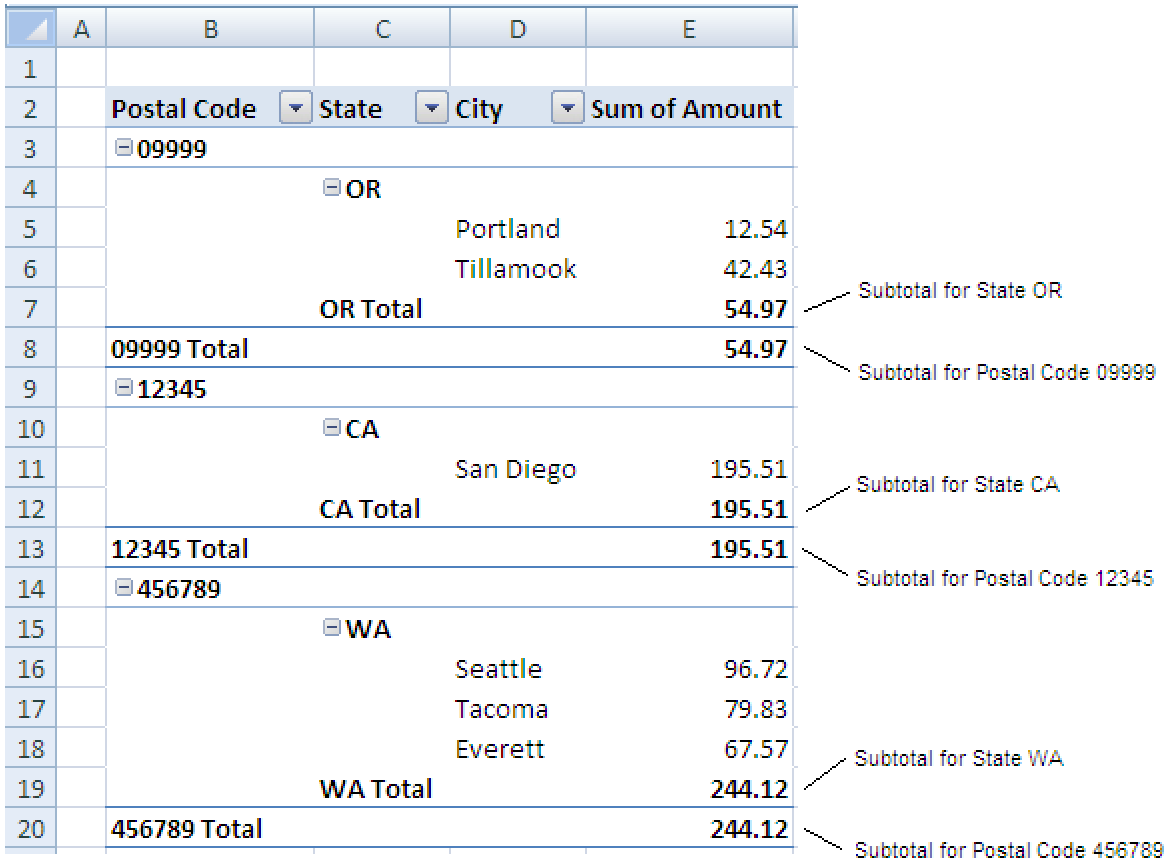


In this example, there are 3 row fields (**Postal Code**, **State**, and **City**) and 1 data field **Amount**.

end example]

If row subtotals are placed at the bottom of each data group, then a new row is inserted directly below the data group in question, and a new row field value is inserted, in the same column as the row field in question, whose caption indicates that this row represents a subtotal value.

[*Example:*



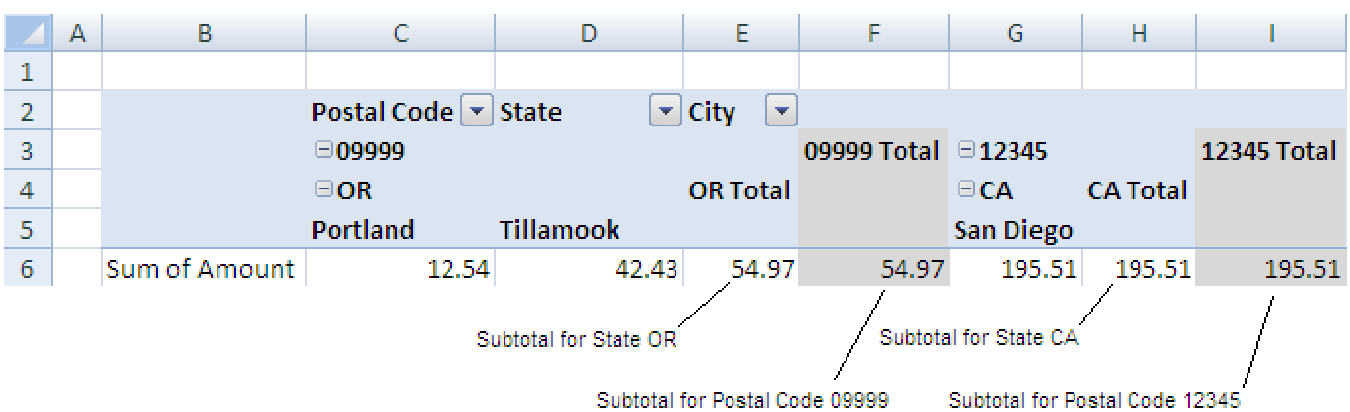
	A	B	C	D	E
1					
2		**Postal Code**	**State**	**City**	**Sum of Amount**
3		**⊟09999**			
4			**⊟OR**		
5				Portland	12.54
6				Tillamook	42.43
7			**OR Total**		**54.97**
8		**09999 Total**			**54.97**
9		**⊟12345**			
10			**⊟CA**		
11				San Diego	195.51
12			**CA Total**		**195.51**
13		**12345 Total**			**195.51**
14		**⊟456789**			
15			**⊟WA**		
16				Seattle	96.72
17				Tacoma	79.83
18				Everett	67.57
19			**WA Total**		**244.12**
20		**456789 Total**			**244.12**

In this example, there are 3 row fields (**Postal Code**, **State**, and **City**) and 1 data field **Amount**.

end example]

If subtotals are *on*, for column subtotals a new column is inserted directly after the data group being subtotalled. A new column field value is inserted, in the same row as the column field in question, whose caption indicates that this column represents a subtotal value. When there is only a single column field, no subtotal is shown.

[*Example:*



	A	B	C	D	E	F	G	H	I
1									
2			**Postal Code**	**State**	**City**				
3			**⊟09999**			**09999 Total**	**⊟12345**		**12345 Total**
4			**⊟OR**		**OR Total**		**⊟CA**	**CA Total**	
5			**Portland**	**Tillamook**			**San Diego**		
6		Sum of Amount	12.54	42.43	54.97	54.97	195.51	195.51	195.51

In this example, there are 3 column fields (**Postal Code**, **State**, and **City**) and a data field **Amount**. *end example*]

If row grand totals are on and there are column fields, a new column item is inserted at the very edge of the PivotTable body furthest away from column A, in the same row as the outermost column field values. The caption indicates that this is a grand total, and the values total all values across the row. When row grand totals are *on* but there are no column fields, no row grand total is shown.

[*Example:*

	A	B	C	D	E	F
1						
2			State			
3			CA	OR	WA	Grand Total
4		Sum of Amount	195.51	54.97	244.12	494.6

In this example there is 1 column field **State** and 1 data field **Amount**, and row grand totals are on.

end example]

When column grand totals are *on* and there are row fields, a new row item is inserted at the very bottom of the PivotTable body, in the same column as the outermost row field values. The caption indicates that this is a grand total, and the values total all values in the column. When column grand totals are *on* but there are no row fields, no column grand total is shown.

[*Example:*

	A	B	C
1			
2		State	Sum of Amount
3		CA	195.51
4		OR	54.97
5		WA	244.12
6		Grand Total	494.6

In this example there is 1 row field **State** and 1 data field **Amount, and column grand totals are on**.

end example]

Other Layout Modes

A PivotTable can be displayed in Compact, Outline, or Tabular form. In addition, Classic layout can be applied to any of the 3 layout forms.

Outline mode has been discussed in the above sections, and all examples are shown using outline mode with classic layout off (gridDropZones = 0).

For Compact mode, the layout differs from outline mode by:

- Instead of multiple row fields occupying multiple columns, the A1-most column of the PivotTable body contains all row field labels and values. A single label, “Row Labels”, is located where the first (outermost) row label is placed. When there are multiple row fields, the outermost list of values is not indented, then next inner row field values are indented (as specified in the indent attribute), and so on until each set of values for inner row fields are shown.
- Instead of multiple column fields being listed and located across a row, the first column field position is labeled “Column Labels”, and there is only this label, located in the first column field position.

[*Example:*

Outline mode:

	A	B	C	D	E	F	G
1							
2		**Sum of Amount**		**Postal Code**	**Last Name**		
3				⊟**09999**		⊟**12345**	
4		**State**	**City**	**Cencini**	**Kotas**	**Neipper**	**Sergienko**
5		⊟**CA**					
6			San Diego			63.67	50.69
7		⊟**OR**					
8			Portland	12.54			
9			Tillamook		42.43		
10		⊟**WA**					
11			Seattle				
12			Tacoma				
13			Everett				

The above picture shows 2 column fields (**Postal Code** and **Last Name**), 1 data field (**Amount**), and 2 row fields (**State** and **City**). There are no page fields shown.

Same PivotTable in compact mode:

	A	B	C	D	E	F
1						
2		Sum of Amount	Column Labels			
3			⊟09999		⊟12345	
4		Row Labels	Cencini	Kotas	Neipper	Sergienko
5		⊟CA				
6		San Diego			63.67	50.69
7		⊟OR				
8		Portland	12.54			
9		Tillamook		42.43		
10		⊟WA				
11		Seattle				
12		Tacoma				
13		Everett				

The above picture shows all column field labels collapsed into a single label **Column Labels** and all row field labels collapsed into a single label **Row Labels**. There is 1 data field **Amount** and no page fields. *end example*]

For Tabular mode, the layout differs from outline mode by:

Instead of beginning new inner row field values on the next row down from the outer row field value parent, the first next-inner row field value is located on the same row as the parent value.
Row subtotals can only appear at the bottom of a group, not at the top

[*Example:*

Outline mode:

	A	B	C	D	E
1					
2		Postal Code	State	City	Sum of Amount
3		⊟09999			
4			⊟OR		
5				Portland	12.54
6				Tillamook	42.43
7		⊟12345			
8			⊟CA		
9				San Diego	195.51
10		⊟456789			
11			⊟WA		
12				Seattle	96.72
13				Tacoma	79.83
14				Everett	67.57

The above picture shows 3 row fields (**Postal Code**, **State**, and **City**) and 1 data field, **Amount**.

Same PivotTable in tabular mode:

	A	B	C	D	E
1					
2		Postal Code	State	City	Sum of Amount
3		⊟09999	⊟OR	Portland	12.54
4				Tillamook	42.43
5		⊟12345	⊟CA	San Diego	195.51
6		⊟456789	⊟WA	Seattle	96.72
7				Tacoma	79.83
8				Everett	67.57

The above picture shows 3 row fields (**Postal Code**, **State**, and **City**) and 1 data field, **Amount**.

end example]

For Classic layout, the layout differs by:

- When there are row fields, no column fields, and 1 data field, instead of displaying the data field label adjacent to and in the same row as the row field labels, the data field label is located in the A1-most cell of the PivotTable body, and the row directly under this cell contains the row field labels.
- In the exact location where the data field label is located when classic layout is off, a label titled "Total" is displayed when classic layout is on.

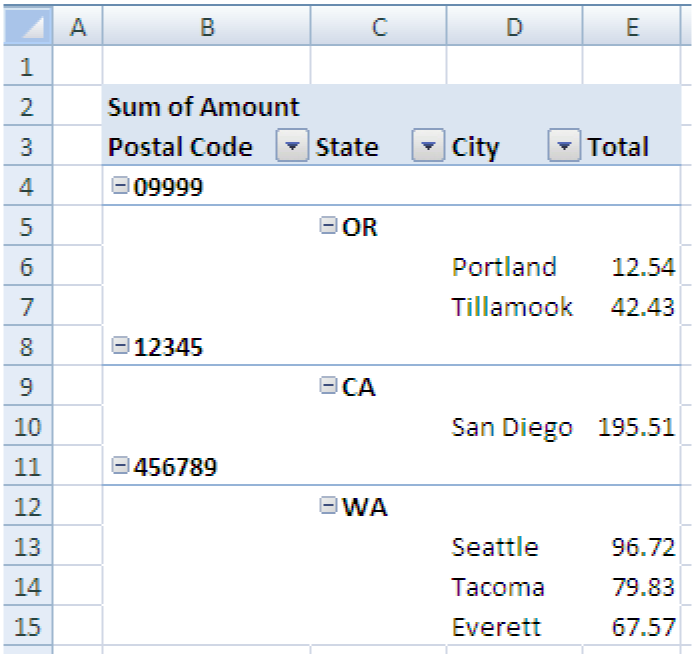
[*Example:*

Outline mode, classic layout off:

	A	B	C	D	E
1					
2		Postal Code	State	City	Sum of Amount
3		⊟09999			
4			⊟OR		
5				Portland	12.54
6				Tillamook	42.43
7		⊟12345			
8			⊟CA		
9				San Diego	195.51
10		⊟456789			
11			⊟WA		
12				Seattle	96.72
13				Tacoma	79.83
14				Everett	67.57

The above picture shows 3 row fields (**Postal Code**, **State**, and **City**) and 1 data field, **Amount**.

Same PivotTable in Outline mode, classic layout applied:



	A	B	C	D	E
1					
2		Sum of Amount			
3		Postal Code	State	City	Total
4		⊟09999			
5			⊟OR		
6				Portland	12.54
7				Tillamook	42.43
8		⊟12345			
9			⊟CA		
10				San Diego	195.51
11		⊟456789			
12			⊟WA		
13				Seattle	96.72
14				Tacoma	79.83
15				Everett	67.57

The above picture shows 3 row fields (**Postal Code**, **State**, and **City**) and 1 data field, **Amount**.

end example]

Attributes	Description
applyAlignmentFormats (Apply Alignment Formats)	If `true` apply legacy table autoformat alignment properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyBorderFormats (Apply Border Formats)	If `true` apply legacy table autoformat border properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyFontFormats (Apply Font Formats)	If `true` apply legacy table autoformat font properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyNumberFormats (Apply Number Formats)	If `true` apply legacy table autoformat number format properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyPatternFormats (Apply Pattern Formats)	If `true` apply legacy table autoformat pattern properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
applyWidthHeightFormats (Apply Width / Height Formats)	If `true` apply legacy table autoformat width/height properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
asteriskTotals (Asterisk Totals)	Specifies a boolean value that indicates whether an asterisks should be displayed in subtotals and totals when visual totals are not used in OLAP -based PivotTables. A value of `1` or `true` indicates an asterisks are displayed in subtotals and totals for OLAP PivotTables when visual tools are not available. A value of `0` or `false` indicates an asterisk will not be displayed. This attribute depends on the implementation and availability of visual tools in the application user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoFormatId (Auto Format Id)	Identifies which legacy table autoformat to apply. Annex D contains a listing of the supported PivotTable AutoFormats, example formatting, and a sample workbook with each of those AutoFormats applied. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
cacheId (PivotCache Definition Id)	Specifies the identifier of the related PivotCache definition. This Id is listed in the pivotCaches collection in the workbook part. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
chartFormat (Chart Format Id)	Specifies the next chart formatting identifier to use on the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
colGrandTotals (Grand Totals On Columns)	Specifies a boolean value that indicates whether grand totals should be displayed for the PivotTable columns. A value of `1` or `true` indicates grand totals should be displayed. A value of `0` or `false` indicates grand totals should not be displayed for PivotTable columns. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
colHeaderCaption (Column Header Caption)	Specifies the string to be displayed in column header in compact mode. This attribute depends on whether the application implements a compact mode for displaying PivotTables in the user interface. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
compact (Compact New Fields)	Specifies a boolean value that indicates whether new fields should have their `compact` flag set to true. A value of `1` or `true` indicates new fields should default to `compact` mode equal to true. A value of `0` or `false` indicates new fields should default to `compact` mode equal to false. This attribute depends on whether the application implements a compact mode in the user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
compactData (Compact Data)	Specifies a boolean value that indicates whether the field next to the data field in the PivotTable should be displayed in the same column of the spreadsheet The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
createdVersion (PivotCache Created Version)	Specifies the version of the application that created the cache. This attribute is application-dependent. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
customListSort (Custom List AutoSort)	Specifies a boolean value that indicates whether the "custom lists" option is offered when sorting this PivotTable. A value of `1` or `true` indicates custom lists are offered when sorting this PivotTable. A value of `0` or `false` indicates custom lists are not offered. This attribute depends on the implementation of sorting features in the application. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dataCaption (Data Field Header Name)	Specifies the name of the value area field header in the PivotTable. This caption is shown when the PivotTable when two or more fields are in the values area. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
dataOnRows (Data On Rows)	Specifies a boolean value that indicates whether the field representing multiple fields in the data region is located in the row area or the column area.

Attributes	Description
	A value of 1 or true indicates that this field is located in the row area. A value of 0 or false indicates that this field is located in the column area. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dataPosition (Default Data Field Position)	Specifies the position for the field representing multiple data field in the PivotTable, whether that field is located in the row area or column area. Missing attribute indicates this field is last, or innermost in the field list. 0 indicates this field is first, or outermost in the field list. 1 indicates this field is second in the field list. 2 indicates this field is third in the field list, and increasing values follow this pattern. If this value is higher than the number of fields in the field list, then this field is last, or innermost in the field list. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
disableFieldList (Disable Field List)	Specifies a boolean value that indicates whether to disable the PivotTable field list. A value of 1 or true indicates the field list, or similar mechanism for selecting fields in the user interface, is disabled. A value of 0 or false indicates the field list is enabled. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
editData (Allow Edit Data)	Specifies a boolean value that indicates whether the user is allowed to edit the cells in the data area of the PivotTable. A value of 1 or true indicates the user can edit values in the data area. A value of 0 or false indicates the cells in the data area are not editable. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
enableDrill (Enable Drill Down)	Specifies a boolean value that indicates whether the user is prevented from drilling down on a PivotItem or aggregate value. A value of 1 or true indicates the user can drill down on a pivot item or aggregate value.

Attributes	Description
	A value of 0 or false indicates the user is prevented from drilling down pivot item. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
enableFieldProperties (Enable Field Properties)	Specifies a boolean value that indicates whether the user is prevented from displaying PivotField properties. A value of 1 or true indicates the user can display pivot field properties. A value of 0 or false indicates the user cannot display pivot field properties. This attribute depends on how pivot field properties are exposed in the application user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
enableWizard (Enable PivotTable Wizard)	Specifies a boolean value that indicates whether the user is prevented from displaying the PivotTable wizard. A value of 1 or true indicates the user can display the PivotTable wizard. A value of 0 or false indicates the user can not display the PivotTable wizard. This attribute depends on whether the application exposes a wizard or similar mechanism for creating and working with PivotTables in the user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
errorCaption (Error Caption)	Specifies the string to be displayed in cells that contain errors. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
fieldListSortAscending (Default Sort Order)	Specifies a boolean value that indicates whether fields in the PivotTable are sorted in non-default order in the field list. A value of 1 or true indicates fields for the PivotTable are sorted in the field list. The sort order from the data source is applied for range-based PivotTables. Alphabetical sorting is applied for external data PivotTables. A value of 0 or false indicates fields in the field list are not sorted. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
fieldPrintTitles (Field Print Titles)	Specifies a boolean value that indicates whether the row and column titles from the PivotTable should be printed.

Attributes	Description
	A value of 1 or true indicates row and column titles should be printed. A value of 0 or false indicates row and column titles should not be printed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
grandTotalCaption (Grand Totals Caption)	Specifies the string to be displayed for grand totals. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
gridDropZones (Enable Drop Zones)	Specifies a boolean value that indicates whether the in-grid drop zones should be displayed at runtime, and whether classic layout is applied. A value of 1 or true indicates in-grid drop zones should be displayed and classic layout should be applied to the PivotTable. A value of 0 or false indicates in-grid drop zones should be disabled and classic layout should not be applied. [*Note:* Grid drop zones are optional runtime UI, determined by the application, that indicate to the user the locations of the page, row, column, and data fields in the PivotTable report. See layout discussion under pivotTableDefinition for the precise locations of these areas. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
immersive (Stop Immersive UI)	Specifies a boolean value that indicates whether PivotTable immersive experience user interface should be turned off. A value of 1 or true indicates the PivotTable immersive experience should be turned off for this PivotTable. A value of 0 or false indicates the immersive experience should be left on. This attribute depends on whether the application implements an immersive experience in the user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
indent (Indentation for Compact Axis)	Specifies the indentation increment for compact axis and can be used to set the Report Layout to Compact Form. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

Attributes	Description
itemPrintTitles (Item Print Titles)	Specifies a boolean value that indicates whether PivotItem names should be repeated at the top of each printed page. A value of 1 or true indicates pivot items names should be repeated at the top of each page. A value of 0 or false indicates should not be repeated. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
mdxSubqueries (MDX Subqueries Supported)	Specifies a boolean value that indicates whether MDX sub-queries are supported by OLAP data provider for this PivotTable. A value of 1 or true indicates MDX sub-queries are supported by the OLAP data provider. A value of 0 or false indicates MDX sub-queries are not supported. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
mergeItem (Merge Titles)	Specifies a boolean value that indicates whether row or column titles that span multiple cells should be merged into a single cell. A value of 1 or true indicates that titles that span multiple cells are merged into a single cell. A value of 0 or false indicates titles are not merged. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
minRefreshableVersion (Minimum Refreshable Version)	Specifies the minimum version of the application required to update this PivotTable view. This attribute is application-dependent. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
missingCaption (Caption for Missing Values)	Specifies the string to be displayed in cells with no value The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
multipleFieldFilters (Multiple Field Filters)	Specifies a boolean value that indicates whether the fields of a PivotTable can have multiple filters set on them. A value of 1 or true indicates the fields of a PivotTable can have multiple filters. A value of 0 or false indicates the fields of a PivotTable can only have a simple filter.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
name (Name)	Specifies the PivotTable name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
outline (Outline New Fields)	Specifies a boolean value that indicates whether new fields should have their `outline` flag set to true. A value of `1` or `true` indicates new fields are created with `outline` equal to true. A value of `0` or `false` indicates new fields are created with `outline` equal to false. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
outlineData (Outline Data Fields)	Specifies a boolean value that indicates whether data fields in the PivotTable should be displayed in outline form. A value of `1` or `true` indicates data fields will display in outline form. A value of `0` or `false` indicates data fields will not display in outline form. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
pageOverThenDown (Page Over Then Down)	Specifies a boolean value that indicates how the page fields are laid out when there are multiple PivotFields in the page area. A value of `1` or `true` indicates the fields will display "Over, then down" A value of `0` or `false` indicates the fields will display "down, then Over" The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
pageStyle (Page Header Style Name)	Specifies the name of the style to apply to each of the field item headers in the page area of the PivotTable. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
pageWrap (Page Wrap)	Specifies the number of page fields to display before starting another row or column. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
pivotTableStyle	Specifies the name of the style to apply to the main table area of the PivotTable.

Attributes	Description
(Table Style Name)	The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
preserveFormatting (Preserve Formatting)	Specifies a boolean value that indicates whether the formatting applied by the user to the PivotTable cells is discarded on refresh. A value of `1` or `true` indicates the formatting applied by the end user is discarded on refresh. A value of `0` or `false` indicates the end-user formatting is retained on refresh. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
printDrill (Print Drill Indicators)	Specifies a boolean value that indicates whether drill indicators expand collapse buttons should be printed. A value of `1` or `true` indicates that these buttons should be printed. A value of `0` or `false` indicates that these buttons should not be printed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
published (Data Fields Published)	Specifies a boolean value that indicates whether data fields in the PivotTable are published and available for viewing in a server rendering environment. A value of `1` or `true` indicates that the data fields in the PivotTable are published and shall be available for viewing in a server rendering environment. A value of `0` or `false` indicates that the data fields in the PivotTable are not published and shall not be available for viewing in a server rendering environment. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rowGrandTotals (Row Grand Totals)	Specifies a boolean value that indicates whether grand totals should be displayed for the PivotTable rows. The default value for this attribute is true. A value of `1` or `true` indicates grand totals are displayed for the PivotTable rows. A value of `0` or `false` indicates grand totals will not be displayed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rowHeaderCaption (Row Header	Specifies the string to be displayed in row header in `compact` mode.

Attributes	Description
Caption)	The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
showCalcMbrs (Show Calculated Members)	Specifies a boolean value that indicates whether calculated members should be shown in the PivotTable view. This attribute applies to PivotTables from OLAP-sources only. A value of `1` or `true` indicates that calculated members should be shown. A value of `0` or `false` indicates calculated members should not be shown. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showDataDropDown (Show Drop Down)	Specifies a boolean value that indicates whether the drop-down lists for the fields in the PivotTable should be hidden. This attribute depends on whether the application implements drop down lists or similar mechanism in the user interface. A value of `1` or `true` indicates drop down lists are displayed for fields. A value of `0` or `false` indicates drop down lists will not be displayed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showDataTips (Show ToolTips on Data)	Specifies a boolean value that indicates whether tooltips should be displayed for PivotTable data cells. A value of `1` or `true` indicates tooltips are displayed. A value of `0` or `false` indicates tooltips will not be displayed. This attribute depends on whether the application employs tooltips or similar mechanism in the user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showDrill (Show Expand Collapse)	Specifies a boolean value that indicates whether drill indicators should be hidden. A value of `1` or `true` indicates drill indicators are displayed. A value of `0` or `false` indicates drill indicators will not be displayed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showDropZones (Show Drop Zones)	Specifies a boolean value that indicates whether the PivotTable should display large drop zones when there are no fields in the data region. A value of `1` or `true` indicates a large drop zone is displayed.

Attributes	Description
	A value of 0 or false indicates a large drop zone will not be displayed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showEmptyCol (Show Empty Column)	Specifies a boolean value that indicates whether to include empty columns in the table. A value of 1 or true indicates empty columns are included in the PivotTable. A value of 0 or false indicates empty columns are excluded. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showEmptyRow (Show Empty Row)	Specifies a boolean value that indicates whether to include empty rows in the table. A value of 1 or true indicates empty rows are included in the PivotTable. A value of 0 or false indicates empty rows are excluded. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showError (Show Error)	Specifies a boolean value that indicates whether to show error messages in cells. A value of 1 or true indicates error messages are shown in cells. A value of 0 or false indicates error messages are shown through another mechanism the application provides in the user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showHeaders (Show Field Headers)	Specifies a boolean value that indicates whether to suppress display of pivot field headers. A value of 1 or true indicates field headers are shown in the PivotTable. A value of 0 or false indicates field headers are excluded. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showItems (Show Item Names)	Specifies a boolean value that indicates whether to display item names when adding a field onto a PivotTable that has no data fields. A value of 1 or true indicates item names are displayed. A value of 0 or false indicates item names will not be displayed.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showMemberPropertyTips (Show Member Property ToolTips)	Specifies a boolean value that indicates whether member property information should be omitted from PivotTable tooltips. A value of `1` or `true` indicates member property information is included. A value of `0` or `false` indicates member property information is excluded. This attribute depends on whether the application employs tooltips or similar mechanism in the user interface. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showMissing (Show Missing)	Specifies a boolean value that indicates whether to show a message in cells with no value. A value of `1` or `true` indicates to show a message string in cells without values. A value of `0` or `false` indicates no message string will shown in cells without values. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showMultipleLabel (Show Multiple Labels)	Specifies a boolean value that indicates whether a page field with multiple selected items should display "(multiple items)" instead of "All". This attribute applies only to non-OLAP PivotTables. The messages displayed depend on the application implementation. A value of `1` or `true` indicates a different message string is displayed for a page field with multiple items. A value of `0` or `false` indicates the same message string is displayed for all page fields. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
subtotalHiddenItems (Subtotal Hidden Items)	Specifies a boolean value that indicates whether data for hidden pivotItems for PivotFields in the data area should be included in subtotals. A value of `1` or `true` indicates that data for hidden pivot items in the data area is included in subtotals. A value of `0` or `false` indicates hidden pivot items will not be included in subtotals. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
tag (PivotTable Custom String)	Specifies a user-defined string that is associated with this PivotTable. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
updatedVersion (PivotTable Last Updated Version)	Specifies the version of the application that last updated the PivotTable view. This attribute is application-dependent. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
useAutoFormatting (Auto Formatting)	Specifies a boolean value that indicates whether legacy auto formatting has been applied to the PivotTable view. A value of `1` or `true` indicates that legacy auto formatting has been applied to the PivotTable. A value of `0` or `false` indicates that legacy auto formatting has not been applied to the PivotTable. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
vacatedStyle (Vacated Style)	Specifies the name of the style to apply to the cells left blank when a PivotTable shrinks during a refresh operation The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
visualTotals (Total Visual Data)	Specifies a boolean value that indicates whether totals should be based on visible data only. This attribute applies to OLAP PivotTables only. A value of `1` or `true` indicates subtotals are computed on visible data only. A value of `0` or `false` indicates subtotals are computed on all data. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_pivotTableDefinition) is located in §A.2. *end note*]

18.10.1.74 pivotTableStyleInfo (PivotTable Style)

Represent information on style applied to the PivotTable.

[*Example*:


```
<sh:pivotTableStyleInfo name="PivotStyleLight16" showRowHeaders="1"
  showColHeaders="1" showRowStripes="0" showColStripes="0" showLastColumn="1"/>
```

end example]

Attributes	Description
name (Table Style Name)	Specifies the name of the table style to use with this table. The possible values for this attribute are defined by the W3C XML Schema string datatype.
showColHeaders (Show Table Style Column Header Formatting)	Specifies a boolean value that indicates whether to show column headers for the table. A value of `1` or `true` indicates column headers are shown. A value of `0` or `false` indicates column headers are omitted. 'True' if table style column header formatting should be displayed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showColStripes (Show Column Stripes)	Specifies a boolean value that indicates whether to show column stripe formatting for the table. A value of `1` or `true` indicates column stripe formatting is shown. A value of `0` or `false` indicates no column formatting is shown. True if table style column stripe formatting should be displayed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showLastColumn (Show Last Column)	Specifies a boolean value that indicates whether to show the last column. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showRowHeaders (Show Row Header Formatting)	Specifies a boolean value that indicates whether to show row headers for the table. A value of `1` or `true` indicates table style formatting is displayed. A value of `0` or `false` indicates table style formatting will not be displayed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showRowStripes (Show Row Stripes)	Specifies a boolean value that indicates whether to show row stripe formatting for the table.

Attributes	Description
	A value of `1` or `true` indicates row stripe formatting is displayed. A value of `0` or `false` indicates no row formatting is shown. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotTableStyle) is located in §A.2. *end note*]

18.10.1.75 query (Query)

Represents an OLAP sheet data cached query.

Attributes	Description
mdx (MDX Query String)	Specifies the Multidimensional Expressions (MDX) query string. [*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn2.microsoft.com/en-us/library/ms145595.aspx *end note*] See the MDX Language Reference for more information: The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Query) is located in §A.2. *end note*]

18.10.1.76 queryCache (OLAP Query Cache)

Represents the cache of OLAP sheet data queries.

Attributes	Description
count (Cached Query Count)	Specifies the number of cached queries in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_QueryCache) is located in §A.2. *end note*]

18.10.1.77 r (PivotCache Record)

Represents a single record of data in the PivotCache.

[*Example*:

```
<r>
  <s v="3550"/>
  <s v="Road-150 Red, 62"/>
  <s v="This bike is ridden by race winners. Developed with the Adventure Works
    Cycles professional race team, it has a extremely light heat-treated
    aluminum frame, and steering that allows precision control."/>
  <s v="No Discount"/>
  <x v="0"/>
  <s v="Australian Dollar"/>
  <n v="1"/>
  <n v="3578.27"/>
  <n v="0"/>
  <n v="2171.2941999999998"/>
  <n v="3578.27"/>
  <n v="89.456800000000001"/>
</r>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Record) is located in §A.2. *end note*]

18.10.1.78 rangePr (Range Grouping Properties)

Represents the collection of range grouping properties.

[*Example*:

```
<rangePr groupBy="months" startDate="2002-01-01T00:00:00"
  endDate="2006-05-06T00:00:00"/>
```

end example]

Attributes	Description
autoEnd (Source Data Ending Range)	Specifies a boolean value that indicates whether the application uses the source data to set the ending range value. A value of `1` or `true` indicates the ending range value is set from the source data. A value of `0` or `false` indicates ending range values are set by the value specified in endDate or endNum.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoStart (Source Data Set Beginning Range)	Specifies a boolean value that indicates whether we use source data to set the beginning range value. A value of `1` or `true` indicates the beginning range value is set from the source data. A value of `0` or `false` indicates the beginning range value is set from the value specified in startDate or startNum. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
endDate (Date Grouping End Value)	Specifies the ending value for date grouping if `autoEnd` is false. The possible values for this attribute are defined by the W3C XML Schema dateTime datatype.
endNum (Numeric Grouping End Value)	Specifies the ending value for numeric grouping if `autoEnd` is false. The possible values for this attribute are defined by the W3C XML Schema double datatype.
groupBy (Group By)	Specifies the grouping. The possible values for this attribute are defined by the ST_GroupBy simple type (§18.18.38).
groupInterval (Grouping Interval)	Specifies the grouping interval for numeric range grouping. Specifies the number of days to group by in date range grouping. The possible values for this attribute are defined by the W3C XML Schema double datatype.
startDate (Date Grouping Start Value)	Specifies the starting value for date grouping if `autoStart` is false. The possible values for this attribute are defined by the W3C XML Schema dateTime datatype.
startNum (Numeric Grouping Start Value)	Specifies the starting value for numeric grouping if `autoStart` is false. The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RangePr) is located in §A.2. *end note*]

18.10.1.79 rangeSet (Range Set)

Represents a single range in the rangeSets collection. element is intended to facilitate creating a PivotTable report by consolidating SpreadsheetML ranges that have similar categories of data to be summarized. The simplest layout for the data source is for each rangeSets of data to be in list-like format, with column labels in the first row, row labels in the first column, the rest of the rows having similar items in the same row and column, and no blank rows or columns within the range. A particular rangeSet can consist of a built-in named range that is provided by the application, a user defined named range, a range reference, or a reference to an external workbook.

When multiple ranges are consolidated using this functionality, up to 4 custom report filters (also known as page fields) can be created to help filter the PivotTable report, by specifically enabling one or more of the individual ranges to be selected in the report filter. For each custom page field created, a custom label can be specified and assigned to each range participating in the consolidation range, so that the PivotTable can be filtered by one or more of the ranges being summarized.

[*Example*: Consider a workbook with 6 worksheets. On Sheet1 we have:

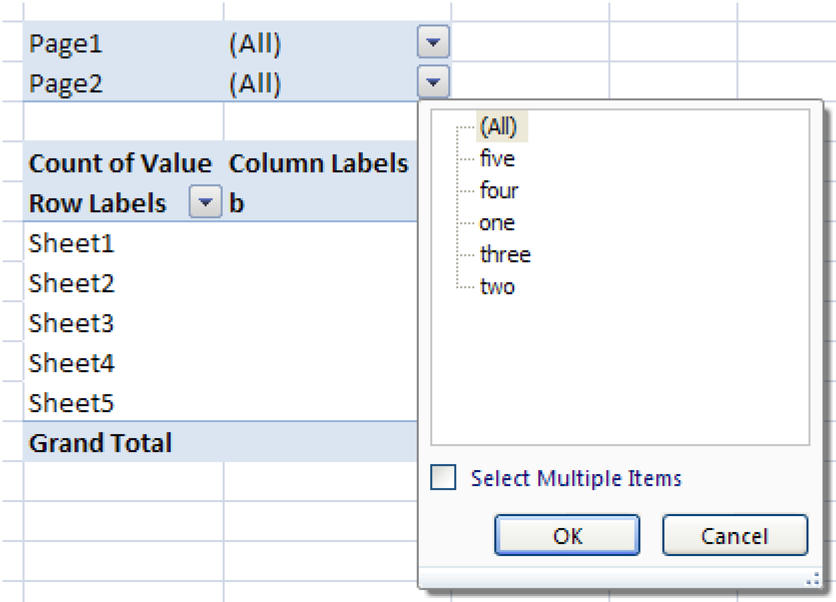
a	b
Sheet1	Sheet1
Sheet1	Sheet1

On Sheet2 we have:

a	b
Sheet2	Sheet2
Sheet2	Sheet2

… and so on up through Sheet5.

On Sheet6, we have the consolidated ranges being summarized by a PivotTable, and two page filters exist for the PivotTable.



Notice that for the second page filter, the items have been assigned a custom label, "one", "two", ..., "five", for each of Sheet1, Sheet2, ..., Sheet5 data sources, respectively. Similarly, the items have been assigned a custom label, "1", "2", ..., "5" for each of Sheet1, Sheet2, ..., Shet5 data sources, respectively.

The XML representing these custom page filters must be like the following:

```
<cacheSource type="consolidation">
  <consolidation autoPage="0">
    <pages count="2">
      <page count="5">
        <pageItem name="1"/>
        <pageItem name="2"/>
        <pageItem name="3"/>
        <pageItem name="4"/>
        <pageItem name="5"/>
      </page>
      <page count="5">
        <pageItem name="one"/>
        <pageItem name="two"/>
        <pageItem name="three"/>
        <pageItem name="four"/>
        <pageItem name="five"/>
      </page>
    </pages>
```



```
      <rangeSets count="5">
        <rangeSet i1="0" i2="0" ref="A1:B3" sheet="Sheet1"/>
        <rangeSet i1="1" i2="1" ref="A1:B3" sheet="Sheet2"/>
        <rangeSet i1="2" i2="2" ref="A1:B3" sheet="Sheet3"/>
        <rangeSet i1="3" i2="3" ref="A1:B3" sheet="Sheet4"/>
        <rangeSet i1="4" i2="4" ref="A1:B3" sheet="Sheet5"/>
      </rangeSets>
    </consolidation>
  </cacheSource>
```

end example]

[*Note*: Attributes i1, i2, i3, and i4 correspond to custom page fields created in the user interface. Spreadsheet ML only supports 4 custom page fields. *end note*]

Attributes	Description
i1 (Field Item Index Page 1)	Specifies the index of a page field item in page filter one. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
i2 (Field Item Index Page 2)	Specifies the index of a page field item in page filter two. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
i3 (Field Item index Page 3)	Specifies the index of a page field item in page filter three. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
i4 (Field Item Index Page 4)	Specifies the index of a page field item in page filter four. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the unique identifier of the Workbook part where the range set is stored. See Workbook (§18.2) for more information. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
name (Named Range)	Specifies the named range. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
ref (Reference)	Specifies the cell range.

Attributes	Description
	The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
sheet (Sheet Name)	Specifies the sheet name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_RangeSet) is located in §A.2. *end note*]

18.10.1.80 rangeSets (Range Sets)

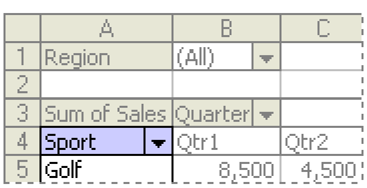
Represents the collection of reference-page items pairs.

Attributes	Description
count (Reference and Page Item Count)	Specifies the number of reference and page items. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RangeSets) is located in §A.2. *end note*]

18.10.1.81 rowFields (Row Fields)

Represents the collection of row fields for the PivotTable.



In the image above, the blue field is a row field. A PivotTable report that has more than one row field has one inner row field (Sport, in the example below), the one closest to the data area. Any other row fields are outer row fields (Region, in the example below). Items in the outermost row field are displayed only once, but items in the rest of the row fields are repeated as needed.

[*Example*:

	A	B	C
3	Sum of Sales		Quarter
4	Region	Sport	Qtr1
5	East	Golf	5,000
6		Safari	9,000
7		Tennis	1,500
8	East Total		15,500
9	West	Golf	3,500

In the image above, Region is an outer row field. Sport is an inner row field.

```
<rowFields count="2">
  <field x="7"/>
  <field x="8"/>
</rowFields>
```

end example]

Attributes	Description
count (Repeated Items Count)	Specifies the number of repeated items in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RowFields) is located in §A.2. *end note*]

18.10.1.82 rowHierarchiesUsage (Row OLAP Hierarchy References)

Represents the collection of references to OLAP hierarchies on the row axis of a PivotTable.

[*Example*:

```
<sh:rowHierarchiesUsage count="1">
  <sh:rowHierarchyUsage hierarchyUsage="9"/>
</sh:rowHierarchiesUsage>
```

end example]

Attributes	Description
count (Item Count)	Specifies the number of items in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RowHierarchiesUsage) is located in §A.2. *end note*]

18.10.1.83 rowHierarchyUsage (Row OLAP Hierarchies)

Represents a references to an OLAP Hierarchy on the row axis of a PivotTable.

[*Example*:

```
<sh:rowHierarchyUsage hierarchyUsage="9"/>
```

end example]

Attributes	Description
hierarchyUsage (Hierarchy Usage)	Specifies the reference to an OLAP hierarchy in a PivotTable. The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_HierarchyUsage) is located in §A.2. *end note*]

18.10.1.84 rowItems (Row Items)

Represents the collection of items in row axis of the PivotTable.

[*Example*: In the SpreadsheetML example below, the item values are found in cells B10:B13. For example "Bikes" is in B10, and corresponds to the first <i> element below.

```
<rowItems count="4">
  <i>
    <x/>
  </i>
  <i r="1">
    <x/>
  </i>
  <i r="1">
    <x v="1"/>
  </i>
  <i t="grand">
    <x/>
  </i>
</rowItems>
```

Looking at the layout of the PivotTable in this example, "Bikes" is the first (and only) item value in the first row, in cell B10. In the XML defining the PivotTable row item values, the first <i> element corresponds to the first row. There is a single index element <x>. The first (and only) <x> element corresponds to the first field on the row axis, namely "Product Category", and an index value of "0" indicates that the 0th item in the items collection for that pivotField definition is how to obtain the item value. Note that "Bikes" isn't explicitly listed as a value here, but instead the 0th item is an index to this field's shared items collection in the pivotCacheDefinition part.

For the second row there are two item values, one item value (Bikes) from the first field in that row (Product Category) and one item value (Mountain Bikes) from the second field in that row (Product Subcategory). In the PivotTable, the first item value "Bikes" is hidden from view. In the XML for this example, the second `<i>` element expresses both item values for this row. The first item value "Bikes" is expressed implicitly, because the value of `@r` on the second `<i>` element is '1', indicating that the first item value from the previous row is reused again as the first item value for the current row. The second item value is expressed explicitly via the `<x>` element under the second `<i>` element. The index of '0' indicates that the 0th item in the `<pivotField>` element for that field is how to obtain the item value. Note again that the 0th item is itself an index into this field's shared items collection in the pivotCacheDefinition part.

The item values for the third row can be discovered in a similar way. *end example*]

Attributes	Description
count (Items in a Row Count)	Specifies the number of items in the row axis of the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_rowItems) is located in §A.2. *end note*]

18.10.1.85 s (Character Value)

Represents a character value in a PivotTable.

[*Example*:

```
<sharedItems count="2">
  <s v="7527 Brook Way"/>
  <s v="3310 Harvey Way"/>
</sharedItems>
```

end example]

Attributes	Description
b (Bold)	Specifies a boolean value that indicates whether this value contains bold formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of 1 or true indicates this value contains bold formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
bc (Background Color)	Specifies the background color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
c (Item Caption)	Specifies the caption for the this item. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
cp (Member Property Count)	Specifies the number of member property values. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
f (Calculated Item)	Specifies a boolean value that indicates whether this is a calculated item value. A value of 1 or true indicates this item is a calculated value. A value of 0 or false indicates this item is not a calculated value. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
fc (Foreground Color)	Specifies the foreground color for this value that was provided by the OLAP server. This attribute applies to OLAP-based PivotTables only. The color is specified as a HEX value in RGB space. The possible values for this attribute are defined by the ST_UnsignedIntHex simple type (§18.18.86).
i (Italic)	Specifies a boolean value that indicates whether the value contains italic formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of 1 or true indicates this value contains italic formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
in (Format Index)	Specifies the index to the OLAP serverformat element where the format string for this entry is stored. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
st (Strikethrough)	Specifies a boolean value that indicates whether the value contains strikethrough formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of 1 or true indicates this value contains strikethrough formatting on the server.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
u (Unused Item)	Specifies a boolean value that indicates whether this is an unused item. The application marks an item as unused when an item is deleted from the data source. The item and associated metadata are retained in the cache until the threshold for unused items specified in missingItemsLimit is reached. A value of `1` or `true` indicates this item is unused. A value of `0` or `false` indicates this item is used. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
un (Underline)	Specifies a boolean value that indicates whether the value contains underline formatting on the OLAP server. This attribute applies to OLAP-based PivotTables only. A value of `1` or `true` indicates this value contains underline formatting on the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
v (Value)	Specifies the value of the item. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_String) is located in §A.2. *end note*]

18.10.1.86 serverFormat (Server Format)

Represents the numeric format specified by the OLAP server for a tuple.

Attributes	Description
culture (Culture)	Specifies a language used to determine the currency symbol to display for currency values. [*Example*: if the culture is "en-us", the values in the application will format the values with a dollar sign. If the culture is "fr-fr" the application will format the values with a euro sign. *end example*] This value conforms to the language tagging conventions of RFC 3066 and later. The pattern <language>-<REGION> is used, e.g., "en-us" or "fr-fr". The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
format (Format)	Specifies the format string to use for all other numeric values. This string is supplied by

Attributes	Description
	the OLAP server. Therefore, the syntax for reading the format string depends on the server implementation. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ServerFormat) is located in §A.2. *end note*]

18.10.1.87 serverFormats (Server Formats)

Represents the collection of numeric and currency formats specified by the OLAP server for a tuple

Attributes	Description
count (Format Count)	Specifies the number of formats in the collection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ServerFormats) is located in §A.2. *end note*]

18.10.1.88 set (OLAP Set)

Represents an OLAP sheet data set or tuple set. The set is defined by a Multidimensional Expressions (MDX) query that specifies criteria for the dimension members that belong to the set.

[*Example*: the following MDX expression defines the set for the 10 salespersons with the lowest sales:

BottomCount([Salesperson].[Salesperson Name].Members,10,[Measures].[Sales])

end example]

The MDX expression is specified in the setDefinition attribute.

Attributes	Description
count (Number of Tuples)	Specifies the number of tuples in the set. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
maxRank (Maximum Rank Requested)	Specifies the largest rank entry the user has requested. The possible values for this attribute are defined by the W3C XML Schema int datatype.

Attributes	Description
queryFailed (Query Failed)	Specifies a boolean value that indicates whether querying on this set failed. A value of `1` or `true` indicates a query against this set failed. A value of `0` or `false` indicates a query against this set succeeded. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
setDefinition (MDX Set Definition)	Specifies the Multidimensional Expressions (MDX) set definition. [*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn2.microsoft.com/en-us/library/ms145595.aspx *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
sortType (Set Sort Order)	Specifies the sort order of the set. The possible values for this attribute are defined by the ST_SortType simple type (§18.18.74).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Set) is located in §A.2. *end note*]

18.10.1.89 sets (Sets)

Represents the collection of OLAP sheet data entries or tuple sets.

Attributes	Description
count (Tuple Set Count)	Specifies the number of tuple sets. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Sets) is located in §A.2. *end note*]

18.10.1.90 sharedItems (Shared Items)

Represents the collection of unique items for a field in the PivotCacheDefinition. The sharedItems complex type stores data type and formatting information about the data in a field. Items in the PivotCacheDefinition can be shared in order to reduce the redundancy of those values that are referenced in multiple places across all the PivotTable parts. [*Example*: A value might be part of a filter, it might appear on a row or column axis, and will appear in the pivotCacheRecords definition as well. However, because of the performance cost of creating the optimized shared items, items are only shared if they are actually in use in the PivotTable. Therefore, depending on user actions on the PivotTable layout, the pivotCacheDefinition and underlying PivotCacheRecords part can be updated. *end example*]

If there are no shared items, then field values are stored directly in the pivotCacheRecords part.

[*Example*:

```
<sharedItems count="1">
  <s v="[Customer].[Customer Geography].[Country].&amp;[United States]"
    c="United States"/>
</sharedItems>
```

end example]

The following attributes are not required or used if there are no items in sharedItems.

- containsBlank
- containsSemiMixedTypes
- containsMixedTypes
- longText

The following attributes are not used unless there is more than one item in sharedItems or the one and only item is not a blank item. If the first item is a blank item the data type the field cannot be verified.

- containsNumber
- containsDates
- containsString
- containsInteger

The following attributes can be omitted without loss of functionality.

- containsNonDate
- count

The following attributes are not required and can be omitted. However, refreshing the PivotTable could produce different groupings than before.

- maxDate
- minDate
- maxValue
- minValue

Applications should ensure that “date” attributes are not mixed with “value” attributes.

Attributes	Description
containsBlank	Specifies a boolean value that indicates whether this field contains a blank value.

Attributes	Description
(Contains Blank)	A value of `1` or `true` indicates this field contains one or more blank values. A value of `0` or `false` indicates this field does not contain blank values. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
containsDate (Contains Date)	Specifies a boolean value that indicates that the field contains at least one date. A value of `1` or `true` indicates the field contains at least one date value. A value of `0` or `false` indicates the field does not contain any date values. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
containsInteger (Contains Integer)	Specifies a boolean value that indicates whether this field contains integer values. A value of `1` or `true` indicates this field contains integer values. A value of `0` or `false` indicates non-integer or mixed values. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
containsMixedTypes (Contains Mixed Data Types)	Specifies a boolean value that indicates whether this field contains more than one data type. A value of `1` or `true` indicates this field contains more than one data type. A value of `0` or `false` indicates contains only one data type. The field can still contain blank values. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
containsNonDate (Contains Non Date)	Specifies a boolean value that indicates that the field contains at least one value that is not a date. A value of `1` or `true` indicates the field contains at least one non-date values. A value of `0` or `false` indicates this field contains no date fields. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
containsNumber (Contains Numbers)	Specifies a boolean value that indicates whether this field contains numeric values.

Attributes	Description
	A value of 1 or true indicates this field contains at least one numeric value. A value of 0 or false indicates this field contains no numeric values. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
containsSemiMixedTypes (Contains Semi Mixed Data Types)	Specifies a boolean value that indicates that this field contains text values. The field can also contain a mix of other data type and blank values. A value of 1 or true indicates at least one text value, and can also contain a mix of other data types and blank values. A value of 0 or false indicates the field does not have a mix of text and other values. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
containsString (Contains String)	Specifies a boolean value that indicates whether this field contains a text value. A value of 1 or true indicates this field contains at least one text value. A value of 0 or false indicates this field does not contain any text values. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
count (Shared Items Count)	Specifies the number of shared items to load for this field. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
longText (Long Text)	Specifies a boolean value that indicates whether this field contains a long text value. A string is considered long if it is over 255 Unicode scalar values. A value of 1 or true indicates the value contains more than 255 Unicode scalar valuesof text. A value of 0 or false indicates the value contains less than 255 Unicode scalar values. [*Note*: This is used as many legacy spreadsheet application support a limit of 255 characters for text values. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
maxDate (Maximum Date Time Value)	Specifies the maximum date/time value found in a date field. The possible values for this attribute are defined by the W3C XML Schema dateTime

Attributes	Description
	datatype.
maxValue (Maximum Numeric Value)	Specifies the maximum numeric value found in a numeric field. The possible values for this attribute are defined by the W3C XML Schema double datatype.
minDate (Minimum Date Time)	Specifies the minimum date/time value found in a date field. The possible values for this attribute are defined by the W3C XML Schema dateTime datatype.
minValue (Minimum Numeric Value)	Specifies the minimum numeric value found in a numeric field. The possible values for this attribute are defined by the W3C XML Schema double datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SharedItems) is located in §A.2. *end note*]

18.10.1.91 sortByTuple (Sort By Tuple)

Represents the sort applied to a tuple.

Attributes	Description
c (Member Name Count)	Specifies the number of member names. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Tuples) is located in §A.2. *end note*]

18.10.1.92 tpl (Tuple)

Represents an OLAP sheet data entry member.

Attributes	Description
fld (Field Index)	Specifies the index of the field to which the member belongs. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
hier (Hierarchy Index)	Specified the index of the hierarchy to which the member belongs. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

Attributes	Description
item (Item Index)	Specifies the index of the item in the field that represents this item. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Tuple) is located in §A.2. *end note*]

18.10.1.93 tpls (Tuples)

Represents members for the OLAP sheet data entry, also known as a tuple.

Attributes	Description
c (Member Name Count)	Specifies the number of member names. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Tuples) is located in §A.2. *end note*]

18.10.1.94 tupleCache (Tuple Cache)

Represents the cache of OLAP sheet data members, or tuples.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TupleCache) is located in §A.2. *end note*]

18.10.1.95 worksheetSource (Worksheet PivotCache Source)

Represents the location of the source of the data that is stored in the cache.

[*Example*:

```
<cacheSource type="worksheet">
  <worksheetSource name="Table1" r:id="rId2"/>
</cacheSource>
```

end example]

Attributes	Description
id (Relationship Id) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationshi	Specifies the identifier to the Sheet part whose data is stored in the cache. See the Sheet section (§18.2) for more information. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).

Attributes	Description
ps	
name (Named Range)	Specifies the named range that is the source of the data. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
ref (Reference)	Specifies the reference that defines a cell range that is the source of the data. This attribute depends on how the application implements cell references. The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
sheet (Sheet Name)	Specifies the name of the sheet that is the source for the cached data. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_WorksheetSource) is located in §A.2. *end note*]

18.10.1.96 x (Member Property Index)

Represents an array of indexes to cached member property values.

Attributes	Description
v (Shared Items Index)	Specifies the index into the shared items table in the PivotCache that identifies this item. The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_X) is located in §A.2. *end note*]

18.10.1.97 x (Shared Items Index)

This element represents an array of indexes to cached shared item values

Attributes	Description
v (Shared Items Index)	Specifies the index into the shared items table in the PivotCache that identifies this item. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Index) is located in §A.2. *end note*]

18.10.2 Shared Pivot Table Data

This section defines the part where shared PivotTable data is stored.

18.10.2.1 reference (Reference)

Represents a set of selected fields and selected items within those fields.

[*Example*:

```
<sh:reference field="4294967294" count="1" selected="0">
  <sh:x v="0"/>
</sh:reference>
```

end example]

Attributes	Description
avgSubtotal (Include Average Filter)	Specifies a boolean value that indicates whether the 'average' aggregate function is included in the filter. A value of `1` or `true` indicates the average aggregation function is included in the filter. A value of `0` or `false` indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
byPosition (Positional Reference)	Specifies a boolean value that indicates whether the item is referred to by position rather than item index. A value of `1` or `true` indicates the item is referred to by position. A value of `0` or `false` indicates the item is referred to by index. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
count (Item Index Count)	Specifies the number of item indexes in the collection of indexes (x tags). The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
countASubtotal (Include CountA Filter)	Specifies a boolean value that indicates whether the 'countA' subtotal is included in the filter. A value of `1` or `true` indicates the count aggregation function is included in the filter. A value of `0` or `false` indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
countSubtotal (Include Count	Specifies a boolean value that indicates whether the count aggregate function is included in the filter.

Attributes	Description
Subtotal)	A value of 1 or true indicates the count aggregation function is included in the filter. A value of 0 or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
defaultSubtotal (Include Default Filter)	Specifies a boolean value that indicates whether the default subtotal is included in the filter. A value of 1 or true indicates the default subtotal is included in the filter. The default is to display the total or the grand total. A value of 0 or false indicates another subtotal or aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
field (Field Index)	Specifies the index of the field to which this filter refers. A value of -2 indicates the 'data' field. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
maxSubtotal (Include Maximum Filter)	Specifies a boolean value that indicates whether the 'maximum' aggregate function is included in the filter. A value of 1 or true indicates the maximum aggregation function is included in the filter. A value of 0 or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
minSubtotal (Include Minimum Filter)	Specifies a boolean value that indicates whether the 'minimum' aggregate function is included in the filter. A value of 1 or true indicates the minimum aggregation function is included in the filter. A value of 0 or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
productSubtotal (Include Product Filter)	Specifies a boolean value that indicates whether the 'product' aggregate function is included in the filter.

Attributes	Description
	A value of 1 or true indicates the product aggregation function is included in the filter. A value of 0 or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
relative (Relative Reference)	Specifies a boolean value that indicates whether the item is referred to by a relative reference rather than an absolute reference. This attribute is used if posRef is set to true. A value of 1 or true indicates the item is referred to by a relative reference. A value of 0 or false indicates the item is referred to by an absolute reference. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
selected (Selected)	Specifies a boolean value that indicates whether this field has selection. This attribute is used when the PivotTable is in Outline view. It is also used when both header and data cells have selection. A value of 1 or true indicates the field has selection. A value of 0 or false indicates the field does not have selection. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
stdDevPSubtotal (Include StdDevP Filter)	Specifies a boolean value that indicates whether the population standard deviation aggregate function is included in the filter. A value of 1 or true indicates the population standard deviation aggregation function is included in the filter. A value of 0 or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
stdDevSubtotal (Include StdDev Filter)	Specifies a boolean value that indicates whether the standard deviation aggregate function is included in the filter. A value of 1 or true indicates the standard deviation aggregation function is included in the filter. A value of 0 or false indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
	datatype.
sumSubtotal (Include Sum Filter)	Specifies a boolean value that indicates whether the sum aggregate function is included in the filter. A value of `1` or `true` indicates the sum aggregation function is included in the filter. A value of `0` or `false` indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
varPSubtotal (Include VarP Filter)	Specifies a boolean value that indicates whether the population variance aggregate function is included in the filter. A value of `1` or `true` indicates the population variance aggregation function is included in the filter. A value of `0` or `false` indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
varSubtotal (Include Var Filter)	Specifies a boolean value that indicates whether the variance aggregate function is included in the filter. A value of `1` or `true` indicates the variance aggregation function is included in the filter. A value of `0` or `false` indicates another aggregation function is included in the filter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotAreaReference) is located in §A.2. *end note*]

18.10.2.2 references (References)

Represents the set of selected fields and the selected items within those fields.

[*Example*:

```
<sh:references count="5">
  <sh:reference field="4294967294" count="1" selected="0">
    <sh:x v="0"/>
  </sh:reference>
```



```
    <sh:reference field="2" count="1" selected="0">
      <sh:x v="0"/>
    </sh:reference>
    <sh:reference field="14" count="1" selected="0">
      <sh:x v="0"/>
    </sh:reference>
    <sh:reference field="15" count="2" selected="0">
      <sh:x v="2"/>
      <sh:x v="3"/>
    </sh:reference>
  </sh:references>
```

end example]

Attributes	Description
count (Pivot Filter Count)	Specifies the number of filtered records available in the PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PivotAreaReferences) is located in §A.2. *end note*]

18.11 Shared Workbook Data

The Shared Workbooks architecture enables a spreadsheet application to record revisions made to a workbook (e.g., track changes), and is designed to enable either single, or multiple users editing the same workbook at the same time. Therefore, the application needs to support the ability to read changes made by another user and update its own state of the same workbook with those changes, even when those changes are made concurrently with other changes made by other users. Inevitably there are conflicts, and therefore merge conflict resolution should be supported the runtime application. The file format only contains enough information so that the spreadsheet application can deal with conflicts, and can undo/redo changes from the change history at run time.

18.11.1 Shared Workbook Data

Within a shared workbook, the changes made to the spreadsheet at runtime are persisted as sets of different revisions collectively forming a revision history. These are persisted to the file on disk during a save event, and are saved in different xml parts known as revision logs. There is a headers table xml part that summarizes when changes were made, who made them, and it lists the relationship from each header to the individual revision log that records the specific changes.

[*Example*: This example shows the header and revision log for two simple events: adding text to a cell, and inserting a new sheet.

First, take a look at the header table, and revision log:

```
<headers xmlns="…" xmlns:r="…" guid="{A84A6777-8908-4CB9-9EB6-625CEFF419D3}">
  <header guid="{A84A6777-8908-4CB9-9EB6-625CEFF419D3}"
    dateTime="2006-07-14T13:42:54" maxSheetId="4" userName="UserName"
    r:id="rId1">
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
</headers>
```

And the revision log is essentially empty:

```
<revisions xmlns="…" xmlns:r="…"/>
```

Now, after inserting the text "foo" into cell A1, and saving, the header looks like this:

```
<headers xmlns="…" xmlns:r="…" guid="{CFEA9B63-728B-4274-A346-0440E1573AB4}"
  diskRevisions="1" revisionId="1" version="2">
  <header guid="{A84A6777-8908-4CB9-9EB6-625CEFF419D3}"
    dateTime="2006-07-14T13:42:54" maxSheetId="4" userName="UserName"
    r:id="rId1">
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
  <header guid="{CFEA9B63-728B-4274-A346-0440E1573AB4}"
    dateTime="2006-07-14T13:44:40" maxSheetId="4" userName="UserName"
    r:id="rId2" minRId="1">
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
</headers>
```

A new header entry is added, with a GUID and a revision ID (rId2) that specifies which log to look into to see the details about the revision.

The old log is saved, and the newly created log (corresponding to rId2) now looks like this:

```
<revisions xmlns="…" xmlns:r="…">
  <rcc rId="1" sId="1">
    <nc r="A1" t="inlineStr">
      <is>
        <t>foo</t>
        <phoneticPr fontId="0"/>
      </is>
    </nc>
  </rcc>
</revisions>
```

The log shows that the contents of a cell were revised, and the new cell contents is text containg "foo" as the string.

After inserting a new sheet, the header looks like this:

```
<headers xmlns="…" xmlns:r="…" guid="{7E1DAFA8-EF95-4865-8FE8-CC17B28635CF}"
  diskRevisions="1" revisionId="2" version="3">
  <header guid="{A84A6777-8908-4CB9-9EB6-625CEFF419D3}"
    dateTime="2006-07-14T13:42:54" maxSheetId="4"
    userName="UserName" r:id="rId1">
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
  <header guid="{CFEA9B63-728B-4274-A346-0440E1573AB4}"
    dateTime="2006-07-14T13:44:40" maxSheetId="4" userName="UserName"
    r:id="rId2" minRId="1">
    <sheetIdMap count="3">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
    </sheetIdMap>
  </header>
  <header guid="{7E1DAFA8-EF95-4865-8FE8-CC17B28635CF}"
    dateTime="2006-07-14T13:48:56" maxSheetId="5" userName="UserName"
    r:id="rId3" minRId="2">
```



```
    <sheetIdMap count="4">
      <sheetId val="1"/>
      <sheetId val="2"/>
      <sheetId val="3"/>
      <sheetId val="4"/>
    </sheetIdMap>
  </header>
</headers>
```

You can see that the last, most recent, header entry shows an entry for the new sheet. The most recent log looks like this:

```
<revisions xmlns="…" xmlns:r="…">
  <ris rId="2" sheetId="4" name="[shared example.xlsx]Sheet4"
    sheetPosition="3"/>
  <rcv guid="{841DBE00-ECD0-478E-893B-30CE5DABBEF5}" action="delete"/>
  <rcv guid="{841DBE00-ECD0-478E-893B-30CE5DABBEF5}" action="add"/>
</revisions>
```

This shows the new sheet, sheetId 4, is added to the workbook. The custom view (rcv) for the user is updated as a new sheet was added.

end example]

18.11.1.1 header (Header)

This element is essentially a table that contains metadata about a list of specific changes that have taken place for this workbook. It lists when the changes were made, who made them, and the relationship IDs so that the log detailing the specific change can be found. If tracking changes, or sharing workbooks, are enabled, then changes are persisted on the Save event, or at a specified time interval. A header is created for each set of changes.

Attributes	Description
dateTime (Date Time)	The date and time when this set of revisions was saved. [*Note*: This can happen when the user explicitly saves, or the save can occur due to a time interval, specified in the spreadsheet application, elapsing. *end note*] The possible values for this attribute are defined by the W3C XML Schema dateTime datatype.
guid (GUID)	A globally unique identifier for this set of revisions. The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).
id (Relationship ID)	This is the ID that is used to find the corresponding log record of the changes made for this header.

Attributes	Description
Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Use the corresponding relationship expressed in the revisionHeaders part to locate the log record that lists the specific changes. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
maxRId (Max Revision Id)	The highest revision Id that belongs to this header. [*Note*: This can be used when, given a revision ID, the spreadsheet application needs to determine which revision log to access. *end note*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
maxSheetId (Last Sheet Id)	Internal identifier of the next available sheet in this workbook. The numbering here is the index of the next available sheet in the workbook in a 1-based index system. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
minRId (Minimum Revision Id)	The lowest revision id that belongs to this header. [*Note*: this can be used when, given a revision ID, the spreadsheet application needs to determine which revision log to access. *end note*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
userName (User Name)	A string representing the name of the user making the revision.. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionHeader) is located in §A.2. *end note*]

18.11.1.2 headers (Revision Headers)

This element represents the list of revision headers.

This section contains many references to history, versions, and revisions, and it is helpful to clarify the relationships here. In general, a series of changes (revisions) can be made to a spreadsheet. When a batch of those revisions is saved to disk, the version number of the spreadsheet is incremented. The batch of changes is saved to the revision history, which is persisted on disk with the file in the form of different log files and headers.

There are some attributes that deal with history which might seem redundant (such as `diskRevisions`, and `history`, among others) - these are there for backwards compatibility with older versions of spreadsheet applications and do not need to be used for creating new files.

Attributes	Description
diskRevisions (Disk Revisions)	A Boolean value indicating that this shared workbook file contains revisions. `True` when the workbook does have revisions, `false` otherwise. [*Note*: this attribute is used for backwards compatibility. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
exclusive (Exclusive Mode)	A Boolean value indicating that this shared workbook is in exclusive mode. A workbook is in exclusive mode when a user has a lock on it for appending revisions to the file. [*Note*: This is used for backwards compatibility with older spreadsheet applications. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
guid (Last Revision GUID)	The globally unique identifier of the last set of revisions. This shall match the GUID for the most recent header. The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).
history (History)	A Boolean value indicating that this shared workbook maintains a revision history. True if a history is maintained, false otherwise. [*Note*: This is used for backwards compatibility with older spreadsheet applications. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
keepChangeHistory (Keep Change History)	A Boolean value indicating whether the revision history should be kept for this shared workbook. True if the history should be kept, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
lastGuid (Last GUID)	Unique identifier of the last set of revisions that was saved into the file. The spreadsheet application can have certain modes, such as a timed save mode, where the application doesn't do a full save, but instead just appends the most recent revision records. In cases like this, for a new user that opens such a file while it is being edited, the file that was loaded from disk only have the changes that were saved during a full

Attributes	Description
	save. To get the current state of the file which includes edits by other users, the spreadsheet application would need to apply all the revisions from `lastGuid` to `guid`. The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).
preserveHistory (Preserve History)	An integer representing the number of days the spreadsheet application shall keep the change history for this workbook. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
protected (Protected)	A Boolean value indicating whether the change tracking in this shared workbook can be removed. True if the tracking can be removed, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
revisionId (Revision Id)	The current revision number of this shared workbook. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
shared (Shared Workbook)	A Boolean value indicating that this workbook is shared. True when the workbook is shared, false otherwise. [*Note*: This is used for backwards compatibility with older spreadsheet applications. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
trackRevisions (Track Revisions)	A Boolean value indicating that revisions are tracked in this shared workbook. True when revisions are tracked, false otherwise. [*Note*: This is used for backwards compatibility with older spreadsheet applications. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
version (Version)	An integer representing the current version of this shared workbook. The integer should begin counting from 1 for the first version. The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionHeaders) is located in §A.2. *end note*]

18.11.1.3 nc (New Cell Data)

This element represents new cell data that was added to the worksheet.

For most spreadsheet application purposes, only the data type and reference need to be used for revision tracking purposes. The rest of the cell properties can be written out, but are not necessarily needed as they can be recorded in other areas of the spreadsheet. For instance the <rfmt> element can be used to record style information instead of the `S (style index)` attribute.

Attributes	Description
cm (Cell Metadata Index)	The zero-based index of the cell metadata record associated with this cell. Metadata information is found in the Metadata Part. Cell metadata is extra information stored at the cell level, and is attached to the cell (travels through moves, copy / paste, clear, etc). Cell metadata is not accessible via formula reference. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ph (Show Phonetic)	A Boolean value indicating if the spreadsheet application should show phonetic information. Phonetic information is displayed in the same cell across the top of the cell and serves as a 'hint' which indicates how the text should be pronounced. This should only be used for East Asian languages. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
r (Reference)	An A1 style reference to the location of this cell The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
s (Style Index)	The index of this cell's style. Style records are stored in the Styles Part. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
t (Cell Data Type)	An enumeration representing the cell's data type. The possible values for this attribute are defined by the ST_CellType simple type (§18.18.11).
vm (Value Metadata Index)	The zero-based index of the value metadata record associated with this cell's value. Metadata records are stored in the Metadata Part. Value metadata is extra information stored at the cell level, but associated with the value rather than the cell itself. Value metadata is accessible via formula reference. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Cell) is located in §A.2. *end note*]

18.11.1.4 ndxf (New Formatting Information)

This element represents new differential formatting information for this cell. This formatting is applied to the existing formatting of the cell.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Dxf) is located in §A.2. *end note*]

18.11.1.5 oc (Old Cell Data)

This element represents old cell data. Old cell data is data that was previously stored in the cell.

For most spreadsheet application purposes, only the data type and reference need to be used for revision tracking purposes. The rest of the cell properties can be written out, but are not necessarily needed as they can be recorded in other areas of the spreadsheet. For instance the <rfmt> element can be used to record style information instead of the `S (style index)` attribute.

Attributes	Description
cm (Cell Metadata Index)	The zero-based index of the cell metadata record associated with this cell. Metadata information is found in the Metadata Part. Cell metadata is extra information stored at the cell level, and is attached to the cell (travels through moves, copy / paste, clear, etc). Cell metadata is not accessible via formula reference. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ph (Show Phonetic)	A Boolean value indicating if the spreadsheet application should show phonetic information. Phonetic information is displayed in the same cell across the top of the cell and serves as a 'hint' which indicates how the text should be pronounced. This should only be used for East Asian languages. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
r (Reference)	An A1 style reference to the location of this cell The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
s (Style Index)	The index of this cell's style. Style records are stored in the Styles Part. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
t (Cell Data Type)	An enumeration representing the cell's data type. The possible values for this attribute are defined by the ST_CellType simple type (§18.18.11).
vm (Value Metadata Index)	The zero-based index of the value metadata record associated with this cell's value. Metadata records are stored in the Metadata Part. Value metadata is extra information stored at the cell level, but associated with the value rather than the cell itself. Value

Attributes	Description
	metadata is accessible via formula reference. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Cell) is located in §A.2. *end note*]

18.11.1.6 odxf (Old Formatting Information)

This element represents the old differential formatting information for this cell. Old differential formatting is differential formatting that was previously applied to the cell.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Dxf) is located in §A.2. *end note*]

18.11.1.7 oldFormula (Old Formula)

This element represents the old formula for a defined name in this cell. This is only used for named cells. Formulas that are entered in a cell with no name are represented by the formula element `<f>`.

The possible values for this element are defined by the ST_Formula simple type (§18.18.35).

[*Note*: The W3C XML Schema definition of this element's content model (ST_Formula) is located in §A.2. *end note*]

18.11.1.8 raf (Revision AutoFormat)

This element represents a revision record of auto formatting change information for a table.

Attributes	Description
applyAlignmentFormats (Apply Alignment Formats)	If `true` apply legacy table autoformat alignment properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyBorderFormats (Apply Border Formats)	If `true` apply legacy table autoformat border properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyFontFormats (Apply Font Formats)	If `true` apply legacy table autoformat font properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyNumberFormats (Apply Number Formats)	If `true` apply legacy table autoformat number format properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
applyPatternFormats (Apply Pattern Formats)	If `true` apply legacy table autoformat pattern properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyWidthHeightFormats (Apply Width / Height Formats)	If `true` apply legacy table autoformat width/height properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoFormatId (Auto Format Id)	Identifies which legacy table autoformat to apply. Annex D contains a listing of the supported PivotTable AutoFormats, example formatting, and a sample workbook with each of those AutoFormats applied. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ref (Reference)	A-1 style reference to the location where the formatting was applied The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionAutoFormatting) is located in §A.2. *end note*]

18.11.1.9 rcc (Revision Cell Change)

This element stores information about the contents of the cell that was replaced.

Attributes	Description
dxf (Formatting)	A Boolean flag indicating that there was a differential formatting change for this cell - true if there was a formatting change, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
endOfListFormulaUpdate (End of List Formula Update)	A Boolean flag indicating that indicates that the formula used at the end of a list has been updated. True if the formula was updated, false otherwise. List in this context does not mean table, rather it refers to the feature where the

Attributes	Description
	spreadsheet application automatically creates an internal structure for making data input more consistent on adjacent rows or columns. For instance, if 3 cells in a row are entered with the same format, then when entering data into the 4th adjacent cell, the spreadsheet application might automatically apply that same format. In this case, those cells are treated as a list. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
numFmtId (Number Format Id)	Zero-based index of the number format (Fmt) record used by this cell format (XF). The possible values for this attribute are defined by the ST_NumFmtId simple type (§18.18.47).
odxf (Old Formatting)	Flag indicating that there is old formatting information available for this cell. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
oldPh (Old Phonetic Text)	A Boolean flag indicating whether there is old phonetic text information available. True when there is old phonetic text information available, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
oldQuotePrefix (Old Quote Prefix)	A Boolean value indicating if a single quote prefix is was used on this cell previously. Single quote prefixes are used to cause a formula to be evaluated as a string. True if a single quote prefix was used previously, false otherwise The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ph (Phonetic Text)	A Boolean flag indicating whether this cell contains phonetic text or not. True when the cell contains phonetic text, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
quotePrefix (Quote Prefix)	A Boolean value indicating if a single quote prefix is used. Single quote prefixes are used to cause a formula to be evaluated as a string. True if a single quote prefix is used, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
s (Style Revision)	Flag indicating that formatting change for this cell affected the cell's style. (Only applicable for Undo operations) The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sId (Sheet Id)	Internal identifier of the sheet on which the revision occurred. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
xfDxf (Row Column Formatting Change)	Flag indicating that the formatting change had an effect on the formatting of the entire row or column that this cell belongs to. (Only applicable for Undo operations). The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionCellChange) is located in §A.2. *end note*]

18.11.1.10 rcft (Revision Merge Conflict)

This element represents a revision record which indicates that there was a merge conflict.

Attributes	Description
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

Attributes	Description
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionConflict) is located in §A.2. *end note*]

18.11.1.11 rcmt (Revision Cell Comment)

This element represents a revision record of a cell comment change.

Attributes	Description
action (User Action)	An enumeration identifying what kind of an operation the user performed on the comment. The possible values for this attribute are defined by the ST_RevisionAction simple type (§18.18.65).
alwaysShow (Always Show Comment)	A Boolean value indicating that the user has set this comment to always be visible. True if the comment is set to always be visible, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
author (Author)	A string representing the name of the author who changed this comment. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
cell (Cell)	An A-1 style reference to the cell where the comment was changed. The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
guid (GUID)	A globally unique identifier of this comment. The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).
hiddenColumn (Hidden Column)	A Boolean value indicating that the comment belongs to a cell in a hidden column. True if the comment is in a hidden column, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
	datatype.
hiddenRow (Comment In Hidden Row)	A Boolean value indicating that the comment belongs to a cell in a hidden row. True if the comment is in a hidden row, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
newLength (New Comment Length)	Length of the comment text added in this revision. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
old (Old Comment)	An ignorable Boolean value used for backwards compatibility that indicates that the original comment was created by a legacy spreadsheet application. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
oldLength (Original Comment Length)	Length of the comment before this revision was made. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionComment) is located in §A.2. *end note*]

18.11.1.12 rcv (Revision Custom View)

This element represents a revision record of adding or removing a custom view to the workbook

Attributes	Description
action (User Action)	An enumeration representing the action that the user performed. The possible values for this attribute are defined by the ST_RevisionAction simple type (§18.18.65).
guid (GUID)	A globally unique identifier of the custom view. 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_RevisionCustomView) is located in §A.2. *end note*]

18.11.1.13 rdn (Revision Defined Name)

This element represents a revision record of a defined name change.

Attributes	Description
comment (Name Comment)	A string representing a comment about the defined name. This comment can be shown by the spreadsheet application in a names management UI so that users have more information about what the defined name is used for. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
customMenu (New Custom Menu)	A string representing the new custom menu text The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
customView (Custom View)	A Boolean flag indicating that this named range belongs to a custom view The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
description (Description)	A string representing the new description text for the defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
function (Function)	A Boolean value indicating that the defined name refers to a function. True if the defined name is a function, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
functionGroupId (Function Group Id)	Represents the new function group id. Function group ids are used to help classify functions. For instance, functions in the same group can be searched or selected easily from the spreadsheet applications UI. For instance, filtering the list of all functions to allow the user to choose from functions used for financial data. The following group ids should be used: ID Function group 1 Financial 2 Date and Time 3 Math and Trig 4 Statistical 5 Lookup & Reference 6 Database 7 Text 8 Logical

Attributes	Description
	9 Information 10 Commands 11 Customizing 12 Macro Control 13 DDE/External 14 User Defined 15 Engineering 14 Cube The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
help (New Help Topic)	A string representing the new help topic text. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
hidden (Named Range Hidden)	A Boolean value indicating whether the named range is now hidden. Hidden refers to whether the defined name is of a 'hidden' type. This applies to things like a custom filter on a cell, it has a name, but is hidden and so is not visible in any name management UI. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
localSheetId (Local Name Sheet Id)	An integer representing the id of the sheet to which this defined name belongs. This shall be used local defined names only. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (Name)	A string representing the name for this defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
oldComment (Old Name Comment)	A string representing the old comment about the defined name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
oldCustomMenu (Old Custom Menu Text)	A string representing the old custom menu text The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
oldDescription (Old Description)	A string representing the old description text The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

Attributes	Description
oldFunction (Old Function)	A Boolean flag indicating that the old name was a function The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
oldFunctionGroupId (Old Function Group Id)	Old function group ID. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
oldHelp (Old Help Topic)	A string representing the old help topic text The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
oldHidden (Old Hidden)	A Boolean flag indicating whether the named range was hidden Hidden refers to whether the defined name is of a 'hidden' type. This applies to things like a custom filter on a cell, it has a name, but is hidden and so is not visible in any name management UI. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
oldShortcutKey (Old Short Cut Key)	Old keyboard shortcut. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
oldStatusBar (Old Status Bar)	A string representing the old status bar text The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
shortcutKey (Shortcut Key)	Represents the new keyboard shortcut. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
statusBar (Status	A string representing the new status bar text.

Attributes	Description
Bar)	The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionDefinedName) is located in §A.2. *end note*]

18.11.1.14 reviewed (Reviewed)

This element represents an identifier of a single reviewed revision. A reviewed revision, is a revision that has been reviewed via the spreadsheet application's track changes feature, has been accepted, and has been saved.

Attributes	Description
rId (revision Id)	ID of a reviewed revision. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Reviewed) is located in §A.2. *end note*]

18.11.1.15 reviewedList (Reviewed List)

This element maintains a list of reviewed revisions.

Attributes	Description
count (Reviewed Revisions Count)	Number of reviewed revisions. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ReviewedRevisions) is located in §A.2. *end note*]

18.11.1.16 revisions (Revisions)

This element represents the root node of a list of revisions made in this shared workbook. This root node shows up at the beginning of every log file that contains specific revisions made to the workbook.

When multiple users are sharing, and editing, a workbook at the same time, there can be conflicting changes. The spreadsheet application should have logic to resolve such conflicts, and the file format should only contain enough information so that the spreadsheet application can restore the workbook to the correct state after conflict resolution. Revisions can also be tracked by the spreadsheet application for review by the user at a later time (as opposed to only dealing with conflicts on a save event.) Some edits to workbooks are made as a result of this conflict resolution. So, there are cases where a revision is effectively undone by another user, and as a result that undoing is itself a revision that adds or changes data in the file. These operations are tracked by the ua and ra attributes of many different elements.

[*Example*:

Step 1:

User 1 inserts Column A. So the XML in the revision log would look like this:

```
<revisions xmlns="…" xmlns:r="…">
  <rrc rId="1" sId="1" ref="A1:A1048576" action="insertCol"/>
</revisions>
```

Step 2:

User 2 synchronizes the file to pick up that change, but then activates the Track Changes feature, and rejects that change. This effectively performs an undo on User 1's insertion. This is denoted in the file with the ua attribute meaning that this change happened as the result of an undo. The XML for the revision log would look like this:

```
<revisions xmlns="…" xmlns:r="…">
  <rrc rId="2" ua="1" sId="1" ref="A1:A1048576" action="deleteCol"/>
  <rcft rId="1" ua="1" sheetId="1"/>
</revisions>
```

Step 3:

User 1 enters "foo" in A1, and saves the file. A conflict resolution dialog is shown since User 2's version of the file removed the inserted Column A. User 1 chooses to accept their own changes. This undoes the change that User 2 made. So, in effect, it performed an undo on a previous undo operation. This is denoted in the file format by the ra attribute meaning that a the change occurred because a previous undo was undone. So the resulting XML for the newest log file looks like this:


```
<revisions xmlns="…" xmlns:r="…">
  <rrc rId="3" ua="1" ra="1" sId="1" ref="A1:A1048576" action="insertCol"/>
  <rcft rId="2" ua="1" sheetId="1"/>
  <rcc rId="4" sId="1">
    <nc r="A1" t="inlineStr">
      <is>
        <t>foo</t>
      </is>
    </nc>
  </rcc>
  <rcft rId="2" sheetId="1"/>
</revisions>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Revisions) is located in §A.2. *end note*]

18.11.1.17 rfmt (Revision Format)

This element represents a revision record of information about a formatting change.

Attributes	Description
length (Length)	The number of characters that were affected by a string change, counting from `start`. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
s (Style)	Flag indicating that this formatting change affected a cell's style. (Only applicable for Undo operations). The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sqref (Sequence Of References)	A worksheet range to which this formatting was applied. [*Note*: For applications supporting the default grid size (see §18.17.5), full column and row references shall explicitly state the row and column components, e.g., "A1:A1048576" For column "A", and A1:XFD1 for row "1". Applications with larger grid sizes shall interpret these to mean "column A" and "row 1" respectively, for their larger grid size. *end note*] The possible values for this attribute are defined by the ST_Sqref simple type (§18.18.76).
start (Start index)	An integer representing an index showing which character a string change starts at within

Attributes	Description
	the string in the cell. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
xfDxf (Row or Column Formatting Change)	A Boolean flag indicating that this formatting change had an affect on the formatting of an entire row or column that an affected cell(s) belongs to. (Only applicable for Undo operations) The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionFormatting) is located in §A.2. *end note*]

18.11.1.18 ris (Revision Insert Sheet)

This element represents a revision record of a sheet that was inserted.

Attributes	Description
name (Sheet Name)	The name of the new sheet. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sheetPosition (Sheet Position)	An integer representing the zero based position of the new sheet in the sheet tab bar. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionInsertSheet) is located in §A.2. *end note*]

18.11.1.19 rm (Revision Cell Move)

This element represents a revision record on a cell(s) that moved.

Attributes	Description
destination (Destination)	New A1 style location of the cell(s) that were moved The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
source (Source)	The original A1 style location of the cell(s) that were moved The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
sourceSheetId (Source Sheet Id)	An integer representing the internal id of the sheet where the cell(s) originally resided. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionMove) is located in §A.2. *end note*]

18.11.1.20 rqt (Revision Query Table)

This element represents a revision record of a query table field change.

Attributes	Description
fieldId (Field Id)	ID of the specific query table field that was removed. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ref (QueryTable Reference)	Location of the affected query table. The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionQueryTableField) is located in §A.2. *end note*]

18.11.1.21 rrc (Revision Row Column Insert Delete)

This element represents a revision record of a row/column insert/delete action.

Attributes	Description
action (User Action)	Indicates the action most recently performed on the row or column. The possible values for this attribute are defined by the ST_rwColActionType simple type (§18.18.66).
edge (Edge Deleted)	A Boolean flag indicating that a row or column is being deleted at the edge of a sorted range (only applicable to a Delete Row/Column revision types). The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
eol (End Of List)	A Boolean flag indicating that a row or a column is being inserted at the end of a list of data. List in this context does not mean table, rather it refers to the feature where the spreadsheet application automatically creates an internal structure for making data input more consistent on adjacent rows or columns. For instance, if 3 cells in a row are entered with the same format, then when entering data into the 4th adjacent cell, the spreadsheet application might automatically apply that same format. In this case, those

Attributes	Description
	cells are treated as a list. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ref (Reference)	A reference to the location of the rows/columns that were inserted or deleted. [*Note*: A reference to a whole column or row must include both the column and row components. For example, column A is referenced by "A1:A1048576", and row 1 is referenced by "A1:XFD1". However, because this attribute value is occurring in the context of an entire row or column insert, the column component of a row reference can be ignored, and the row component of a column reference can be ignored. *end note*] The possible values for this attribute are defined by the ST_Ref simple type (§18.18.62).
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionRowColumn) is located in §A.2. *end note*]

18.11.1.22 rsnm (Revision Sheet Name)

This element represents a revision record tracking the renaming a sheet.

Attributes	Description
newName (New Sheet Name)	A string representing the new sheet name

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
oldName (Old Sheet Name)	A string representing the old sheet name The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
ra (Revision Undo Rejected)	A Boolean flag which indicates that this revision was due to a previous undo (ua) revision being rejected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rId (Revision Id)	An integer representing the number of this revision. This id shall apply to reviewable revision types only. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sheetId (Sheet Id)	An integer representing the internal id of the sheet on which the revision occurred. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
ua (Revision From Rejection)	A Boolean flag indicating that this revision occurred because another revision was rejected and therefore undone. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_RevisionSheetRename) is located in §A.2. *end note*]

18.11.1.23 sheetId (Sheet Id)

This element represents a sheet that revision can take place on. Each sheet in the workbook should be represented by one of these elements, and each sheet has an id associated with it. Sheet ids are used to refer to sheets internally by the spreadsheet application.

Attributes	Description
val (Sheet Id)	An integer serving as a number by which to reference the sheet internally. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SheetId) is located in §A.2. *end note*]

18.11.1.24 sheetIdMap (Sheet Id Map)

This element represents a list of sheets and corresponding ids that are used for tracking revision records.

Attributes	Description
count (Sheet Count)	Number of sheets. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SheetIdMap) is located in §A.2. *end note*]

18.11.1.25 undo (Undo)

This element represents undo information for row/column deletion when there are functions in the spreadsheet that reference the deleted rows/columns. This element is not applicable for insert revisions.

Attributes	Description
array (Array Formula)	Flag indicating that the affected formula is an array formula. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
cs (Cross Sheet Move)	A Boolean flag indicating this was a cross-sheet move. True if it was a cross sheet move, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dn (Defined Name)	Identifies the named range that referenced the deleted cell range. Mutually exclusive with the cell reference attribute. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
dr (Range)	The range which was deleted that is referenced by the affected formula. The possible values for this attribute are defined by the ST_RefA simple type (§18.18.63).
exp (Expression)	Identifies the expression that should be adjusted in the corresponding formula. The possible values for this attribute are defined by the ST_FormulaExpression simple type (§18.18.36).
index (Index)	Index of the expression within the corresponding formula that was affected by this change.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
nf (Defined Name Formula)	A Boolean flag indicating that the corresponding formula is part of a defined name. True if this formula is part of a defined name, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
r (Cell Reference)	Location of the cell whose formula referenced the deleted cell range. Mutually exclusive with the defined name attribute The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
ref3D (Reference 3D)	A Boolean flag indicating that the expression contained the sheet name in addition to the cell reference. True if it contained the sheet name, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sId (Sheet Id)	Internal Id of the worksheet that contained the formula that referenced the deleted cell range. Mutually exclusive with the defined name attribute. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
v (Value Needed)	A Boolean flag indicating the formula needs the actual value of the cell(s) it's referencing. True if the formula requires the value of the cell it references, false otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_UndoInfo) is located in §A.2. *end note*]

18.11.2 Shared Workbook User Data

This subclause specifies information about the users of a shared workbook.

18.11.2.1 userInfo (User Information)

This element represents a user, and it stores information about a specific user as it relates to revisions.

Attributes	Description
dateTime (Date Time)	Date and time when this user opened the shared workbook. The possible values for this attribute are defined by the W3C XML Schema dateTime

Attributes	Description
	datatype.
guid (User Revisions GUID)	A globally unique identifier identifying the last set of revisions that this uses is synchronized to. This attribute can be used by the spreadsheet application to ensure that revisions this user depends on aren't deleted. The possible values for this attribute are defined by the ST_Guid simple type (§22.9.2.4).
id (User Id)	An integer representing an internal user id for this user. This number can be negative. The possible values for this attribute are defined by the W3C XML Schema int datatype.
name (User Name)	Display name for this user [*Note*: User name strings should not be longer than 54 characters. *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SharedUser) is located in §A.2. *end note*]

18.11.2.2 users (User List)

This element represents a list of users who currently have this shared workbook open. This list does not include any users who have the workbook open in Read-Only mode.

Attributes	Description
count (Active User Count)	Number of users who currently have this shared workbook open. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Users) is located in §A.2. *end note*]

18.12 QueryTable Data

Query tables are 2 dimensional tables of data bound to an external query of some kind. [*Example*: A query table could for example show specific data from a text file, from a web query, or from a database query. *end example*]

[*Example*:

Data connectivity can use a number of different technologies. The following spreadsheetML fragment is one an example of a query table connected to a database:

```
<queryTable xmlns="…" name="Northwind Orders" rowNumbers="1"
  growShrinkType="overwriteClear" connectionId="1" autoFormatId="16"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="0"
  applyPatternFormats="0" applyAlignmentFormats="0" applyWidthHeightFormats="0">
  <queryTableRefresh nextId="15">
    <queryTableFields count="12">
      <queryTableField id="1" name="OrderID" tableColumnId="1"/>
      <queryTableField id="2" name="CustomerID" tableColumnId="2"/>
      <queryTableField id="3" name="EmployeeID" tableColumnId="3"/>
      <queryTableField id="4" name="OrderDate" tableColumnId="4"/>
      <queryTableField id="5" name="RequiredDate" tableColumnId="5"/>
      <queryTableField id="6" name="ShippedDate" tableColumnId="6"/>
      <queryTableField id="7" name="ShipName" tableColumnId="7"/>
      <queryTableField id="8" name="ShipAddress" tableColumnId="8"/>
      <queryTableField id="9" name="ShipCity" tableColumnId="9"/>
      <queryTableField id="10" name="ShipRegion" tableColumnId="10"/>
      <queryTableField id="11" name="ShipPostalCode" tableColumnId="11"/>
      <queryTableField id="12" name="ShipCountry" tableColumnId="12"/>
    </queryTableFields>
  </queryTableRefresh>
</queryTable>
```

end example]

[*Example*: And here's an example of the SpreadsheetML fragment defining a query table connected to a text import:

```
<queryTable xmlns="…" name="data in text" connectionId="1" autoFormatId="16"
  applyNumberFormats="0" applyBorderFormats="0" applyFontFormats="1"
  applyPatternFormats="1" applyAlignmentFormats="0"
  applyWidthHeightFormats="0"/>
```

Elsewhere in the spreadsheetML file, a connection element is defined with the name "Northwind Orders" that describes how to connect to the appropriate database to refresh data for the query table. *end example*]

18.12.1 deletedField (Deleted Field)

This element specifies a field that has been deleted from the query table.

[*Example*:


```
<queryTableDeletedFields count="2">
  <deletedField name="ShipVia"/>
  <deletedField name="Freight"/>
</queryTableDeletedFields>
```

end example]

Attributes	Description
name (Deleted Fields Name)	Specifies the name of the deleted field. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DeletedField) is located in §A.2. *end note*]

18.12.2 queryTable (Query Table)

This element specifies all the relevant properties for a query table, one query table element is stored for each query table object in the spreadsheetML document.

Attributes	Description
adjustColumnWidth (Adjust Column Width On Refresh)	Specifies whether to automatically adjust column widths on refresh to fit the data retrieved. `true` if column widths should be adjusted. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyAlignmentFormats (Apply Alignment Formats)	If `true` apply legacy table autoformat alignment properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyBorderFormats (Apply Border Formats)	If `true` apply legacy table autoformat border properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyFontFormats (Apply Font Formats)	If `true` apply legacy table autoformat font properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyNumberFormats (Apply Number Formats)	If `true` apply legacy table autoformat number format properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyPatternForm	If `true` apply legacy table autoformat pattern properties.

Attributes	Description
ats (Apply Pattern Formats)	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
applyWidthHeightFormats (Apply Width / Height Formats)	If true apply legacy table autoformat width/height properties. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoFormatId (Auto Format Id)	Identifies which legacy table autoformat to apply. Annex D contains a listing of the supported PivotTable AutoFormats, example formatting, and a sample workbook with each of those AutoFormats applied. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
backgroundRefresh (Background Refresh)	Specifies whether or not the query table shall try to refresh data in the background. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
connectionId (Connection Id)	Specifies the ID number of the external data connection to use to refresh data in the query table. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
disableEdit (Disable Edit)	Specifies whether the connection element used with this query table shall be editable. If true, then the connection is not editable. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
disableRefresh (Disable Refresh)	Specifies whether the query table shall be refreshable. If true, then then query table is not refreshable. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
fillFormulas (Fill Adjacent Formulas)	Specifies whether or not formulas in columns adjacent to the query table should be filled down whenever the query table is refreshed. This is helpful since the number of rows returned by a query table refresh operation can vary. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
firstBackgroundRe	Specifies whether or not data has ever been refreshed for this query table. If the very

Attributes	Description
fresh (First Background Refresh)	first background data refresh had not completed at the time the file was saved, this attribute is set to `true`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
growShrinkType (Grow Shrink Type)	Specifies the type of behavior expected for dealing with a variable number of rows of data in the query table between refresh operations. The meaning of the possible values of this attribute {`insertClear`, `insertDelete`, `overwriteClear`} are explained in detail in the definition of the simple type. The possible values for this attribute are defined by the ST_GrowShrinkType simple type (§18.18.39).
headers (First Row Column Titles)	Specifies whether or not the query table has first row with column titles. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
intermediate (Intermediate)	Specifies whether this query table is in an intermediate state, having been defined but not fully formed and populated with data. In this state, fields and ranges of the query table can be unknown. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
name (QueryTable Name)	Specifies the name of the query table. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
preserveFormatting (Preserve Formatting On Refresh)	Specifies whether the application should try to preserve formatting in the query table and copy this formatting to any new rows of data. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
refreshOnLoad (Refresh On Load)	Specifies whether the query table shall refresh its data automatically when the spreadsheetML document is loaded or opened. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
removeDataOnSave (Remove Data On Save)	Specifies whether the query table shall remove all data from the worksheet before the spreadsheetML document is saved. This is very helpful for situations where people who have different permissions to view data want to share the same spreadsheetML document. All data from the last user is

Attributes	Description
	removed, and new users re-query the external data sources with their own credentials. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rowNumbers (Row Numbers)	Specifies whether the query table shall include a first column of row numbers. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_QueryTable) is located in §A.2. *end note*]

18.12.3 queryTableDeletedFields (Deleted Fields)

This element is the collection for deletedField (§18.12.1) elements, each of which represents a column or field that has been deleted from the query table.

[*Example*:

```
<queryTableDeletedFields count="2">
  <deletedField name="ShipVia"/>
  <deletedField name="Freight"/>
</queryTableDeletedFields>
```

end example]

Attributes	Description
count (Deleted Fields Count)	Specifies how many deleted fields there are. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_QueryTableDeletedFields) is located in §A.2. *end note*]

18.12.4 queryTableField (QueryTable Field)

This element holds the properties related to a specific field or column in a query table.

Attributes	Description
clipped (Clipped Column)	Specifies whether this field/column is currently clipped and thus not visible in the worksheet.

Attributes	Description
	[*Note*: this state might occur for example when a query table is defined near the edge of a worksheet or other object in the spreadsheet that can't be overwritten with external data. In this case some of the fields are displayed, but not all of them. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dataBound (Data Bound Column)	Specifies whether this column is a user-defined column or comes from the external data query. User defined columns shall be preserved during data refresh operations. User-defined columns are only supported on query tables that are attached to table objects. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
fillFormulas (Fill This Formula On Refresh)	Specifies whether the formula in this field/column should be filled down on data refresh. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
id (Field Id)	Specifies the unique identifier of the query table field. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (Name)	Specifies the unique name of the query table field. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
rowNumbers (Row Numbers)	`true` if this column contains the row numbers for the records returned. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
tableColumnId (Table Column Id)	Specifies the unique identifier for the table column if the query table is attached to a table object rather than just a range in the sheet. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_QueryTableField) is located in §A.2. *end note*]

18.12.5 queryTableFields (Query table fields)

This element is the collection for queryTableField elements.

Attributes	Description
count (Column	Specifies the number of columns there are in this query table. Includes both query-

Attributes	Description
Count)	defined and user-defined columns, but not deleted columns. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_QueryTableFields) is located in §A.2. *end note*]

18.12.6 queryTableRefresh (QueryTable Refresh Information)

This element contains information related to refreshing the query table.

Attributes	Description
fieldIdWrapped (Next Field Id Wrapped)	Whether or not the idFieldNext value wrapped around. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
headersInLastRefresh (Headers In Last Refresh)	Whether or not the Query Table had titles last refresh. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
minimumVersion (Minimum Refresh Version)	For backward compatibility with legacy versions of spreadsheet applications, this attribute specifies the minimum version of the application that is expected to correctly refresh the data in the query table without any problems. If this attribute is specified, an earlier version of a spreadsheet application should alert the user to the potential incompatibilities when a refresh is attempted. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
nextId (Next field id)	Specifies the next unique queryTableField (§18.12.4) id number available for assignment. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
preserveSortFilterLayout (Preserve Sort & Filter Layout)	Specifies whether sorting, autofilter, layout, and table block formatting should be preserved for this query table across data refresh operations. If this attribute is set to `false`, the query table might be more or less recreated from scratch when data is refreshed. In this case, all user deleted or rearranged columns, user inserted columns that aren't bound to external data, and table column formatting are discarded.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
unboundColumnsLeft (Columns Left)	Specifies the number of extra columns included at the left end of the field array that aren't bound to external data. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
unboundColumnsRight (Columns Right)	Specifies the number of extra columns included at the right end of the Table that aren't bound to external data. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_QueryTableRefresh) is located in §A.2. *end note*]

18.13 External Data Connections

SpreadsheetML allows for the definition of top level data connection objects that describe how to retrieve data from external sources. These connection objects are independent of the constructs in the spreadsheet application that display data such as tables, PivotTables, etc.

Some information about a connection is considered part of the connection's definition. Other information is not inherently part of the connection, but it describes the way the connection is to be used by the containing workbook. Note that in many cases, the spreadsheet application does not need knowledge of the command syntax for the external data source (e.g., database query language), and simply stores a command string that was created by a data provider API (e.g., an ODBC driver).

A connection's definition can be established in a standalone connection file for easier sharing and reuse, but this reference documentation deals with the XML representation for external data connections that is directly embedded within a SpreadsheetML document. This embedded representation is expected whenever external data is used, and ensures portability of the document and continued operation of the external query in the most cases.

18.13.1 connection (Connection)

This element contains both the definition of how to get at an external data source as well as information describing how the connection is used within the workbook. Specific constructs in a worksheet, such as OLAP formulas, QueryTables, or PivotTables make use of information in the connection to retrieve or refresh data based on default events or the user's explicit request.

Attributes	Description
background	Indicates whether the connection can be refreshed in the background (asynchronously).

Attributes	Description
(Background Refresh)	true if preferred usage of the connection is to refresh asynchronously in the background; false if preferred usage of the connection is to refresh synchronously in the foreground. This flag should be intentionally ignored in specific cases. [*Example*: An example of when the flag would be ignored is in the case of a connection to OLAP data on Microsoft SQL Server Analysis Services, where the connection is used by both a PivotTable and also by CUBE functions within the workbook. That connection will always be refreshed synchronously by the PivotTable and will always be refreshed asynchronously by the CUBE functions. *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
credentials (Reconnection Method)	Specifies the authentication method to be used when establishing (or re-establishing) the connection. The possible values for this attribute are defined by the ST_CredMethod simple type (§18.18.16).
deleted (Deleted Connection)	Indicates whether the associated workbook connection has been deleted. true if the connection has been deleted; otherwise, false. Deleted connections contain only the attributes name and deleted=true, all other information is removed from the SpreadsheetML file. If a new connection is created with the same name as a deleted connection, then the deleted connection is overwritten by the new connection. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
description (Connection Description)	Specifies the user description for this connection. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
id (Connection Id)	Specifies The unique identifier of this connection. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
interval (Automatic Refresh Interval)	Specifies the number of minutes between automatic refreshes of the connection. When this attribute is not present, the connection is not automatically refreshed. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
keepAlive (Keep Connection Open)	true when the spreadsheet application should make efforts to keep the connection open. When false, the application should close the connection after retrieving the

Attributes	Description
	information. This corresponds to the MaintainConnection property of a PivotCache object. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
minRefreshableVersion (Minimum Version Required for Refresh)	For compatibility with legacy spreadsheet applications. This represents the minimum version # that is required to be able to correctly refresh the data connection. This attribute applies to connections that are used by a QueryTable. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
name (Connection Name)	Specifies the name of the connection. Each connection shall have a unique name. When a connection has been marked as deleted and then a new connection is added with the same name, the deleted connection is replaced with the new connection. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
new (New Connection)	`true` if the connection has not been refreshed for the first time; otherwise, `false`. This state can happen when the user saves the file before a query has finished returning. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
odcFile (Connection File)	Specifies the full path to external connection file from which this connection was created. If a connection fails during an attempt to refresh data, and reconnectionMethod=1, then the spreadsheet application will try again using information from the external connection file instead of the connection object embedded within the workbook. This is a benefit for data source and spreadsheetML document manageability. If the definition in the external connection file is changed (e.g., because of a database server name change), then the workbooks that made use of that connection will fail to connect with their internal connection information, and reload the new connection information from this file. This attribute is cleared by the spreadsheet application when the user manually edits the connection definition within the workbook. Can be expressed in URI or system-specific file path notation. [*Note*: Applications can decide what forms of URI they support, and whether system-specific file path notations are supported. *end note*] The possible values for this attribute are defined by the ST_Xstring simple type

Attributes	Description
	(§22.9.2.19).
onlyUseConnection File (Only Use Connection File)	Indicates whether the spreadsheet application should always and only use the connection information in the external connection file indicated by the odcFile attribute when the connection is refreshed. If `false`, then the spreadsheet application should follow the procedure indicated by the reconnectionMethod attribute described below. Applies to ODBC connections, and may be applied to custom data connections. This attribute is ignored for other types of connections. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
reconnectionMeth od (Reconnection Method)	Specifies what the spreadsheet application should do when a connection fails. The values are as follows: `1 = As required`: On refresh use the existing connection information. If the existing information cannot be used to establish a connection, get updated connection information, if available from the external connection file. `2 = Always`: On every refresh get updated connection information from the external connection file, if available, and use that instead of the existing connection information. In this case the data refresh will fail if the external connection file is unavailable. `3 = Never`: Never get updated connection information from the external connection file even if it is available and even if the existing connection information cannot be used. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
refreshedVersion (Last Refresh Version)	For backward compatibility purposes, this attribute indicates the version of the spreadsheet application that last refreshed the connection. This attribute applies to connections that are used by a query table. The possible values for this attribute are defined by the W3C XML Schema unsignedByte datatype.
refreshOnLoad (Refresh on Open)	`true` if this connection should be refreshed when opening the file; otherwise, `false`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
saveData (Save Data)	`true` if the external data fetched over the connection to populate a table is to be saved with the workbook; otherwise, `false`.

Attributes	Description
	This exists for data security purposes - if no external data is saved in (or "cached") in the workbook, then current user credentials can be required every time to retrieve the relevant data, and people won't see the data the workbook author had last been using before saving the file. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
savePassword (Save Password)	`true` if the password is to be saved as part of the connection string; otherwise, False. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
singleSignOnId (SSO Id)	Identifier for Single Sign On (SSO) used for authentication between an intermediate spreadsheetML server and the external data source. [*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/spptsdk/html/cSSOReturnCodes_SV01001109.asp *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
sourceFile (Source Database File)	Used when the external data source is file-based. When a connection to such a data source fails, the spreadsheet application attempts to connect directly to this file. Can be expressed in URI or system-specific file path notation. [*Note*: Applications can decide what forms of URI they support, and whether system-specific file path notations are supported. *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
type (Database Source Type)	Specifies the data source type. Values are as follows: 1. ODBC-based source 2. DAO-based source 3. File based database source 4. Web query 5. Custom data connection source 6. Text-based source 7. ADO record set 8. DSP Custom data connection source represents an application-defined connection technology. [*Note*: For example, Microsoft Office uses this value to represent OLE DB connections. *end note*]

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Connection) is located in §A.2. *end note*]

18.13.2 connections (Connections)

This element exists when there are one or more connections in the workbook. It is a container for the individual connection objects.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Connections) is located in §A.2. *end note*]

18.13.3 dbPr (Database Properties)

This element stores all properties associated with an ODBC or OLE DB external data connection.

[*Example*:

Data connectivity can use a number of different technologies. The following is one example XML fragment defining an OLE DB connection and the associated dbPr element:

```
<connection id="2"
  odcFile="C:\My Documents\My Data Sources\Northwind Orders.odc" keepAlive="1"
  name="Northwind Orders" description="northwind" type="5" refreshedVersion="3">
  <dbPr connection="Provider=SQLOLEDB.1;Persist
    Security Info=True;Initial Catalog=Northwind;Data Source=dataserver1;Use
    Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation
    ID=LOCAL_MACHINE_NAME;Use Encryption for Data=False;Tag with column
    collation when possible=False"
    command="&quot;Northwind&quot;.&quot;dbo&quot;.&quot;Orders&quot;"
    commandType="3"/>
</connection>
```

end example]

Attributes	Description
command (Command Text)	The string containing the database command to pass to the data provider API that will interact with the external source in order to retrieve data. These strings can be constructed in a variety of ways (from simple UIs built into the spreadsheet application for browsing and choosing tables and fields, to external applications providing user interface to build up complex queries, to advanced users editing text queries). The

Attributes	Description
	spreadsheetML application need not understand the command syntax; it can simply pass the command string to the data provider API in order to retrieve the latest external data. [*Example*: Data connectivity can use a number of different technologies. The following is one example of an ODBC command string of commandType=2 (for a Microsoft SQL Server database): `command="SELECT Orders.OrderID, Orders.OrderDate, Orders.ShipName, Orders.ShipAddress, Orders.ShipCity, Orders.ShipRegion, Orders.ShipPostalCode, Orders.ShipCountry_x000d__x000a_FROM Northwind.dbo.Orders Orders_x000d__x000a_WHERE (Orders.ShipCountry=?)"` Some characters in this string have been escaped - for more information on the escaping scheme, please refer to the ST_Xstring simple type definition. *end example*] [*Note*: the "?" syntax in the string is something that the ODBC data provider is aware of and might replace with a parameter before execution. *end note*] [*Example*: Data connectivity can use a number of different technologies. The following is one example of an OLE DB command string of commandType=3 (for an Oracle database): `command=""TESTDB"."ShippersTable""` *end example*] [*Note*: Data connectivity can use a number of different technologies. A few examples of potential values stored in this attribute can be found at: • http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odbcsql_statements.asp • http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odbcsql_minimum_grammar.asp • http://msdn.microsoft.com/library/default.asp?url=/library/en-us/oledb/htm/oledbusing_commands.asp *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
commandType (custom data source Command Type)	Specifies the custom data source command type. Values are passed to the custom data source provider. [*Example*: For the OLE DB custom data source provider, valid values are as follows: 1. Query specifies a cube name 2. Query specifies a SQL statement

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>3. Query specifies a table name
4. Query specifies that default information has been given, and it is up to the provider how to interpret.
5. Query is against a web based List Data Provider. end example]

The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.</td></tr>
<tr><td>connection (Connection String)</td><td>The connection string is used to make contact with an ODBC or custom data source. These can be constructed in a variety of ways (from UI wizards built into the data provider code, to external query applications, to advanced users editing text files). The spreadsheetML application need not understand the connection syntax at all; it can simply pass the command string to the data provider API in order to re-establish a connection with the external data source.

[Example: ODBC connection string to a database:

<code>connection="DRIVER=SQL Server;SERVER=example_server;UID=example_useralias;APP=Microsoft Office 2007;WSID=user_alias;Trusted_Connection=Yes"</code>

end example]

[Example: of an OLE DB connection string to an Oracle database:

<code>connection="Provider=OraOLEDB.Oracle.1;Password=example_password;Persist Security Info=True;User ID=example_useralias;Data Source=example_server;Extended Properties="""</code>

end example]

[Note: Data connectivity can use a number of different technologies. A few examples of potential values stored in this attribute can be found at:
• http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/dasdkodbcoverview.asp
• http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbcsql/od_odbc_d_4x4k.asp
• http://msdn.microsoft.com/library/default.asp?url=/library/en-us/ado270/htm/mdreforacleprovspec.asp
end note]

Connection strings syntaxes are specific to individual ODBC or custom data provider drivers.

The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).</td></tr>
</table>

Attributes	Description
serverCommand (Command Text)	Specifies a second command text string that is persisted when PivotTable server-based page fields are in use. For ODBC connections, serverCommand is usually a broader query than command (no WHERE clause is present in the former). Based on these 2 commands, parameter UI can be populated and parameterized queries can be constructed. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DbPr) is located in §A.2. *end note*]

18.13.4 m (No Value)

This element is present when tables in a web query are missing.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TableMissing) is located in §A.2. *end note*]

18.13.5 olapPr (OLAP Properties)

This element contains all the properties needed for an OLAP data connection. OLAP connections contain both the dbPr and olapPr child elements.

[*Example*:

Data connectivity can use a number of different technologies. The following is an example of a connection to an SAP BW OLAP data source:

```
<connection id="1" odcFile="C:\My Documents\My Data Sources\$INFOCUBE.odc"
  keepAlive="1" name="SAP demo cube" description="SAP DemoCube" type="5"
  refreshedVersion="3" background="1">
  <dbPr connection="Provider=MDrmSap.2;Data Source=BI2;User
    ID=TESTUSER;Location=TESTSERVERNAME;Cache Authentication=False;Encrypt
    Password=False;Integrated Security=&quot;&quot;;Mask Password=False;Persist
    Encrypted=False;Persist Security Info=True;Impersonation
    Level=Anonymous;Mode=Read;Protection Level=None;Extended
    Properties=&quot;SFC_CLIENT=800;&quot;;Initial Catalog=$INFOCUBE"
    command="$0D_DECU" commandType="1"/>
  <olapPr sendLocale="1" rowDrillCount="1000" serverFill="0"
    serverNumberFormat="0" serverFont="0" serverFontColor="0"/>
</connection>
```

end example]

[*Example*:

Data connectivity can use a number of different technologies. The following is an example of a connection to a Microsoft SQL Server Analysis Services OLAP data source:

```
<connection id="1"
  odcFile="C:\My Documents\My Data Sources\Adventure Works DW.odc" keepAlive="1"
  name="Adventure Works DW" type="5" refreshedVersion="3" background="1">
  <dbPr connection="Provider=MSOLAP.3;Cache Authentication=False;Persist
Security Info=True;Initial Catalog=Adventure Works
    DW;Data Source=DATASERVER1;Impersonation
    Level=Impersonate;Mode=ReadWrite;Protection Level=Pkt Privacy;Auto Synch
    Period=20000;Default Isolation Mode=0;Default MDX Visual Mode=0;MDX
    Compatibility=1;MDX Unique Name Style=0;Non Empty
    Threshold=0;SQLQueryMode=Calculated;Safety Options=2;Secured Cell
    Value=0;SOURCE_DSN_SUFFIX=&quot;Prompt=CompleteRequired;Window
    Handle=0x6A903CC;&quot;;SQL Compatibility=0;Compression Level=0;Real Time
    Olap=False;Packet Size=4096" command="Adventure Works" commandType="1"/>
  <olapPr sendLocale="1" rowDrillCount="1000"/>
</connection>
```

end example]

[*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/oledb/htm/dasdkoledboverview.asp *end note*]

Attributes	Description
local (Local Cube)	Flag indicating whether we should get data from the local cube on refresh versus the original data source. `true` if a local cube has been created for OLAP data, and it should be used instead of the server. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
localConnection (Local Cube Connection)	Specifies a connection string to use when a local cube is available. This is used when local is set to `true`. [*Example*: `<olapPr local="true"` `localConnection="OLEDB;Provider=MSOLAP;Data` `Source=C:\Data\DataCube.cub" >` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
localRefresh (Local Refresh)	Flag indicating whether we should refresh the local cube from the original data source. When true, the original OLAP data source is queried each time the user explicitly refreshes the data in the application, and a new local cube is constructed from this query. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rowDrillCount (Drill Through Count)	Maximum number of drill-through rows to return when the user drills through an aggregate value in a PivotTable. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
sendLocale (Send Locale to OLAP)	When true, the spreadsheetML app should send the user interface locale ID to the OLAP provider to retrieve localized member names and properties, etc. When false, no locale ID is expected. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
serverFill (OLAP Fill Formatting)	When true a PivotTable based on an OLAP source should format the data and aggregate cells in the PivotTable view using the background color from the OLAP source if this information is available. When false, OLAP server background fill colors are ignored, and standard formatting rules within the worksheet are followed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
serverFont (OLAP Server Font)	When true, a PivotTable based on OLAP source should format the data and aggregate cells in the PivotTable view using the font from the OLAP source (e.g., Arial or Tahoma). When false, OLAP server fonts are ignored, and standard formatting rules within the worksheet are followed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
serverFontColor (OLAP Font Formatting)	When true a PivotTable based on OLAP source should format the data and aggregate cells in the PivotTable view using the font color from the OLAP source. When false, OLAP server font colors are ignored, and standard formatting rules within the worksheet are followed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
serverNumberFormat (OLAP Number Format)	When true, a PivotTable based on OLAP source should format the data and aggregate cells in the PivotTable view using the number format from the OLAP source. When false, OLAP server number formats are ignored, and standard formatting rules within

Attributes	Description
	the worksheet are followed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_OlapPr) is located in §A.2. *end note*]

18.13.6 parameter (Parameter Properties)

This element stores properties about any parameters used with external data connections. Parameters are used to change the query executed externally and cause different data to be retrieved into the workbook. The type of parameter used – see ST_parameterType (§18.18.54) – determines whether the user is prompted for a value before data is refreshed, or the value is pulled from a cell in the workbook, or whether the same value should be used until explicitly changed in the data connection. Parameters are permitted for ODBC and web queries.

[*Example*:

Data connectivity can use a number of different technologies. The following is an example of XML defining a connection to a Microsoft Access database, with a parameter based on the value in cell C1 on the first sheet.

```
<connection id="1" name="Connection" type="1" refreshedVersion="2"
  background="1" saveData="1">
  <dbPr connection="DSN=MS Access
    Database;DBQ=C:\Desktop\db1.mdb;DefaultDir=C:\Desktop;DriverId=25;FIL=MS
    Access;MaxBufferSize=2048;PageTimeout=5;" command="SELECT Table1.Field1,
    Table1.Field2_x000d__x000a_FROM `C:\Desktop\db1`.Table1
    Table1_x000d__x000a_WHERE (Table1.Field2=?)"/>
  <parameters count="1">
    <parameter name="user specified value" sqlType="4" parameterType="cell"
      cell="Sheet1!$C$1"/>
  </parameters>
</connection>
```

end example]

Note that the command string in the dbPr element contains a "?" character. This character serves as a parameter marker.

[*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/odbcstatement_parameters.asp *end note*]

Attributes	Description
boolean (Boolean)	Boolean value to use as the query parameter. Used only when parameterType = value. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
cell (Cell Reference)	Cell reference indicating which cell's value to use for the query parameter. Used only when parameterType = cell. [*Example*: <Parameter parameterType="cell" cell="Sheet1!$C$1"> *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
double (Double)	Non-integer numeric value to use as the query parameter. Used only when parameterType = value. The possible values for this attribute are defined by the W3C XML Schema double datatype.
integer (Integer)	Integer value to use as the query parameter. Used when parameterType = value. The possible values for this attribute are defined by the W3C XML Schema int datatype.
name (Parameter Name)	The name of the parameter. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
parameterType (Parameter Type)	Type of parameter used. If the parameterType=value, then the value from boolean, double, integer, or string are used. In this case, it is expected that only one of {boolean, double, integer, or string} is specified. The possible values for this attribute are defined by the ST_ParameterType simple type (§18.18.54).
prompt (Parameter Prompt String)	Prompt string for the parameter. Presented to the spreadsheet user along with input UI to collect the parameter value before refreshing the external data. Used only when parameterType = prompt. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
refreshOnChange (Refresh on Change)	Flag indicating whether the query should automatically refresh when the contents of a cell that provides the parameter value changes. If true, then external data is refreshed using the new parameter value every time there's a change. If false, then external data is only refreshed when requested by the user, or some other event triggers refresh (e.g., workbook opened).

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>The possible values for this attribute are defined by the W3C XML Schema boolean datatype.</td></tr>
<tr><td>sqlType (SQL Data Type)</td><td>SQL data type of the parameter. Only supported for ODBC sources.

Supported values include:
<table>
<tr><th>-22</th><th>SQL_UNSIGNED_OFFSET</th></tr>
<tr><td>-20</td><td>SQL_SIGNED_OFFSET</td></tr>
<tr><td>-11</td><td>SQL_GUID</td></tr>
<tr><td>-10</td><td>SQL_WLONGVARCHAR</td></tr>
<tr><td>-9</td><td>SQL_WVARCHAR</td></tr>
<tr><td>-8</td><td>SQL_WCHAR</td></tr>
<tr><td>-7</td><td>SQL_BIT</td></tr>
<tr><td>-6</td><td>SQL_TINYINT</td></tr>
<tr><td>-5</td><td>SQL_BIGINT</td></tr>
<tr><td>-4</td><td>SQL_LONGVARBINARY</td></tr>
<tr><td>-3</td><td>SQL_VARBINARY</td></tr>
<tr><td>-2</td><td>SQL_BINARY</td></tr>
<tr><td>-1</td><td>SQL_LONGVARCHAR</td></tr>
<tr><td>0</td><td>SQL_UNKNOWN_TYPE</td></tr>
<tr><td>1</td><td>SQL_CHAR</td></tr>
<tr><td>2</td><td>SQL_NUMERIC</td></tr>
<tr><td>3</td><td>SQL_DECIMAL</td></tr>
<tr><td>4</td><td>SQL_INTEGER</td></tr>
<tr><td>5</td><td>SQL_SMALLINT</td></tr>
<tr><td>6</td><td>SQL_FLOAT</td></tr>
<tr><td>7</td><td>SQL_REAL</td></tr>
<tr><td>8</td><td>SQL_DOUBLE</td></tr>
<tr><td>9</td><td>SQL_TYPE_DATE or SQL_DATE</td></tr>
<tr><td>10</td><td>SQL_TYPE_TIME or SQL_TIME</td></tr>
<tr><td>11</td><td>SQL_TYPE_TIMESTAMP or SQL_TIMESTAMP</td></tr>
<tr><td>12</td><td>SQL_VARCHAR</td></tr>
<tr><td>101</td><td>SQL_INTERVAL_YEAR</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td>102</td><td>SQL_INTERVAL_MONTH</td></tr>
<tr><td>103</td><td>SQL_INTERVAL_DAY</td></tr>
<tr><td>104</td><td>SQL_INTERVAL_HOUR</td></tr>
<tr><td>105</td><td>SQL_INTERVAL_MINUTE</td></tr>
<tr><td>106</td><td>SQL_INTERVAL_SECOND</td></tr>
<tr><td>107</td><td>SQL_INTERVAL_YEAR_TO_MONTH</td></tr>
<tr><td>108</td><td>SQL_INTERVAL_DAY_TO_HOUR</td></tr>
<tr><td>109</td><td>SQL_INTERVAL_DAY_TO_MINUTE</td></tr>
<tr><td>110</td><td>SQL_INTERVAL_DAY_TO_SECOND</td></tr>
<tr><td>111</td><td>SQL_INTERVAL_HOUR_TO_MINUTE</td></tr>
<tr><td>112</td><td>SQL_INTERVAL_HOUR_TO_SECOND</td></tr>
<tr><td>113</td><td>SQL_INTERVAL_MINUTE_TO_SECOND</td></tr>
</table>

The possible values for this attribute are defined by the W3C XML Schema int datatype.</td></tr>
<tr><td>string (String)</td><td>String value to use as the query parameter. Used only when parameterType = value.

The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).</td></tr>
</table>

[*Note*: The W3C XML Schema definition of this element's content model (CT_Parameter) is located in §A.2. *end note*]

18.13.7 parameters (Query Parameters)

This element serves as a collection of parameters for an ODBC or web query.

Attributes	Description
count (Parameter Count)	The number of parameters used. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Parameters) is located in §A.2. *end note*]

18.13.8 s (Character Value)

This element is used to specify an HTML table to import by name. If the tables are not named, they shall be specified with the `<x v="[index]">` syntax instead.

Attributes	Description
v (Value)	The name of the table to retrieve when the web query is refreshed. This corresponds to the string used for the id attribute of the HTML <table> tag. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_XStringElement) is located in §A.2. *end note*]

18.13.9 tables (Tables)

This element serves as the collection of tables to be returned via a web query data connection. Tables are then most commonly referenced by <x> via their indices (in order of the <Table> tags in the HTML page).

Attributes	Description
count (Count of Tables)	Number of tables to pull data from when refreshing from a web query. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Tables) is located in §A.2. *end note*]

18.13.10 textField (Text Import Field Settings)

This element specifies field settings for text import.

Attributes	Description
position (Position)	The character position the field starts at for fixed-length fields. The index is 0-based. If this attribute does not exist, position=0 is assumed. Subsequent textField elements or carriage returns in the text stream serve to denote endpoints for text fields. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
type (Field Type)	Specifies the field Type. When text is imported into cells in the worksheet, the data in the cells are converted to the field type defined here. Types can be specified by the user, or determined algorithmically via heuristics and text

Attributes	Description
	analysis. The possible values for this attribute are defined by the ST_ExternalConnectionType simple type (§18.18.27).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextField) is located in §A.2. *end note*]

18.13.11 textFields (Fields)

This element that denotes a set of fields to retrieve from a text file. Contains 1 or more textField elements.

Attributes	Description
count (Count of Fields)	Number of distinct fields to retrieve. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextFields) is located in §A.2. *end note*]

18.13.12 textPr (Text Import Settings)

This element contains all of the text import settings.

[*Example*: Here's an example of the XML for a text connection:

```
<connection id="1" name="text data" type="6" refreshedVersion="3" background="1"
  saveData="1">
  <textPr prompt="0" characterSet="IBM437" sourceFile="C:\Desktop\text data.txt"
    delimiter="|">
    <textFields count="5">
      <textField/>
      <textField type="text" position="7"/>
      <textField type="text" position="28"/>
      <textField position="36"/>
      <textField type="text" position="41"/>
    </textFields>
  </textPr>
</connection>
```

example]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>characterSet (Character Set)</td><td>Name of the character set associated with the text file. Values for this attribute are restricted to the names and aliases listed in the IANA CHARACTER SETS listing found at http://www.iana.org/assignments/character-sets.

[Note: When reading this value, if a system does not support a particular character set, the application is allowed to decide what is the best course of fallback action. end note]

If this attribute is not present then the codePage attribute are used.

The possible values for this attribute are defined by the W3C XML Schema string datatype.</td></tr>
<tr><td>comma (Comma is Delimiter)</td><td>Flag indicating whether to treat comma characters as field delimiters.

The possible values for this attribute are defined by the W3C XML Schema boolean datatype.</td></tr>
<tr><td>consecutive (Consecutive Delimiters)</td><td>Flag indicating whether consecutive delimiters should be treated as just one delimiter. If this flag is <code>true</code> than it's possible or even likely that some rows will return more fields than others, and these fields will always fill cells in the worksheet from left to right.

The possible values for this attribute are defined by the W3C XML Schema boolean datatype.</td></tr>
<tr><td>decimal (Decimal Separator)</td><td>The decimal separator character. This and the thousands attribute are used only when data in the text file contains decimal and thousands separators that are different from those used on the computer, due to a different language setting being used.

The following table shows the results when you import text into a spreadsheet application using various separators. Numeric results are displayed in the rightmost column.

<table>
<tr><th>System decimal separator</th><th>System thousands separator</th><th>Text file decimal separator value</th><th>Text file thousands Separator value</th><th>Text imported</th><th>Cell value (data type)</th></tr>
<tr><td>Period</td><td>Comma</td><td>Comma</td><td>Period</td><td>123.123,45</td><td>123,123.45 (numeric)</td></tr>
<tr><td>Period</td><td>Comma</td><td>Comma</td><td>Comma</td><td>123.123,45</td><td>123.123,45 (text)</td></tr>
<tr><td>Comma</td><td>Period</td><td>Comma</td><td>Period</td><td>123,123.45</td><td>123,123.45 (numeric)</td></tr>
<tr><td>Period</td><td>Comma</td><td>Period</td><td>Comma</td><td>123 123.45</td><td>123 123.45 (text)</td></tr>
<tr><td>Period</td><td>Comma</td><td>Period</td><td>Space</td><td>123 123.45</td><td>123,123.45 (numeric)</td></tr>
</table>

Strings values of this attribute are expected to be one character in length.</td></tr>
</table>

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
delimited (Delimited File)	`true` if the file is Tab or character delimited. `false` if the file should be parsed according to fixed length fields. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
delimiter (Custom Delimiter)	User-specified character to be treated as a field delimiter. Only single characters are supported. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
fileType (File Type)	Ignorable attribute with enum value that defined by ST_FileType. Determines the kind of character set to use during import. Only one of fileType and characterSet or codePage shall be specified for a textPr. The possible values for this attribute are defined by the ST_FileType simple type (§18.18.29).
firstRow (First Row)	Indicates at what row of the file to start the data import. All unsignedInt values are permitted, although it's possible that firstRow is higher than the number of rows in the text file, in which case no data is imported. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
prompt (Prompt for File Name)	Flag indicating whether the user wants to be prompted for the file name on refresh. If `false`, then the user is not prompted. If `true` or not present, then the user is prompted. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
qualifier (Qualifier)	Character used as the text string qualifier. The possible values for this attribute are defined by the ST_Qualifier simple type (§18.18.61).
semicolon (Semicolon is Delimiter)	Flag indicating whether to treat semicolon characters as field delimiters. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sourceFile (Source File Name)	Path to the text file to use to import external data. Can be expressed in URI or system-specific file path notation. [*Note*: Applications can decide what forms of URI they support, and whether system-

Attributes	Description
	specific file path notations are supported. *end note*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
space (Space is Delimiter)	Flag indicating whether to treat space characters as field delimiters. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
tab (Tab as Delimiter)	Flag indicating whether to treat tab characters as field delimiters. If `false`, then tabs will not be used as delimiters. If `true` or not present, then they are used as delimiters. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
thousands (Thousands Separator)	The thousands separator character. This and the decimal attribute are used only when data in the text file contains decimal and thousands separators that are different from those used on the computer, due to a different language setting being used. Please refer to the decimal attribute description above for a Table describing the behavior. Strings values of this attribute are expected to be one character in length. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextPr) is located in §A.2. *end note*]

18.13.13 webPr (Web Query Properties)

This element specifies the properties for a web query source. A web query will retrieve data from HTML tables, and can also supply HTTP "Get" parameters to be processed by the web server in generating the HTML by including the parameters and parameter elements.

Here's an example of a web query connection:

[*Example*:


```
<connection id="1" name="Connection" type="4" refreshedVersion="0"
  background="1" saveData="1">
  <webPr sourceData="1" parsePre="1" consecutive="1"
    url="http://ServerName/Image%20Library/Forms/AllItems.aspx" htmlTables="1">
    <tables count="1">
      <s v="contentthumbnail"/>
    </tables>
  </webPr>
</connection>
```

end example]

Attributes	Description
consecutive (Consecutive Delimiters)	Flag indicating whether consecutive delimiters should be treated as just one delimiter. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
editPage (Edit Query URL)	The URL of the user-facing web page showing the web query data. This URL is persisted in the case that sourceData="true" and url has been redirected to reference an XML file. Then the user-facing page can be shown in the UI, and the XML data can be retrieved behind the scenes. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
firstRow (Use First Row)	Flag indicating whether to parse all tables inside a PRE block with the same width settings as the first row. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
htmlFormat (HTML Formatting Handling)	How to handle formatting from the HTML source when bringing web query data into the worksheet. Relevant when sourceData is True. Values are as follows: 1. None - no formatting at all 2. RTF - honor just rich text formatting 3. All - honor all html formatting. The possible values for this attribute are defined by the ST_HtmlFmt simple type (§18.18.41).
htmlTables (HTML Tables Only)	Flag indicating whether web queries should only work on HTML tables. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
parsePre (Parse PRE)	Flag indicating whether data contained within HTML <PRE> tags in the web page is parsed into columns when you import the page into a query table.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
post (Web Post)	Returns or sets the string used with the post method of inputting data into a web server to return data from a web query. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
sourceData (Import XML Source Data)	Flag indicating that XML source data should be imported instead of the HTML table itself. Used when a web query exists to an HTML table with the following attribute. `<TABLE … o:WebQuerySourceHRef=”http://…” … > … </TABLE>` The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
textDates (Dates as Text)	Flag indicating whether dates should be imported into cells in the worksheet as text rather than dates. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
url (URL)	URL to use to refresh external data. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
xl2000 (Refreshed in Excel 2000)	This flag exists for backward compatibility with older existing spreadsheet files, and is set to `true` if this web query was refreshed in a spreadsheet application newer than or equal to Microsoft Excel 2000. This is an optional attribute that can be ignored. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
xl97 (Created in Excel 97)	This flag exists for backward compatibility with older existing spreadhseet files, and is set to `true` if this web query was created in Microsoft Excel 97. This is an optional attribute that can be ignored. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
xml (XML Source)	`true` if the web query source is XML (versus HTML), otherwise `false`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_WebPr) is located in §A.2. *end note*]

18.14 Supplementary Workbook Data

External links are used when linking the workbook to other workbooks or external data. The most frequent feature for linking a workbook to other workbooks is through the use of formulas. In this case the formula references a range or defined name in another workbook. Hyperlinks on cells and other spreadsheet objects are also considered an external link. Object-linking technologies are yet another technology used to link the workbook to another object. [*Example*: KParts or OLE. *end example*] Finally, Dynamic Data Exchange, or DDE, servers can be used to access external data. DDE servers are accessed through formulas in the workbook.

External links are saved with the target source in a relationship file so that external resources are easily discoverable in lightweight relationship XML rather than deep in the application's XML.

For a workbook consumer.xlsx that makes use of data in another workbook called data.xlsx, the following XML would exist in consumer.xlsx to describe the external link:

[*Example*:

```
<Relationships xmlns="…">
  <Relationship Id="rId1" Type="…/externalLinkPath" Target="data.xlsx"
    TargetMode="External"/>
</Relationships>
```

end example]

And the following XML would exist to describe cached data retrieved from the external workbook:

[*Example*:

```
<externalLink xmlns="…">
  <externalBook xmlns:r="…" r:id="rId1">
    <sheetNames>
      <sheetName val="Sheet1"/>
      <sheetName val="Sheet2"/>
      <sheetName val="Sheet3"/>
    </sheetNames>
```

```
      <sheetDataSet>
        <sheetData sheetId="0"/>
        <sheetData sheetId="1"/>
        <sheetData sheetId="2">
          <row r="11">
            <cell r="B11">
              <v>47</v>
            </cell>
          </row>
          <row r="12">
            <cell r="B12">
              <v>19</v>
            </cell>
          </row>
          <row r="13">
            <cell r="B13">
              <v>38</v>
            </cell>
          </row>
        </sheetData>
      </sheetDataSet>
    </externalBook>
  </externalLink>
```

end example]

The Supplementary Workbook Data section of SpreadsheetML is complimentary to the External Data Connnections (§18.13) section in maintaining all the information about external information that impacts the workbook.

18.14.1 cell (External Cell Data)

This element is used to store cached values from external sources such as other workbooks. Formulas from external cells are not stored in the consuming workbook. Also, for this context, the attribute t cannot have a value of `inlineStr`. Rich text is not supported in this context either.

Attributes	Description
r (Reference)	Describes the cell location in the external book. [*Example*: `<cell r="B12">` `  <v>74</v>` `</cell>`

Attributes	Description
	end example] The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
t (Type)	Indicates the data type of the cell value. The possible values for this attribute are defined by the ST_CellType simple type (§18.18.11).
vm (Value Metadata)	The index of the cell's value metadata, if any exists. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExternalCell) is located in §A.2. *end note*]

18.14.2 ddeItem (DDE Item definition)

This element represents a DDE item.

Attributes	Description
advise (Advise)	Specifies whether the DDE server should notify the application when the external data changes. Default value is `false`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
name (DDE Name)	Specifies the DDE item name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
ole (Object Linking TechnologyE)	Set to `true` if this is item uses an object linking technology. Default value is `false`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
preferPic (Data is an Image)	Set to `true` if data from this DDE item is an image format. Default value is `false`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DdeItem) is located in §A.2. *end note*]

18.14.3 ddeItems (DDE Items Collection)

This element serves as a collection for ddeItem elements.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DdeItems) is located in §A.2. *end note*]

18.14.4 ddeLink (DDE Connection)

This element represents a connection to an external Dynamic Data Exchange (DDE) server. DDE is a method of sending data between applications using Windows messages according to a documented protocol that has been stable since about 1990.

The hierarchy of names defined by a DDE server is Application, Topics, and Items. Topics often correspond to units such as files or documents or database names, and Items refer to subsets of the data such as cell ranges, rows, fields, columns. DDE items can have multiple values as well.

[*Example*:

Data connectivity can use a number of different technologies.The following is just one example of a spreadsheetML fragment describing the product Microsoft Excel being used as a DDE server to provide data to the current spreadsheet document:

```
<ddeLink xmlns:r="…" ddeService="excel" ddeTopic="[ddesource.xls]Sheet1">
  <ddeItems>
    <ddeItem name="R1C1" advise="1"/>
    <ddeItem name="StdDocumentName" ole="1" advise="1"/>
  </ddeItems>
</ddeLink>
```

end example]

Attributes	Description
ddeService (Service name)	Service name (i.e., application name) for the DDE connection. This is a required attribute. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
ddeTopic (Topic for DDE server)	Describes something for the DDE application to which the channel pertains— usually a document of that application. This is a required attribute. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DdeLink) is located in §A.2. *end note*]

18.14.5 definedName (Defined Name)

This element contains information about a named range in an external workbook.

Attributes	Description
name (Defined Name)	The defined name. This attribute is required. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
refersTo (Refers To)	Name range definition string. [*Example*: `<definedNames>` `  <definedName name="namedrange"` `    refersTo="='Sheet1'!$D$5:$D$10"/>` `</definedNames>` *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
sheetId (Sheet Id)	The index of the worksheet that the named range applies to for named ranges that are scoped to a particular worksheet rather than the full workbook. This attribute is optional. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExternalDefinedName) is located in §A.2. *end note*]

18.14.6 definedNames (Named Links)

This element is a collection of the defined names associated with the supporting workbook.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExternalDefinedNames) is located in §A.2. *end note*]

18.14.7 externalBook (External Workbook)

This element represents an external workbook which is supplying data to the current workbook.

Attributes	Description
id (Relationship to supporting book file	Relationship ID that references a link in the relationships collection. The target attribute in the associated relationship will specify the worksheet XML file in the current

Attributes	Description
path) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	spreadsheetML document ZIP archive that makes use of this externalbook. 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_ExternalBook) is located in §A.2. *end note*]

18.14.8 externalLink (External Reference)

This element is a container for specific types of external links.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExternalLink) is located in §A.2. *end note*]

18.14.9 oleItem (Object Link Item)

This element represents a single link within the object referenced by the parent element.

Attributes	Description
advise (Advise)	Set to `true` if the linked object should notify the application when the external data changes. Default value is `false`. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
icon (Icon)	Set to `true` if the linked object is represented by an icon. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
name (Object Name)	The linked object's name. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).
preferPic (Object is an Image)	Set to `true` if the linked object is represented by an image. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_OleItem) is located in §A.2. *end note*]

18.14.10 oleItems (Object Link Items)

This element is a collection of items within the link specified by the parent element.

[*Note*: The W3C XML Schema definition of this element's content model (CT_OleItems) is located in §A.2. *end note*]

18.14.11 oleLink (Generic Object Link Connection)

This element represents an external link to an embedded object, specified by a progID/object pair. The type of object link is determined by reading the target of the id attribute.

[*Example*: The following markup defines a reference to a linked object using Bonobo. The progId attribute contains the shared library that contains the widget. The r:id identifies the referenced Bonobo object.

```
<oleLink r:id="rb1" progId="OAFIID:Bonobo_Sample_Calculator">
  …
</oleLink>
```

The target of the relationship with ID rb1, defines the Bonobo object itself. This example shows a link to a sample Bonobo widget taken from the following article, which also provides an introduction to Bonobo – http://www.ibm.com/developerworks/webservices/library/co-bnbo2.html. *end example*]

[*Example*: The following markup defines a reference to a linked object using KParts. The progId attribute contains the shared library that contains the plugin. The r:id identifies the referenced KParts object.

```
<oleLink r:id="rKp1" progId="libhtmlvalidatorplugin">
  …
</oleLink>
```

The following XML, contained in the target of the relationship with ID rKp1, defines the KPart object, and will follow the kpartgui DTD:

```
<!DOCTYPE kpartgui SYSTEM "kpartgui.dtd">
<kpartgui library="libhtmlvalidatorplugin" name="htmlvalidatorplugin"
version="1" >
  <MenuBar>
    <Menu name="tools"><Text>&amp;Tools</Text>
      <Action name="validatewebpage"/>
    </Menu>
  </MenuBar>
</kpartgui>
```

This example is taken from the kde.org web site, and contains a tutorial on building the plugin referenced by the above markup – http://developer.kde.org/documentation/tutorials/dot/writing-plugins.html. *end example*]

Attributes	Description
id (Object Link Relationship) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Relationship ID that references a link in the relationships collection. The target attribute in the associated relationship will specify the external file name used for this oleLink. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
progId (Object Link Identifier)	The ID for the object link connection. [*Example*: For a KParts link, this would store the name of the appropriate KParts library. For an OLE link, this would store the ProgID of the appropriate OLE object. *end example*] The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_OleLink) is located in §A.2. *end note*]

18.14.12 row (Row)

This element contains data for an external worksheet row.

Attributes	Description
r (Row)	Row number of the row in the external book containing the cell data referenced. This attribute is required. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExternalRow) is located in §A.2. *end note*]

18.14.13 sheetData (External Sheet Data Set)

This element contains the cached worksheet data associated with a supporting workbook.

[*Note*: For an example, please refer to example at the beginning of this section. *end note*]

Attributes	Description
refreshError (Last Refresh Resulted in Error)	Specifies that the last external data refresh for this sheet did not succeed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

Attributes	Description
sheetId (Sheet Id)	Index of sheet in the external workbook that is referenced and partially cached in this data set. This is a 1-based index. This attribute is required. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExternalSheetData) is located in §A.2. *end note*]

18.14.14 sheetDataSet (Cached Worksheet Data)

This element serves as the collection for 1 or more sheetData elements.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExternalSheetDataSet) is located in §A.2. *end note*]

18.14.15 sheetName (Sheet Name)

Name of a worksheet in the supporting workbook

Attributes	Description
val (Sheet Name Value)	Name of the sheet. This attribute is required. The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExternalSheetName) is located in §A.2. *end note*]

18.14.16 sheetNames (Supporting Workbook Sheet Names)

This element is the container for all of the worksheet names in a supporting workbook.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExternalSheetNames) is located in §A.2. *end note*]

18.14.17 val (DDE Link Value)

This element specifies a value associated with a particular DDE item.

[*Example*: Here's an example of how values, value, and val elements are written out in the spreadsheetML for a ddeItem supplied by a DDE server. In this example different cells in the workbook are bound to these specific DDE items:

```
<ddeLink xmlns:r="…" ddeService="StockSrv" ddeTopic="Prices">
  <ddeItems>
    <ddeItem name="Bread" advise="1">
      <values>
        <value>
          <val>3.5</val>
        </value>
      </values>
    </ddeItem>
    <ddeItem name="Milk" advise="1">
      <values>
        <value>
          <val>5.7400000000000002</val>
        </value>
      </values>
    </ddeItem>
    <ddeItem name="MSFT" advise="1">
      <values>
        <value>
          <val>54.130000000000003</val>
        </value>
      </values>
    </ddeItem>
    <ddeItem name="StdDocumentName" ole="1" advise="1"/>
  </ddeItems>
</ddeLink>
```

end example]

The possible values for this element are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (ST_Xstring) is located in §A.6.9. *end note*]

18.14.18 value (Value)

This element contains a value associated with a particular DDE item. This serves as a container for the val element.

Attributes	Description
t (DDE Value Type)	Indicates the DDE value type. The possible values for this attribute are defined by the ST_DdeValueType simple type (§18.18.23).

[*Note*: The W3C XML Schema definition of this element's content model (CT_DdeValue) is located in §A.2. *end note*]

18.14.19 values (DDE Name Values)

This element defines a collection of values associated with DDE item.

Attributes	Description
cols (Columns)	The number of columns of data that is returned by the DDE server for this DDE item. The default value of this attribute is 1. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rows (Rows)	The number of rows of data that is returned by the DDE server for this DDE item. The default value of this attribute is 1. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DdeValues) is located in §A.2. *end note*]

18.15 Volatile Dependencies

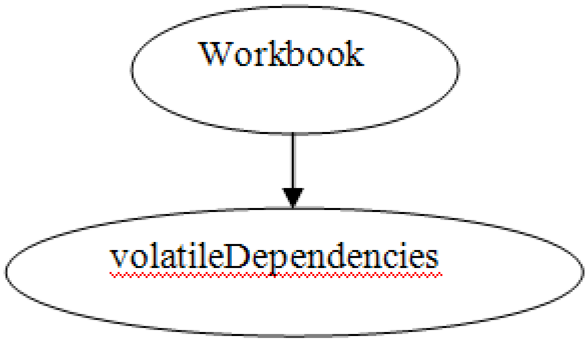
The volatileDependencies part provides a cache of data that supports Real Time Data (RTD) and CUBE functions in the workbook. Both of these types of functions require connectivity to external servers to retrieve their data. For RTD functions, an RTD interface defines how data is provided on the server, and how it is retrieved on the client. Similarly, CUBE functions access data in OLAP cubes via their own function syntax. The volatileDependencies part provides that cache of data and supporting information about these functions and their data servers and connections. This allows the spreadsheet application to work with cached values when recalculating the workbook when the external server is not available.

[*Note*: How users of SpreadsheetML access RTD data depends on the integration the user's spreadsheet application provides for RTD. *end note*]

[*Note*: Data connectivity can use a number of different technologies. One example of potential values stored in this attribute can be found at: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbaxl11/html/xlobjIRtdServer_HV03085058.asp *end note*]

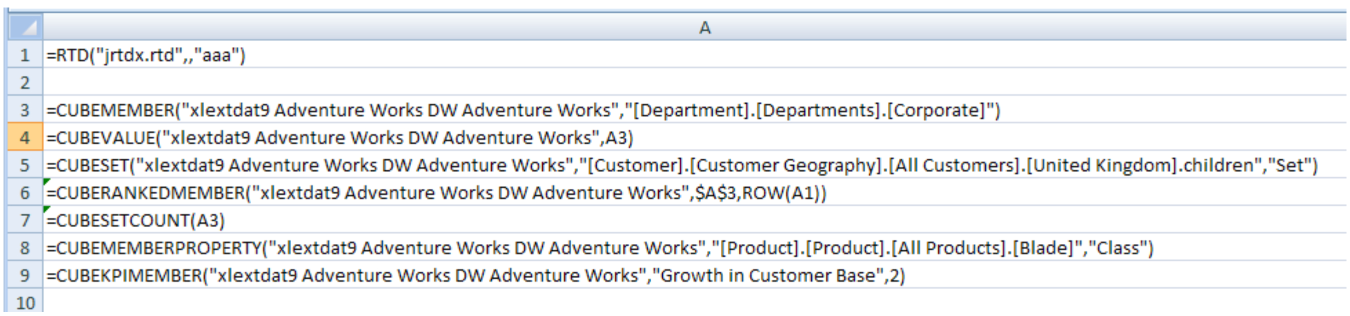
File Architecture

The workbook holds the relationship to the volatile dependencies part.



Illustration

[*Example*: The following image shows an example implementation of CUBE and Real Time Data (RTD) functions in a worksheet.



	A
1	=RTD("jrtdx.rtd",,"aaa")
2	
3	=CUBEMEMBER("xlextdat9 Adventure Works DW Adventure Works","[Department].[Departments].[Corporate]")
4	=CUBEVALUE("xlextdat9 Adventure Works DW Adventure Works",A3)
5	=CUBESET("xlextdat9 Adventure Works DW Adventure Works","[Customer].[Customer Geography].[All Customers].[United Kingdom].children","Set")
6	=CUBERANKEDMEMBER("xlextdat9 Adventure Works DW Adventure Works",$A$3,ROW(A1))
7	=CUBESETCOUNT(A3)
8	=CUBEMEMBERPROPERTY("xlextdat9 Adventure Works DW Adventure Works","[Product].[Product].[All Products].[Blade]","Class")
9	=CUBEKPIMEMBER("xlextdat9 Adventure Works DW Adventure Works","Growth in Customer Base",2)
10	

The following example shows the XML that describes the functions in the illustration.

```
<volTypes xmlns="…">
  <volType type="realTimeData">
    <main first="jrtdx.rtd">
      <tp t="s">
        <v>aaa: 4447</v>
        <stp/>
        <stp>aaa</stp>
        <tr r="A1" s="1"/>
      </tp>
    </main>
  </volType>
```

```
    <volType type="olapFunctions">
      <main first="xlextdat9 Adventure Works DW Adventure Works">
        <tp t="e">
          <v>#N/A</v>
          <stp>1</stp>
          <tr r="A6" s="1"/>
          <tr r="A9" s="1"/>
          <tr r="A8" s="1"/>
          <tr r="A5" s="1"/>
          <tr r="A4" s="1"/>
          <tr r="A3" s="1"/>
        </tp>
      </main>
    </volType>
  </volTypes>
```

end example]

While RTD and Cube functions share the cache, there are differences in how the data is interpreted. [*Example*: RTD dependencies, volTypes/volType/main@`first` specifies the ProgId of the RTD server. Whereas for OLAP dependencies, main@`first` indicates the connection name. *end example*]

18.15.1 main (Main)

Represents dependency information for all topics within a volatile dependency type that share the same first string or function argument.

Attributes	Description
first (First String)	Specifies the first string of all topics within this main. This string corresponds to the first argument to the RTD or CUBE function. For RTD functions, this argument represents the progID of the IRTDServer. [*Example*: `<main first="jrtdx.rtd">` *end example*] For CUBE functions, this argument represents the CUBE connection. [*Example*: `<main first="xlextdat9 Adventure Works DW Adventure Works">` *end example*] For more information on RTD and CUBE functions in SpreadsheetML, see §18.17 in Formulas.

Attributes	Description
	The possible values for this attribute are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_VolMain) is located in §A.2. *end note*]

18.15.2 stp (Strings in Subtopic)

Represents all strings in the topic except for the first. An stp is allocated for each additional argument. [*Example*: For the topic {"progid","","foo"}, there would be two STPs: "" and "foo". *end example*]

For Cube functions, value of "1" indicates that all of the related cells with calling cube functions have been refreshed.

The possible values for this element are defined by the ST_Xstring simple type (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this element's content model (ST_Xstring) is located in §A.6.9. *end note*]

18.15.3 tp (Topic)

Represents dependency information for all topics within a volatile dependency type that share the same first string or argument.

For the RTD function, this collection will contain the remaining parameters of the function, and indicate the last known value and data type of that value.

Attributes	Description
t (Type)	Specifies the type of the cell value. This value corresponds to the type of data returned by the RTD or CUBE function. [*Example*: In the following RTD example, the value "aaa: 4447" has a string data type. `<tp t="s">` `  <v>aaa: 4447</v>` `</tp>` *end example*] For Cube functions, this attribute can be ignored when stp value is "1". The possible values for this attribute are defined by the ST_VolValueType simple type (§18.18.91).

[*Note*: The W3C XML Schema definition of this element's content model (CT_VolTopic) is located in §A.2. *end note*]

18.15.4 tr (References)

Represents the reference to a cell that depends on this topic. Each topic can have one or more cells dependencies.

For CUBE functions, each <tr> element contains a cell whose cube function call dependent on the connection in main@first.

Attributes	Description
r (Reference)	Specifies a reference to the cell location. The location is scoped to the sheet specified in s. [*Example*: `<tr r="A6" s="1"/>` *end example*] The possible values for this attribute are defined by the ST_CellRef simple type (§18.18.7).
s (Sheet Id)	Specifies the sheet index. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_VolTopicRef) is located in §A.2. *end note*]

18.15.5 volType (Volatile Dependency Type)

Represents dependency information for a specific type of external data server. There is no limit on the number of external dependencies that can exist for a workbook in SpreadsheetML.

Attributes	Description
type (Type)	Specifies the type of the external dependency. [*Example*: `<volType type="olapFunctions">` *end example*] The possible values for this attribute are defined by the ST_VolDepType simple type (§18.18.90).

[*Note*: The W3C XML Schema definition of this element's content model (CT_VolType) is located in §A.2. *end note*]

18.15.6 volTypes (Volatile Dependency Types)

Represents the collection of external dependencies for a workbook. This element defines the structure of the volatilateDependencies part. There can only be one volatileDependencies part for each workbook. However, the part can contain one or more dependency types.

The volatileDependencies part stores the following information for Real Time Data (RTD) and CUBE functions:

- Cached values
- Parameters used
- Connection and Server names

[*Example*: *Outline of XML Structure*

```
<volTypes xmlns="…">
  <volType type="realTimeData">
  </volType>
  <volType type="olapFunctions">
  </volType>
</volTypes>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_VolTypes) is located in §A.2. *end note*]

18.16 Custom XML Mappings

Custom XML Mappings enable binding of arbitrary XML data structures and arbitrary XML schema definitions to the workbook. Once a DataBinding has been established, then various XML nodes can be mapped to table columns, ranges of cells, or even single cells (for non-repeating attributes and elements). Once an XML Mapping is fully defined, the application is able to import and export XML instance structures according to the schema definition. ISO/IEC 29500 does not require any particular XML schema language.

[*Note:* Some examples of XML schema languages that might be used to implement Custom XML Mappings include:

- W3C XML Schema - http://www.w3.org/XML/Schema
- RELAX NG – ISO/IEC 19757-2
- Schematron – ISO/IEC 19757-3
- NVDL – ISO/IEC 19757-4

end note]

While the original schema or XML definition can reside on disk or at some file location outside the workbook, a copy of the schema is stored in the workbook.

Every time an XML instance or schema is added to the workbook, a new map object is created which ties together the schemas and where the various elements are mapped in the workbook.

[*Example*:

```
<MapInfo SelectionNamespaces="">
  <Schema ID="Schema1">
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:element nillable="true" name="Root">
        <xsd:complexType>
          <xsd:sequence minOccurs="0">
            <xsd:element minOccurs="0" nillable="true" name="EmployeeInfo"
              form="unqualified">
              <xsd:complexType>
                <xsd:sequence minOccurs="0">
                  <xsd:element minOccurs="0" nillable="true" type="xsd:string"
                    name="Name" form="unqualified"></xsd:element>
                  <xsd:element minOccurs="0" nillable="true" type="xsd:date"
                    name="Date" form="unqualified"></xsd:element>
                  <xsd:element minOccurs="0" nillable="true" type="xsd:integer"
                    name="Code" form="unqualified"></xsd:element>
                </xsd:sequence>
              </xsd:complexType></xsd:element>

              <xsd:element minOccurs="0" maxOccurs="unbounded" nillable="true"
                name="ExpenseItem" form="unqualified">
                <xsd:complexType>
                  <xsd:sequence minOccurs="0">
                    <xsd:element minOccurs="0" nillable="true" type="xsd:date"
                      name="Date" form="unqualified"></xsd:element>
                    <xsd:element minOccurs="0" nillable="true" type="xsd:string"
                      name="Description" form="unqualified"></xsd:element>
                    <xsd:element minOccurs="0" nillable="true" type="xsd:double"
                      name="Amount" form="unqualified"></xsd:element>
                  </xsd:sequence>
                </xsd:complexType></xsd:element>
          </xsd:sequence>
```



```
          <xsd:attribute name="Currency" form="unqualified"
            type="xsd:string"></xsd:attribute>
          <xsd:attribute name="Approved" form="unqualified"
            type="xsd:string"></xsd:attribute>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </Schema>
  <Map ID="1" Name="Root_Map" RootElement="Root" SchemaID="Schema1"
    ShowImportExportValidationErrors="false" AutoFit="true" Append="false"
    PreserveSortAFLayout="true" PreserveFormat="true">
    <DataBinding ConnectionID="1" FileBinding="true" DataBindingLoadMode="1"/>
  </Map>
</MapInfo>
```

end example]

For XML mapped into a SpreadsheetML Table there will also be additional information in the SpreadsheetML file which refers back to the XML Map and XPath of the element or attribute mapped. This information is stored in a xmlColumnPr element, under the tableColumn node.

[*Example*:

```
<table xmlns="…" id="1" name="Table1" displayName="Table1" ref="A1:H11"
  tableType="xml" totalsRowShown="0" connectionId="1">
  <tableColumns count="5">
    <tableColumn id="1" uniqueName="Name" name="Name">
      <xmlColumnPr mapId="1" xpath="/Root/EmployeeInfo/Name"
        xmlDataType="string"/>
    </tableColumn>
    <tableColumn id="2" uniqueName="Date" name="Date">
      <xmlColumnPr mapId="1" xpath="/Root/EmployeeInfo/Date"
        xmlDataType="date"/>
    </tableColumn>
    <tableColumn id="3" uniqueName="Code" name="Code">
      <xmlColumnPr mapId="1" xpath="/Root/EmployeeInfo/Code"
        xmlDataType="integer"/>
    </tableColumn>
    <tableColumn id="4" uniqueName="Description" name="Description">
      <xmlColumnPr mapId="1" xpath="/Root/ExpenseItem/Description"
        xmlDataType="string"/>
    </tableColumn>
```

```
      <tableColumn id="5" uniqueName="Amount" name="Amount">
        <xmlColumnPr mapId="1" xpath="/Root/ExpenseItem/Amount"
          xmlDataType="double"/>
      </tableColumn>
    </tableColumns>
    <tableStyleInfo name="TableStyleMedium9" showFirstColumn="0"
      showLastColumn="0" showRowStripes="1" showColumnStripes="0"/>
  </table>
```

end example]

For XML mapped into a single SpreadsheetML cell there will also be additional information in the TableSingleCells part which refers back to the XML Map and XPath of the element or attribute mapped. This information is stored in the xmlPr element under the xmlCellPr node.

[*Example*:

```
  <singleXmlCells xmlns="…">
    <singleXmlCell id="2" name="Table2" displayName="Table2" r="D19"
      connectionId="1">
      <xmlCellPr id="1" uniqueName="Currency">
        <xmlPr mapId="1" xpath="/Root/@Currency" xmlDataType="string"/>
      </xmlCellPr>
    </singleXmlCell>
    <singleXmlCell id="3" name="Table3" displayName="Table3" r="D20"
      connectionId="1">
      <xmlCellPr id="1" uniqueName="Approved">
        <xmlPr mapId="1" xpath="/Root/@Approved" xmlDataType="string"/>
      </xmlCellPr>
    </singleXmlCell
    <singleXmlCell id="4" name="Table4" displayName="Table4" r="D18"
      connectionId="1">
      <xmlCellPr id="1" uniqueName="Name">
        <xmlPr mapId="1" xpath="/Root/EmployeeInfo/Name" xmlDataType="string"/>
      </xmlCellPr>
    </singleXmlCell>
  </singleXmlCells>
```

end example]

18.16.1 DataBinding (XML Mapping)

This element contains properties which specify how the XML mapping should work.

[*Example*:


```
<DataBinding ConnectionID="1" FileBinding="true" DataBindingLoadMode="1"/>
```

end example]

[*Note*: This element is not intended to reintroduce transitional schema into the strict conformance class. *end note*]

Attributes	Description
ConnectionID (Reference to Connection ID)	Specifies the Connection ID to the external connection in the External Data Connections part. Required if FileBinding is true. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
DataBindingLoadMode (XML Data Loading Behavior)	Specifies the mode for loading XML data related to this DataBinding. Supported values are as follows: 0 - None 1 - Normal 2 - Delay Load 3 - Asynchronous 4 - Object Model The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
DataBindingName (Unique Identifer)	Specifies the data binding name. These shall be unique for each DataBinding. [*Example*: `<DataBinding DataBindingName="Binding1" FileBinding="true" FileBindingName="Binding1" DataBindingLoadMode="1"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema string datatype.
FileBinding (Binding to External File)	Specifies whether the data should be retrieved directly from an XML file. The path to the file is in the corresponding connection element The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
FileBindingName (File Binding Name)	Specifies the file binding name. These shall be unique for each DataBinding. [*Example*: `<DataBinding DataBindingName="Binding1" FileBinding="true" FileBindingName="Binding1" DataBindingLoadMode="1"/>` *end example*]

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_DataBinding) is located in §A.2. *end note*]

18.16.2 Map (XML Mapping Properties)

This element contains all of the properties related to the XML map, and the behaviors expected during data refresh operations.

[*Example*:

```
<Map ID="1" Name="Root_Map" RootElement="Root" SchemaID="Schema1"
  ShowImportExportValidationErrors="false" AutoFit="true" Append="false"
  PreserveSortAFLayout="true" PreserveFormat="true">
  <DataBinding ConnectionID="1" FileBinding="true" DataBindingLoadMode="1"/>
</Map>
```

end example]

Attributes	Description
Append (Append Data to Table)	Specifies whether XML data should overwrite or be appended to the end of the table or range of mapped cells when data is refreshed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
AutoFit (AutoFit Table on Refresh)	Specifies whether columns should be resized to fit the XML data after a data refresh operation. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ID (XML Mapping ID)	Specifies the ID of the XML map. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
Name (XML Mapping Name)	Specifies the name of the XML map. The possible values for this attribute are defined by the W3C XML Schema string datatype.
PreserveFormat (Preserve Cell	Specifies whether cell number formatting in the sheet should be preserved during data refresh operations, or whether the number formatting defined by the XML data type

Attributes	Description
Formatting)	should be used. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
PreserveSortAFLayout (Preserve AutoFilter State)	Specifies whether to keep the filter state of the Table or cell range intact during a data refresh. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
RootElement (Root Element Name)	Specifies the names of the root XML element. The possible values for this attribute are defined by the W3C XML Schema string datatype.
SchemaID (Schema Name)	Specifies the unique name of the schema used for the mapping. The possible values for this attribute are defined by the W3C XML Schema string datatype.
ShowImportExportValidationErrors (Show Validation Errors)	Specifies whether XML schema validation errors should be displayed during data refresh or data export. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Map) is located in §A.2. *end note*]

18.16.3 MapInfo (XML Mapping)

This element acts as the container for all of the XML schemas and maps attached to the SpreadsheetML document.

Attributes	Description
SelectionNamespaces (Prefix Mappings for XPath Expressions)	Specifies namespaces for use in XPath expressions when it is necessary to define new namespaces externally. Namespaces are defined in the XML style, as a space-separated list of namespace declaration attributes [*Example*: The following example contains elements that belong to "a" and "b", in addition to elements that do not belong to any namespace. `<?xml version="1.0"?>` `  <root>` `    <branch>branch</branch>` `    <a:root xmlns:a="http://myserver.com">` `      <a:branch>a-branch</a:branch>` `      <b:branch xmlns:b="http://yourserver.com">`

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td><pre>   b-branch</b:branch>
    </a:root>
  </root></pre>end example]

This is used when writing Xpath expressions at runtime against the XML instance structures, because the Xpath expressions use namespace prefixes instead of the fully spelled out namespace.

The possible values for this attribute are defined by the W3C XML Schema string datatype.</td></tr>
</table>

[*Note*: The W3C XML Schema definition of this element's content model (CT_MapInfo) is located in §A.2. *end note*]

18.16.4 Schema (XML Schema)

This element contains the XML tree for an attached schema.

[*Note*: This element is not intended to reintroduce transitional schema into the strict conformance class. *end note*]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>ID (Schema ID)</td><td>Specifies the unique name or ID for this attached schema.
[Example:<pre> ID = "Schema1"</pre>end example]

The possible values for this attribute are defined by the W3C XML Schema string datatype.</td></tr>
<tr><td>Namespace (Schema Root Namespace)</td><td>Specifies the namespace used by the schema.

[Example:<pre> <MapInfo SelectionNamespaces="…">
    <Schema ID="Schema1" Namespace="…"></pre>end example]

The possible values for this attribute are defined by the W3C XML Schema string datatype.</td></tr>
<tr><td>SchemaLanguage (Schema Language)</td><td>Specifies the media type of the schema language.

[Example:</td></tr>
</table>

Attributes	Description
	`<Schema … SchemaLanguage="application/relax-ng-compact-syntax"/>` *end example*] The possible values for this attribute are defined by the W3C XML Schema token datatype.
SchemaRef (Schema Reference)	The schemaRef attribute is used in the specific case where the schema definition happens to include another schema file that contributes to the same namespace. The value of this attribute is the relative path to a "root" schema file on disk which in turn references the other schema files contributing type definitions to the same namespace. The possible values for this attribute are defined by the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Schema) is located in §A.2. *end note*]

18.17 Formulas

18.17.1 Introduction

A SpreadsheetML *formula* is the syntactic representation of a series of calculations that is parsed or interpreted by the spreadsheet application into a function that calculates a value or array of values based upon zero-to-many inputs.

A formula is an expression that can contain the following: constants, operators, cell references, calls to functions, and names.

[*Example*: Consider the formula `PI()*(A2^2)`. In this case,

- `PI()` results in a call to the function `PI`, which returns the value of π.
- The cell reference A2 returns the value in that cell.
- 2 is a numeric constant.
- The caret (^) operator raises its left operand to the power of its right operand.
- The parentheses, `(` and `)`, are used for grouping.
- The asterisk (*) operator performs multiplication of its two operands.

end example]

18.17.2 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 an 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 are modified by the context-sensitive rules stated in other subclauses of §18.17.*, as indicated by an EBNF comment of the form "(* see semantic rules at 18.17… *)". [*Note*: Thus, in order to produce an automated verifier, the context-sensitive rules in subclauses 18.17.* must all be considered. The context-free syntax rules shown in this subclause provide an overview to assist comprehension by the reader, but do not represent the entirety of the context-sensitive rules. *end note*]

When used in narrative, production names are set in an italic style, as in *array-constant*, *expression*, and *function-name*.

The syntax of a formula is as follows:

```
formula=
  expression  ;
expression=
  "(",  expression,  ")"  |
  constant  |
  prefix-operator,  expression  |
  expression,  infix-operator,  expression  |
  expression,  postfix-operator  |
  cell-reference  |
  function-call  |
  name  ;
```

where *expression* is an arbitrarily complex expression involving constants (§18.17.2.1), operators (§18.17.2.2), cell references (§18.17.2.3), calls to functions (§18.17.2.4), and names (§18.17.2.5).

A *token* is the minimal lexical element of a formula. The categories of tokens are: constants (except for *array-constant*), operators, cell references, function names, names, and punctuators. The *punctuator*s are:

- Left parenthesis (`(`) and right parenthesis (`)`) used for expression grouping and in a function call.
- Comma (`,`) used in a function call and an *array-constant*.
- Left brace (`{`), right brace (`}`), and semicolon (`;`) used in an *array-constant*.

In a formula, an arbitrary number of space characters (U+`0020`) can precede the first token or follow the final token. An arbitrary number of space characters can separate two adjacent tokens, except that no space characters shall separate a *function-name* from the left parenthesis (`(`) that follows it. Such space characters have no effect on the semantics of a formula; however, such spaces shall be distinguished from the space operator (§18.17.2.2).

All arithmetic terms in an *expression* are numbers that can be represented by the workbook's value space. For any given workbook, the default value space is IEC 60559's double precision. However, an implementation is permitted to override that default value space by use of the characteristics markup (§22.7). [*Example*: In the expression `1/3`, although the operands appear to be integers, they are, in fact, floating-point numbers, and the result is `0.33`…, not `0`, as would result from integer division. *end example*]

As ranges of data are fundamental to spreadsheet calculations, many SpreadsheetML functions are able to take arrays as inputs and to return arrays as outputs. Based on whether the formula is an *array formula* or not, the way in which input ranges are interpreted and output values are understood to relate to cells sharing the formula can mean one of two things.

For an array formula: All range arguments are interpreted to be their full range. If the result of the formula is an array, the values of the array are meant to be returned across all of the cells in the sheet sharing the formula. (When the size of the range for an array formula exceeds in either dimension the size of the returned array, the excess cells take on a value of `#N/A.`)

For a normal (non-array) formula:

- Implicit intersection (§18.17.2.2) is performed on all arguments to functions except for those that allow a range.
- If the formula results in an array, only the first value from the array is returned to the cell.

Implicit intersection is determined as follows: When a range is passed to a function which expects only a single cell, a test is made to discover whether the calling cell intersects that range at any point horizontally or vertically. If it does, the cell at the point of intersection is passed to the function. [*Example*: The formula `ABS(B1:B3)` is entered into A2. Because the ABS function does not expect a range, implicit intersection is performed. A2 intersects `B1:B3` horizontally on row 2, and so the value in B2 is passed into the function. *end example*]

The list of function arguments that allow a range is as follows:

- AND - all arguments
- AREAS – *reference* argument
- AVEDEV - all arguments
- AVERAGE - all arguments
- AVERAGEA - all arguments
- AVERAGEIF - all arguments except for *criteria*
- AVERAGEIFS - all arguments except for *criteria1*, *criteria2*, etc.
- CELL - *reference* argument
- CHITEST - all arguments
- COLUMN - all arguments

- COLUMNS - all arguments
- CORREL - all arguments
- COUNT - all arguments
- COUNTA - all arguments
- COUNTBLANK - all arguments
- COUNTIF - all arguments except *criteria*
- COUNTIFS - all arguments except for *criteria1*, *criteria2*, etc.
- COVAR - all arguments
- DAVERAGE - all arguments
- DCOUNT - all arguments
- DCOUNTA - all arguments
- DEVSQ - all arguments
- DGET - all arguments
- DMAX - all arguments
- DMIN - all arguments
- DPRODUCT - all arguments
- DSTDEV - all arguments
- DSTDEVP - all arguments
- DSUM - all arguments
- DVAR - all arguments
- DVARP - all arguments
- FORECAST - all arguments except for *x*
- FREQUENCY - all arguments
- FTEST - all arguments
- FVSCHEDULE - *schedule* argument
- GCD - all arguments
- GEOMEAN - all arguments
- GROWTH - all arguments
- HARMEAN - all arguments
- HLOOKUP - *table_array* argument
- IMPRODUCT - all arguments
- IMSUM - all arguments
- INDEX - *array* or *reference* argument
- INTERCEPT - all arguments
- LARGE - *array* argument
- LCM - all arguments
- LINEST - *known-xs* and *known-ys* arguments
- LOGEST - *known-xs* and *known-ys* arguments
- LOOKUP - all arguments except *lookup_value*
- MATCH - *lookup_array* argument

- MAX - all arguments
- MAXA - all arguments
- MDETERM – *array* argument
- MEDIAN - all arguments
- MIN - all arguments
- MINA - all arguments
- MINVERSE - all arguments
- MIRR - *values* argument
- MMULT - all arguments
- MODE - all arguments
- MULTINOMIAL - all arguments
- NETWORKDAYS - *holidays* argument
- NETWORKDAYS.INTL - *holidays* argument
- NPV - all arguments except *rate*
- OFFSET - *reference* argument
- OR - all arguments
- PEARSON - all arguments
- PERCENTILE - *array* argument
- PERCENTRANK - *array* argument
- PROB - *x_range* and *prob_range* arguments
- PRODUCT - all arguments
- QUARTILE – *array* argument
- RANK - *ref* argument
- ROW – *reference* argument
- ROWS – *array* argument
- RSQ - all arguments
- SKEW - all arguments
- SLOPE - all arguments
- SMALL - *array* argument
- STDEV - all arguments
- STDEVA - all arguments
- STDEVP - all arguments
- STDEVPA - all arguments
- STEYX - all arguments
- SUBTOTAL - all arguments except *function_num*
- SUM - all arguments
- SUMIF - all arguments except *criteria*
- SUMIFS - all arguments except *criteria1*, *criteria2*, etc.
- SUMPRODUCT - all arguments
- SUMSQ - all arguments

- SUMX2MY2 - all arguments
- SUMX2PY2 - all arguments
- SUMXMY2 - all arguments
- TRANSPOSE – all arguments
- TREND – all arguments except *const-flag*
- TRIMMEAN – *array* argument
- TRUNC – *x* argument
- TTEST – *array-1* and *array-2* arguments
- TYPE – all arguments
- VALUE – all arguments
- VAR – all arguments
- VARA – all arguments
- VARP – all arguments
- VARPA – all arguments
- VLOOKUP – *table-array* argument
- WORKDAY – *holidays* argument
- WORKDAY.INTL – *holidays* argument
- XIRR – all arguments except *guess*
- XNPV – all arguments except *rate*
- ZTEST – *array* argument

[*Example*: Here are some formulas taking array constants and ranges:

- `(B2:B4*C2:C4)+10.5` performs three calculations: `(B2*C2)+10.5`, `(B3*C3)+10.5`, and `(B4*C4)+10.5`.
- `SQRT({1,2,3,4})` returns 1 when entered normally.
- `SQRT({1,2,3,4})` returns 1 when used in an array formula in a single cell, but if the array formula spans four or more contiguous cells in a row, it will return 1, 1.41, 1.73, and 2 in the first four cells, respectively, and #N/A in any additional cells in the horizontal range spanned by the array formula. (For display purposes, the values returned have been truncated to two decimal places.)
- `SUM(SQRT({1,2,3,4}))` returns 6.14 when entered normally; when used in an array formula, the intermediate calculation (in this instance performed by the SQRT function) is performed upon each element of the array.

With `A1:A4` holding the values 1, 2, 3, and 4, respectively:

- `SQRT(A1:A4)` entered normally will do implicit intersection if it is in any of the rows 1–4, and return the SQRT of the number in the same row.
- `SQRT(A1:A4)` returns 1 when used in an array formula in a single cell, since it does not do implicit intersection in this case. If it is used in an array formula spanning multiple contiguous cells in a column, it will return 1, 1.41, 1.73, 2, #N/A, …, respectively, in the cells in its vertical output range. *end example*]

18.17.2.1 Constants

A *constant* represents a fixed value that can be used in the calculation of a formula. A constant has the following form:

```
constant=
  error-constant  |
  logical-constant  |
  numerical-constant  |
  string-constant  |
  array-constant  ;
error-constant=
  "#DIV/0!"  |  "#N/A"  |  "#NAME?"  |  "#NULL!"  |
  "#NUM!"  |  "#REF!"  | "#VALUE!" |  "#GETTING_DATA"  ;
logical-constant=
  "FALSE"  |  "TRUE"  ;
numerical-constant=
  whole-number-part,  [full-stop],  [exponent-part]  |
  full-stop,  fractional-part,  [exponent-part]  |
  whole-number-part,  full-stop,  fractional-part  [exponent-part]  ;
full-stop=
  "."    ;  (* also known as “period” *)
whole-number-part=
  digit-sequence  ;
fractional-part=
  digit-sequence  ;
exponent-part=
  "e" [ sign ] digit-sequence  |
  "E" [ sign ] digit-sequence  ;
sign=
  "+"  |
  "-"  ;
digit-sequence=
  decimal-digit,  {decimal-digit}  ;
decimal-digit=
  "0"  |  "1"  |  "2”  |  "3"  |  "4"  |
  "5”  |  "6"  |  "7"  |  "8”  |  "9"  ;
string-constant=
  double-quote, [string-chars], double-quote  ;
double-quote=
  '"'  ; (* one double-quote character*)
string-chars=
  string-char,  {string-char}  ;
```



```
string-char=
  '""'  |   (* consecutive double-quotes, with no space between them *)
  character  -  double-quote  ;   (* any character except double-quote *)
character=
  as defined by the production Char in the XML 1.0 specification, §2.2.
```

[*Example*: =$A$1/3 divides the value in A1 by the constant value three; =$A$1&"a" appends the constant string "a" to the string representation of the contents of cell A1. *end example*]

To include a double-quote character (`"`) in *string-chars*, precede it with another double-quote character. [*Example*: `"ab""cd"` contains the characters `ab"cd`, and `"""abcd"""` contains the characters `"abcd"`. *end example*]

An *array constant* is a list of one or more constants organized in one or two dimensions, and delimited by braces. An array constant has the following form:

```
array-constant=
  "{",  constant-list-rows,  "}"  ;
constant-list-rows=
  constant-list-row,  {  semicolon,  constant-list-row  }  ;
semicolon=
  ";"    ;
constant-list-row=
  constant,  {  comma,  constant  }  ;
```

An *array-constant* shall not contain

- An *array-constant.*
- Columns or rows of unequal length.

Any *numerical-constant* in an *array-constant* can be preceded immediately by a *prefix-operator*.

The *constant*s in an *array-constant* can have different types.

[G*uidance* An implementation is encouraged to not unnecessarily limit the number of rows and columns in an *array-constant*. *end guidance*]

[*Example*: `{1,3.5,TRUE,"Hello"}` is a 1x4 array of constants.

To represent the values 10, 20, 30, and 40, as a 1x4 array, use `{10,20,30,40}`.

To represent the values 10, 20, 30, and 40 in the first row, and 50, 60, 70, and 80 in the second row, use the following 2x4 array constant: `{10,20,30,40;50,60,70,80}`. *end example*]

error-constant is described in §18.17.3.

Each *constant* has a corresponding type (§18.17.2.6), as follows:

Constant Form	Type
array-constant	array
error-constant	error
logical-constant	logical
numerical-constant	number
string-constant	text

In the context of cell formulas and values in SpreadsheetML, the following definition of precision shall apply:

By default, default representation of precision shall be as defined by the XML schema double type: http://www.w3.org/TR/xmlschema-2/#double. The default is therefore 53-bits of mantissa precision.

An application that uses XML schema double can optionally state the precision in the Additional Characteristics part by writing out the number of bits in the mantissa and exponent.

A compliant consumer shall parse numbers of arbitrary precision without error.

18.17.2.2 Operators

An *operator* is a symbol that specifies the type of operation to perform on one or more operands. There are arithmetic, comparison, text, and reference operators.

```
operator=
  ":"  |  comma  |  space  |  "^"  |  "*"  |  "/"  |  "+"  |  "-"
  |  "&"  |  "="  |  "<>"  |  "<"  |  "<="  |  ">"  |  ">="  |  "%"  ;
infix-operator=
  ":"  |  comma  |  space  |  "^"  |  "*"  |  "/"  |  "+"  |  "-"  |  "&"
  |  "="  |  "<>"  |  "<"  |  "<="  |  ">"  |  ">="  ;
postfix-operator=
  "%"  ;
prefix-operator=
  "-"  ;
```

The *operators* permitted in *expression* are:

Operators			
Family	Operator	Description	Precedence
Reference operators	:	Binary range operator, which takes two cell reference (§18.17.2.3) operands, and results in one reference to the cells inclusive of, and between, those references. [*Example*: `SUM(B5:C15)`, which references 11 cells. *end example*]	highest
	,	Binary union operator, which takes two cell reference (§18.17.2.3) operands, and results	

Operators			
		in one reference to all those, possibly non-contiguous, cells. [*Example*: `SUM((B5:B15,D5:D15)))`, which references 22 cells, 11 from column B, and 11 from column D. The grouping parentheses are necessary to indicate that the comma is an operator rather than a punctuator separating two arguments. *end example*]	
	space	Binary intersection operator, which takes two cell reference (§18.17.2.3) operands, and results in one reference to those, possibly non-contiguous, cells that are common. If the intersection is empty, the result is `#NULL!`. [*Example*: `((B1:C1) (C1:D1))` results in a reference to C1, while `((B1:D1) (B1,D1))` results in a single reference to B1 and D1. *end example*]	
Arithmetic operators	-	Unary minus	
	%	Percentage (unary postfix), which divides its operand by 100. [*Example*: `10.5%`, which results in `0.105`. *end example*]	
	^	Exponentiation	
	*	Multiplication	
	/	Division	
	+	Addition	
	-	Subtraction	
Text operator	&	Text concatenation (Each of the two operands is converted to text, if necessary, before concatenation.)	
Comparison operators	=	Equal-to	lowest
	<>	Not-equal-to	
	<	Less-than	
	<=	Less-than or equal-to	
	>	Greater-than	
	>=	Greater-than-or-equal-to	

expression can contain grouping parentheses to document the default precedence or to override it.

*operator*s in *expression* having the same precedence associate left-to-right.

[*Example*: Given that cell E38 contains the value 4, and cell F38 contains the value 2, the formula

```
((-1+E38^2)*3-F38)/2
```

produces the result 21.5. *end example*]

The comparison operators yield TRUE for true and FALSE for false. An expression with value 0 tests logically false while one with any non-zero numeric value tests true.

For any given operator in an expression, if only one operand is an error value, the result is that error value. If more than one operand has an error value and those error values are the same, the result is that error value. If more than one operand has an error value and those error values are not all the same, as to which of those error values is used as the result is unspecified.

It the semantics of an operator having a given operand are not specified by ISO/IEC 29500, the result is #VALUE!. [*Example*: "abc"+1 results in #VALUE!, and "abc"/0 results in #VALUE! rather than #DIV/0!. *end example*]

18.17.2.3 Cell References

Each set of horizontal cells in a worksheet is a *row*, and each set of vertical cells is a *column*. A cell's row and column combination designates the location of that cell. [G*uidance* An implementation is encouraged to not unnecessarily limit the number of rows and columns in a worksheet. *end guidance*]

A *cell reference* designates one or more cells on the same worksheet. Using references, one can:

- Use data contained in different parts of the same worksheet in a single formula.
- Use the value from a single cell in several formulas.
- Refer to cells on other sheets in the same workbook, and even to other workbooks. (References to cells in other workbooks are called *links*.)

A cell reference has the following form:

```
cell-reference=
  name      |
  [work-sheet-prefix]  A1-reference     |
  [work-sheet-prefix]  A1-reference,  ":",  A1-reference      |
  [work-sheet-prefix]  R1C1-reference    |
  [work-sheet-prefix]  R1C1-reference,  ":",  R1C1-reference    ;
work-sheet-prefix=
  work-sheet-prefix-special  |
  sheet-name,  "!"  |
  sheet-name,  ":",  sheet-name,  "!"  |
  "[",  workbook-name,  "]",  sheet-name,  ":",  sheet-name,  "!"   ;
```



```
work-sheet-prefix-special=
  apostrophe,  sheet-name-special,  apostrophe, "!"  |
  apostrophe,  sheet-name-special,  ":",
                     sheet-name-special,  apostrophe,  "!"  |
  apostrophe,  "[",  workbook-name-special,  "]",
                      sheet-name-special,  apostrophe,  "!"  |
  apostrophe,  "[",  workbook-name-special,  "]",
                      sheet-name-special,  ":",  sheet-name-special,
                      apostrophe,  "!"   ;
workbook-name=
  book-name-characters  ;
book-name-characters=
  book-name-character,  {book-name-character}  ;
book-name-character=
  character  –  (operator  |  apostrophe  |  "["  |  "]"  |  "?")  ;
      (* any character except operator or ', [, ], or ? *)
apostrophe=
  "'"  ;   (* one apostrophe character *)
sheet-name=
  sheet-name-characters   ;
sheet-name-character=
  character  –  (operator  |  apostrophe  |  "["  |  "]"  |  "\"  |  "?")  ;
             (* any character except operator or ', [, ], \, or ? *)
sheet-name-characters=
  sheet-name-character,  {sheet-name-character}  ;
workbook-name-special=
  book-name-start-character-special,
    [ book-name-characters-special ];
book-name-start-character-special=
  character  –  (apostrophe  |  "*"  |  "["  |  "]"  |  ":"  |  "?")  ;
      (* any character, including operator, except ', *, [, ], :, or ? *)
book-name-characters-special=
  book-name-character-special,  {book-name-character-special}  ;
book-name-character-special=
  apostrophe, apostrophe  |
  character  –  (apostrophe  |  "*"  |  "["  |  "]"  |   ":"  |  "?")  ;
        (* any character, including operator, except ', *, [, ], :, or ? *)
sheet-name-special=
  sheet-name-start-character-special,
    [ [sheet-name-characters-special], sheet-name-end-character-special]  ;
```



```
sheet-name-start-character-special=
  character  –  (apostrophe  |  "*"  |  "["  |  "]"  |  "\"  |  ":"  |  "/"
     |  "?")  ;
     (* any character, including operator, except ', *, [, ], \, :, /, or ? *)
sheet-name-end-character-special=
  sheet-name-start-character-special  ;
sheet-name-characters-special=
  sheet-name-character-special,  {sheet-name-character-special}  ;
sheet-name-character-special=
  apostrophe, apostrophe  |
  character  –  (apostrophe  |  "*"  |  "["  |  "]"  |  "\"  |  ":"  |  "/"
     |  "?")  ;
     (* any character, including operator, except ‘ *, [, ], \, :, /, or ? *)
```

A *relative cell reference* is based on the relative position of the cell that contains the formula and the cell to which the reference refers. If the position of the cell that contains the formula changes, the reference is changed along with it.

An *absolute cell reference* always refers to the absolute location of a cell. If the position of the cell that contains the formula changes, the absolute reference remains the same.

A *mixed cell reference* has either an absolute column and relative row, or an absolute row and relative column.

A *link* or *external reference* to a workbook is a reference that specifies the location of the workbook, including file or network path, book name, sheet name, and cell reference. Instead of writing the full file path or network location and workbook name directly in the f (formula) element, in order to make all external references more accessible, the workbook name shall be written in a Relationship part according to the Relationships semantic:

- Type shall be set to http://purl.oclc.org/ooxml/officeDocument/relationships/externalLinkPath,
- Target shall specify the full file path and file name, and
- TargetMode shall be set to `External`.

Additionally, in order to support the possibility that the external workbook is offline or otherwise inaccessible, a cache of the relevant sheet values in the referenced external workbook shall be stored within the referencing workbook according to §18.14.

This Supplementary Workbook Data part shall be the source part of the relationship that points to the external workbook as a target resource, and the Id of this relationship shall be referenced within the markup of the Supplementary Workbook Data part, using the externalBook element (§18.14.7).

The Supplementary Workbook Data part shall also be the target of a relationship whose source is the Workbook part (§18.2). The markup within the Workbook part shall reference this relationship Id using the externalReference element (§18.2.8).

Finally, a 1-based index referencing an externalReference within the collection externalReferences shall be written inline within the formula expression containing the reference to the external workbook. The index in this context shall be enclosed within square brackets, i.e.; left square bracket ([), followed by the index, followed by a right square bracket (]).

In this way, external resource files can more easily be accessed and updated.

[*Example*:

Consider the formula =SUM('C:\[Source.xlsx]Sheet1'!$A$1:$A$3)

This formula is expressed in the f element (formula) like this:

```
<f>SUM([1]Sheet1!$A$1:$A$3)</f>
```

The external reference to another workbook in this case is tokenized to [1]. The value inside the brackets is a 1-based index to the externalReferences collection in the workbook part, referencing a specific externalReference element.

The corresponding content of externalReferences in the workbook part is:

```
<externalReferences>
    <externalReference r:id="rId4"/>
</externalReferences>
```

The workbook part's externalReferences collection indicates that there is an external workbook reference in this workbook. The Supplementary Workbook Data cache, also stored in this workbook, can be found by following the relationship from the workbook whose id value is rId4.

That particular relationship (rId4) is expressed as:

```
<Relationship Id="rId4" Type=
"http://purl.oclc.org/ooxml/officeDocument/relationships/externalLinkPath"
Target="externalLinks/externalLink1.xml"/>
```

The above relationship indicates that the formula is supported by the Supplementary Workbook Data cache located at externalLinks/externalLink1.xml in the package.

The corresponding content in externalLink1.xml follows the markup specified in §18.14, Supplementary Workbook Data. The externalBook element in that markup indicates the id of the relationship that points from the source part externalLink1.xml to the location of the actual external workbook:

```
<externalBook xmlns:r="http://purl.oclc.org/ooxml/officeDocument/relationships"
r:id="rId1">
```

That relationship (rId1) is shown here:


```
<Relationship Id="rId1" Type=
"http://purl.oclc.org/ooxml/officeDocument/relationships/externalLinkPath"
Target="file:///C:/Source.xlsx" TargetMode="External"/>
```

This relationship indicates that the external workbook that the formula references resides on a local drive, at c:\source.xlsx. *end example*]

It is possible to process the same cell or set of cells on multiple worksheets within a workbook, using a *3-D reference*. A reference of this type is made up of the cell reference, preceded by a range of worksheet names, and an exclamation mark character (`!`), in that order. A 3-D reference can be used to refer to cells on other sheets, to defined names, and to create formulas by using the following functions: AVERAGE, AVERAGEA, COUNT, COUNTA, MAX, MAXA, MIN, MINA, PRODUCT, STDEV, STDEVA, STDEVP, STDEVPA, SUM, VAR, VARA, VARP, and VARPA.

3-D references shall not be used in array formulas.

By default, a cell reference is understood to refer to one or more cells in the current worksheet. However, a cell reference can be preceded by its parent worksheet name and an exclamation mark (`!`), in that order. This allows cells in one worksheet to be referenced in another worksheet of the same workbook. [*Example*: The cell reference `MonthlyTotals!D1:D12` might be used from within a sibling (or the same) worksheet of `MonthlyTotals` to refer to those 12 cells. *end example*]

An *area* is a set of rectangular-shaped contiguous cells. An area can be a single cell. [*Example*: `A5` and `B6:C10` each designate one area, and `D3:D5,E12:F15` designates two areas (the comma (`,`) being the union operator). *end example*] [*Note*: The number of areas designated by a cell reference can be obtained by calling the function AREAS (§18.17.7.10). *end note*]

There are two cell reference styles: A1 (§18.17.2.3.1) and R1C1 (§18.17.2.3.2).

18.17.2.3.1 A1-Style Cell References

A cell reference using the A1 reference style has the following form:

```
A1-reference=
  A1-column,  ":",  A1-column    |
  A1-row,  ":", A1-row    |
  A1-column,  A1-row  ;
A1-column=
  A1-relative-column  |
  A1-absolute-column  ;
A1-relative-column=
  letter, {letter}  ;  (* see semantic rules at 3.17.2.3.1 *)
```



```
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" ;
A1-absolute-column=
  "$", A1-relative-column  ;
A1-row=
  A1-relative-row  |
  A1-absolute-row  ;
A1-relative-row=
  digit-sequence  ;  (* shall be non-zero; see 3.17.2.3.1 *)
A1-absolute-row=
  “$”, A1-relative-row  ;
```

In this style, each row has a numeric heading numbered sequentially from the top down, starting at 1. Each column has an alphabetic heading named sequentially from left-to-right, A–Z, then AA–AZ, BA–BZ, …, ZA–ZZ, AAA–AAZ, ABA–ABZ, and so on. Column letters are not case-sensitive.

A relative reference to a single cell is written as its column letter immediately followed by its row number. A relative reference to a whole row is written as its row number. A relative reference to a whole column is written as its column letter. A reference to a range of two or more cells is written as two single-cell references separated by the binary range operator (`:`). An absolute A1 reference is made up of a cell's column letter followed by its row number, with each being preceded by a dollar character ($). [*Example*: A2, B34, and `B5:D8` are relative A1 references. `$A$2`, `$B$34`, and `$B$5:$D$8` are absolute A1 references. `$A2`, `B$34`, and `$B5:D$8` are mixed A1 references. *end example*]

[*Example*: `SUM(Sheet2:Sheet13!B5)` adds all the values contained in cell B5 on all the worksheets between and including Sheet2 and Sheet13. *end example*]

For rules on how deal with potential ambiguities between cell references and defined names, see §18.17.5.1.

18.17.2.3.2 R1C1-Style Cell Reference

A cell reference using the R1C1 reference style has the following form:

```
R1C1-reference=
  R1C1-row-only    |
  R1C1-column-only    |
  R1C1-row,  R1C1-column  ;
R1C1-row-only=
  "R",  R1C1-absolute-number,    |
  "R[",  R1C1-relative-number, "]"  ;
```



```
R1C1-row=
  R1C1-relative-row  |
  R1C1-absolute-row  ;
R1C1-relative-row=
  "R[",  R1C1-relative-number, "]"  ;
R1C1-absolute-row=
  "R"    |
  "R", R1C1-absolute-number  ;
R1C1-column-only=
  "C",  R1C1-abolute-number  |
  "C[",  R1C1-relative-number,  "]"  ;
R1C1-column=
  R1C1-relative-column  |
  R1C1-absolute-column  ;
R1C1-relative-column=
  "C[",  R1C1-relative-number,  "]"  ;
R1C1-absolute-column=
  "C"    |
  "C", R1C1-absolute-number  ;
R1C1-relative-number=
  ["-"], digit-sequence   ;
R1C1-absolute-number=
  digit-sequence  ;  (* shall be non-zero; see 3.17.2.3.2 *)
```

In this style, each row has a numeric heading numbered sequentially from the top down, starting at 1. Each column has a numeric heading numbered sequentially from left-to-right, starting at 1.

A whole row is referenced by omitting the column, and a whole column is referenced by omitting the row. An absolute row or column reference uses absolute row or column numbers, respectively. A relative row or column reference uses, respectively, row or column offsets from the cell containing the formula, with a negative offset indicating a row to the left or a column above, and a positive offset indicateing a row to the right or a column below. Specifying an offset of zero is equivalent to omitting that offset and its delimiting brackets. [*Example*: `R[-2]C` refers to the cell two rows up and in the same column, `R[2]C[2]` refers to the cell two rows down and two columns to the right, R2C2 refers to the cell in the second row and in the second column, `R[-1]` refers to the entire row above the active cell, and R refers to the current row. *end example*]

The R1C1 alternate reference style can only be used at runtime. See §18.17.6.1 for XML-related details.

18.17.2.4 Functions

A *function* is a named formula that takes zero or more arguments, performs an operation, and, optionally, returns a result. A function call has the following form:

```
function-call=
  function-name,  "(",  [argument-list],  ")"    ;
```



```
function-name=
  prefixed-function-name       |
  predefined-function-name     |
  user-defined-function-name     ;
predefined-function-name=
  "ABS"  |  "ACOS"  |  "ACOSH"
     |  ( any of the other functions defined in §18.17.7)  ;
prefixed-function-name=
  "ISO.", predefined-function-name |
  "ECMA.", predefined-function-name
user-defined-function-name=
  letter,  [ user-defined-name-characters ]    ;
user-defined-name-characters=
  user-defined-name-character,  {user-defined-name-character}  ;
user-defined-name-character=
  letter  |  decimal-digit  |  full-stop    ;
argument-list=
  argument,  {  comma,  argument  }  ;
comma=
  ","  ;
argument=
  expression  ;
```

*predefined-function-name*s and *user-defined-function-name*s are not case-sensitive.

A *user-defined-function-name* shall not have any of the following forms:

- TRUE or FALSE
- *name*
- *cell-reference*

[*Guidance*: An implementation is encouraged to support *user-defined-function-name*s at least as long as 255 characters. *end guidance*]

The semantics of a call to a function having a *user-defined-function-name* are unspecified.

[*Example*: Here are some function calls: `PI(), POWER(A1,B3)`, and `SUM(C6:C10)`. *end example*]

An argument to a function can be a call to a function. That is, function calls can nest. [G*uidance* An implementation is encouraged to support at least 64 levels of nested function calls. *end guidance*]

Some functions take a variable number of arguments. This is indicated in the Syntax sections of §18.17.7 by their having *argument-list* as all, or the trailing part, of their argument list. The total number of arguments that shall be passed to such functions is at least 1.

[*Guidance* An implementation is encouraged to support function calls having at least 255 arguments. *end guidance*]

Expressions can have one or more values. Scalar expressions designate a single value, and cell references and array constants can designate multiple values. In the case of a multi-value expression, the way in which this is handled by a function when passed as an argument depends on a number of factors.

Most functions and operators expect either single- or multi-valued arguments and perform all of the array calculations whenever multi-valued arguments are present. [*Example*: `SQRT({1;2;3;4})`; see the examples in §18.17.2. *end example*]

When a function expects a single-valued argument but a multi-valued expression is passed, an attempt can be made to convert that set of values to a single value. For an array value or constant, the value of the expression is the value of the first element within that array value or constant. For a cell range, the first element can be used, or implicit intersection can be performed—the exact behavior is unspecified.

When a function expects a multi-valued argument but a single-valued expression is passed, that single-valued argument is treated as a 1x1 array.

For rules on how deal with potential ambiguities between function names and defined names, see §18.17.5.1.

18.17.2.5 Names

A *name* is an alias for a constant, a cell reference, or a formula. [*Note*: A name in a formula can make it easier to understand the purpose of that formula. For example, the formula `SUM(FirstQuarterSales)` is easier to identify than `SUM(C20:C30)`. *end note*]

Here is the syntax for *name*:

```
name=
  [ workbook-name, "!" ], name-start-character, [ name-characters ]  ;
name-start-character=
  letter  |  underscore  |  backslash  ;
underscore=
  "_"  ;
backslash=
  "\"  ;
name-characters=
  name-character,  {name-character}  ;
name-character=
  letter  |  decimal-digit  |  underscore  |  full-stop  ;
```

*name*s are not case-sensitive.

All *name*s within a workbook shall be unique. If the same *name*s are defined in two workbooks, both *name*s can be used in the same context be prefixing them with their corresponding workbook name and an exclamation mark (`!`). [*Example*: `SUM(Sales.xlsx!ProjectedSales)` refers to the named range `ProjectedSales` in the workbook named `Sales.xlsx`. *end example*]

A *name* shall not have any of the following forms:

- TRUE or FALSE
- *user-defined-function-name*
- *cell-reference*

[*Guidance* An implementation is encouraged to support *name*s at least as long as 255 characters. *end guidance*]

For rules on how deal with potential ambiguities between function names and defined names, or between cell references and defined names, see §18.17.5.1.

18.17.2.6 Types and Values

Each *expression* has a type. SpreadsheetML formulas support the following types: array, error, logical, number, and text.

An array value or constant represents a collection of one or more elements, whose values can have any type (i.e., the elements of an array need not all have the same type).

An error value (§18.17.3) or constant represents an error, and can have any value defined for *error-constant* (§18.17.2.1).

A logical value or constant represents a truth value, and can have any value defined for *logical-constant* (§18.17.2.1).

A numeric value or constant represents a real number, and can have any value defined for *numeric-constant* (§18.17.2.1). The term "number" is used as a generic name for any expression of type *number*.

A text value or constant represents arbitrary text, and can have any value defined for *string-constant* (§18.17.2.1). The term "string" is used as a generic name for any expression of type *text*.

An implementation is permitted to provide an implicit conversion from *string-constant* to number. However, the rules by which such conversions take place are implementation-defined. [*Example*: An implementation might choose to accept "123"+10 by converting the string "123" to the number 123. Such conversions might be locale-specific in that a *string-constant* such as "10,56" might be converted to 10.56 in some locales, but not in others, depending on the radix point character. *end example*]

[*Guidance* An implementation is encouraged to support strings at least as long as 32,767 characters. *end guidance*]

A complex number is represented as a string in one of two equivalent text formats: *x* + *y*`i` or *x* + *y*`j`, where *x* is the real part, and *y* is the imaginary part. [*Example*: `"3+4i"` and `"-2.5-34.6j"` *end example*]

18.17.2.7 Single- and Array Formulas

A formula can either be a single-cell formula, or an array formula.

A single-cell formula is applied to a single cell while an array formula is applied to a range of cells as a group.

When a single-cell formula results in a single value, the designated cell takes on that value. [*Example*: When cell `A10` contains `SIN(0.3)`, the result stored in that cell is `0.295520207`. *end example*]

When a single-cell formula results in multiple values, the designated cell takes on the first of those values. [*Example*: When cell A10 contains `SIN({0.3,0.4,0.5})`, the result stored in that cell is 0.295520207 (SIN(0.3)). *end example*]

Array formulas are an extension of the formula paradigm. They allow for combining of several formula operations into one, and the returning of multiple results. Array formulas span one or more cells. There are three primary functions that array formulas perform:

2. Returning multiple results from one single function call. [*Example*: the function `LINEST` returns an array of several results, and, therefore, ought to be spanned across several cells as an array formula in order to display those results. *end example*]
3. Executing one formula several times to generate an array of results. [*Example*: The array formula `SIN(A1:A3)` returns an array with three elements, those being the sines of A1, A2 and A3, respectively. *end example*]
4. Returning one single result from an operation that incorporated arrays as an intermediate step. [*Example*: The array formula `SUM(SIN(A1:A3))` returns a single result, that being the SUM of the sines of A1, A2 and A3, respectively. *end example*]

When an array formula results in a single value, all of the cells spanned by the array formula take on that value. [*Example*: When the group of cells `A10:A12` contains `SIN(0.3)`, the result stored in each of those cells is `0.295520207`. *end example*]

When an array formula results in multiple values, the designated cells take on corresponding values, according to the shape of the cell group and the values. Specifically,

- If the cell group and values have the same shape (i.e., the same number of rows and columns), each cell takes on the value corresponding to its relative position.
- If the cell group has fewer columns than the values, the left-most columns of the values are stored in the cells.
- If the cell group has fewer rows than the values, the top-most rows of the values are stored in the cells.
- If the cell group has more columns than the values, each cell takes on the value corresponding to its relative position, except that
  - For a cell group 1x*N* array or a two-dimensional array, the excess right-most cells take on an unspecified value.
  - For a cell group *N*x1 array, the excess columns are clones of the first column.

- If the cell group has more rows than the values, each cell takes on the value corresponding to its relative position, except that:
  - For a cell group *N*x1 array or a two-dimensional array, the excess bottom-most cells take on an unspecified value.
  - For a cell group 1x*N* array, the excess rows are clones of the first row.

[*Example*: Case 1: The 1x3 group of cells `A20:C20` has applied to it the array formula `SIN({0.3,0.4,0.5})`. The number of rows and columns in the group exactly matches the number of rows and columns in the result. Those cells then contain 0.295520207, 0.389418342, and 0.479425539, which correspond to `SIN(0.3)`, `SIN(0.4)`, and `SIN(0.5)`, respectively.

Case 2: The 1x2 group of cells `A20:B20` has applied to it the array formula `SIN({0.3,0.4,0.5})`. The number of columns in the group is less than the number of columns in the result. (The number of rows is the same in each.) Those cells then contain 0.295520207 and 0.389418342, which correspond to `SIN(0.3)` and `SIN(0.4)`, respectively, the left-most part of the set of values.

Case 3: The 1x4 group of cells `A20:D20` has applied to it the array formula `SIN({0.3,0.4,0.5})`. The number of columns in the group is greater than the number of columns in the result. (The number of rows is the same in each.) Those cells then contain 0.295520207, 0.389418342, 0.479425539, and an unspecified value, which correspond to `SIN(0.3)`, `SIN(0.4)`, and `SIN(0.5)`, respectively, with the fourth value being unspecified.

Case 4: The 2x2 group of cells `A30:B31` has applied to it the array formula `SIN({0.1,0.2,0.3})`. The number of columns in the group is less than the number of columns in the result. As a result, the cells in row 30 contain 0.295520207 and 0.389418342, which correspond to `SIN(0.3)` and `SIN(0.4)`, respectively. The number of rows in the group is greater than the number of rows in the result, so the cells in 31 are a copy of the cells in row 30. The left-most part of the set of values is propagated into the cells.

Case 5: The 2x2 group of cells `A40:B41` has applied to it the array formula `SIN({0.1,0.2,0.3; 0.4,0.5,0.6; 0.7,0.8,0.9})`. The number of columns in the group is less than the number of columns in the result. As a result, the left-most column values are stored. The number of rows in the group is less than the number of rows in the result. As a result, the top-most column values are stored. *end example*]

18.17.3 Error values

The evaluation of an expression can result in an error having one of a number of *error value*s. These error values are:

Error Value	Reason for Occurrence
`#DIV/0!`	Intended to indicate when any number (including zero) or any error code is divided by zero.
#GETTING_DATA	Intended to indicate when a cell reference cannot be evaluated because the value for the cell has not been retrieved or calculated. [*Note*: This can happen when connected to an OLAP cube. *end note*]

Error Value	Reason for Occurrence
	This error constant differs from #N/A in that #GETTING_DATA is used when there is an expectation that the value for the cell will eventually be available, whereas #N/A is used when there is no such expectation.
#N/A	Intended to indicate when a designated value is not available. [*Example*: Some functions, such as SUMX2MY2, perform a series of operations on corresponding elements in two arrays. If those arrays do not have the same number of elements, then for some elements in the longer array, there are no corresponding elements in the shorter one; that is, one or more values in the shorter array are not available. *end example*] This error value can be produced by calling the function NA (§18.17.7.223).
#NAME?	Intended to indicate when what looks like a name is used, but no such name has been defined. [*Example*: XYZ/3, where XYZ is not a defined name. Total is & A10, where neither Total nor is is a defined name. Presumably, "Total is " & A10 was intended. SUM(A1C10), where the range A1:C10 was intended. *end example*]
#NULL!	Intended to indicate when two areas are required to intersect, but do not. [*Example*: In the case of SUM(B1 C1), the space between B1 and C1 is treated as the binary intersection operator, when a comma was intended. *end example*]
#NUM!	Intended to indicate when an argument to a function has a compatible type, but has a value that is outside the domain over which that function is defined. (This is known as a *domain error*.) [*Example*: Certain calls to ASIN, ATANH, FACT, and SQRT might result in domain errors. *end example*] Intended to indicate that the result of a function cannot be represented in a value of the specified type, typically due to extreme magnitude. (This is known as a *range error*.) [*Example*: FACT(1000) might result in a range error. *end example*]
#REF!	Intended to indicate when a cell reference cannot be evaluated. [*Example*: If a formula contains a reference to a cell, and then the row or column containing that cell is deleted, a #REF! error results. If a worksheet does not support 20,001 columns, OFFSET(A1,0,20000) will result in a #REF! error. *end example*]
#VALUE!	Intended to indicate when an incompatible type argument is passed to a function, or an incompatible type operand is used with an operator. [*Example*: In the case of a function argument, text was expected, but a number was provided *end example*]

Each error value has a corresponding *error-constant* (§18.17.2.1).

[*Note*: A number of functions operate on error values: They include ERROR.TYPE (§18.17.7.110), ISERR (§18.17.7.175), ISERROR (§18.17.7.176), and ISNA (§18.17.7.179). *end note*]

18.17.4 Dates and Times

Dates and times in cells in SpreadsheetML are stored as strings, using the ISO 8601 lexical formats defined below.

The earliest date permitted is 0001-01-01, 00:00. The latest date permitted is 9999-12-31, 23:59:59.999. The time midnight shall be expressed always with hour component 0 and not with hour component 24. Leap seconds are not permitted – the maximum number of seconds expressed in a minute shall be 60.

Values with only a date component shall be expressed using the Complete, Extended Format Calendar Date representation, as defined in ISO 8601, §B.1.1 and §B2.1.

[*Example*: The date 5 October 1975 is expressed in SpreadsheetML as

1975-10-05

end example]

Values with only a time-of-day component shall be expressed using the Complete, Extended Format Time Of Day representation, as defined in ISO 8601, §B.1.2 and §B2.2. The decimal separator shall be a full stop (period), and fractional seconds shall be expressed with no more than three decimal places.

[*Example*: The time-of-day 08:30 can be expressed in the following ways within SpreadsheetML:

08:30
08:30:00
08:30:00.000
end example]

Values with both date and time-of-day components shall be expressed using the Complete, Extended Format Calendar Date and Time Of Day representation, as defined in ISO 8601, §B.1.3 and §B2.3. For the time component, only seconds may use a decimal separator, the decimal separator shall be a full stop (period) and fractional seconds shall be expressed with no more than three decimal places.

[*Example*: The date 22 November 1976 at local time 08:30 can be expressed in the following ways within SpreadsheetML:

1976-11-22T08:30
1976-11-22T08:30:00
1976-11-22T08:30:00.000

The date 15 October 1582—the day the Gregorian calendar went into effect for some countries—can be expressed in the following ways:

1582-10-15
1582-10-15T00:00

1582-10-15T00:00:00
1582-10-15T00:00:00.000

end example]

[*Note:* SpreadsheetML relates all dates to the proleptic Gregorian calendar of ISO 8601, treating time periods extending into the past and into the distant future as if the Gregorian calendar is in effect for all of those days. January 1 is always the first day of each year, ignoring historical changes to the period of the calendar year. The gaps and shifts introduced as part of calendar reforms and for introduction of leap seconds are ignored under the proleptic Gregorian calendar system. *end note*]

Wherever a calculation in a formula is specified to apply to number values and a date or time is provided, the effect shall be the same as if the date and/or time value is converted to the corresponding serial date-time. Wherever a calculation in a formula is specified to apply to or to deliver a date and/or time value, and a number value is supplied, the number value is interpreted as a serial date-time for the date and/or time. The relationships between serial date-times and dates and times are specified in §18.17.4.1, §18.17.4.2, and §18.17.4.3.

18.17.4.1 Date Conversion for Serial Date-Times

A serial date-time is a number that represents a date and time. This signed value is in units of days relative to the base date for the selected date system. Serial date-times increase by 1 into each successive day and decrease by 1 into each preceding day. Fractional portions of serial date-times represent fractions of a single day. [*Example*: When using the 1900 date system, which has a base date of 30th December 1899, a serial date-time of 1.5 represents midday on the 31st December 1899 (serial date-time day 1), or 1899-12-31T12:00. A serial date-time of -4.25 represents 6 pm on the 25th December 1899, or 1899-12-25T18:00. *end example*] The base dates and the related serial date-times represent local date and time.

Two different bases are used for converting dates to and from serial date-times:

- In the *1900 date system*, the lower limit is January 1st, 0001 00:00:00, which has a serial date-time of -693593. The upper-limit is December 31st, 9999, 23:59:59.999, which has a serial date-time of 2,958,465.9999884. The base date for this system is 00:00:00 on December 30th, 1899, which has a serial date-time of 0.
- In the *1904 date system*, the lower limit is January 1st, 0001, 00:00:00, which has a serial date-time of -695055. The upper limit is December 31st, 9999, 23:59:59.999, which has a serial date-time of 2,957,003.9999884. The base date for this system is 00:00:00 on January 1st, 1904, which has a serial date-time of 0.

A serial date-time outside the temporal range for the selected date system is invalid.

The date system is specified by the value of the date1904 attribute of the workbookPr element. [*Example*:

1900 date system: `<workbookPr showObjects="all"/>`
1904 date system: `<workbookPr date1904="1" showObjects="all"/>`

end example]

18.17.4.2 Time Conversion for Serial Date-Times

Time of day is represented by a serial date-time less than 1, but not less than 0. Values from 0–0.99999999 represent times from *the* starting instant 0:00:00 (12:00:00 AM) to the last instant 23:59:59 (11:59:59 P.M.).

For any serial date-time, the serial time-of-day is the serial date-time minus the serial date-time of the day in which the time-of-day occurs. The serial date-time of the day in which a serial date-time occurs is the greatest integer that does not exceed the serial date-time.

[*Example:*

The serial date-time 4.66666667 is in serial day 4, and the time-of-day serial date-time is 4.66666667 – 4, which is 0.66666667.

The serial date-time -2.00000001 is in serial day -3, and the time-of-day serial date-time is -2.00000001 – (-3), which is 0.99999999.

end example]

Going forward in time, the time component of a serial date-time increases by 1/86,400 each second. [*Note*: As such, the time 12:00 has a serial date-time time component of 0.5. *end note*]

[*Example*:

The serial date-time `0.0000000…` represents `00:00:00`
The serial date-time `0.0000115…` represents `00:00:01`
The serial date-time `0.4207639…` represents `10:05:54`
The serial date-time `0.5000000…` represents `12:00:00`
The serial date-time `0.9999884…` represents `23:59:59`

end example]

18.17.4.3 Combined Date and Time Conversion for Serial Date-Times

The serial date-time corresponding to a date component can be added to any serial date-time for a time-of-day component to determine the serial date-time for the combined date-time.

[*Note*: In the 1900 date system, the serial date-time -1.25 represents December 28, 1899, 18:00. *end note*]

[*Example*: For the 1900 date system:

The serial date-time -2337.999989… represents 1893-08-05T00:00:01
The serial date-time `3687.4207639…` represents 1910-02-03T10:05:54
The serial date-time `2.5000000…` represents 1900-01-01T12:00:00
The serial date-time `2958465.9999884…` represents 9999-12-31T23:59:59

For the 1904 date system:

The serial date-time -3799.999989… represents 1893-08-05T00:00:01
The serial date-time `2225.4207639…` represents 1910-02-03T10:05:54
The serial date-time `0.5000000…` represents 1904-01-01T12:00:00
The serial date-time `2957003.9999884…` represents 9999-12-31T23:59:59

end example]

18.17.5 Limits and Precision

18.17.5.1 Limits

In SpreadsheetML, cell references range from column A1–A1048576 (column A:A) to column XFD1–XFD1048576 (column XFD:XFD).

An implementation can extend this range. However, to avoid ambiguities, it is necessary to ensure that defined names are distinct from cell references, or that one takes precedence over the other. With this in mind, the following rules apply:

- A producer or consumer shall consider a defined name of the form used by cells in the range A1–XFD1048576 to be an error.
- All other names outside this range can be defined names and shall override a cell reference if an ambiguity exists.

[*Example*: `LOG10` is always a cell reference, `LOG10(…)` is always a formula, and `LOGO1000` can be a defined name that overrides a cell reference. *end example*]

18.17.5.2 Precision

In order to clarify the semantics of cell formulas and values in SpreadsheetML, it is necessary to specify the precision of the numbers being represented in the file format. These numbers are therefore regarded as ranging over a specific value space, which defaults to the following:

The *value space* consists of the values (-1)^s × m × 2^n, where s is 0 or 1, m is an integer greater than or equal to 0 and less than 2^53, and n is an integer between -1074 and 971, inclusive. m is herein referred to as the *binary mantissa*, and n is herein referred to as the *binary exponent*. [*Note*: The default precision is patterned after the IEC double-precision 64-bit floating-point type [IEC 60559]. *end note*]

Implementing applications can use the characteristics markup (§22.7) to specify other value spaces to replace the default in a given workbook. When present in the workbook, the value space defined using the characteristics markup overrides the default value space.

Regardless of the specific value space in use, values shall have a lexical representation as described in §18.17.5.3. Any numerical expression conforming to this lexical description is permitted. However, numbers of higher precision than available in the value space, and numbers that lie outside the range representable in the value space shall be handled as prescribed in §18.17.5.4.

18.17.5.3 Lexical Representation

The value space shall have a lexical representation consisting of a base 10 mantissa followed, optionally, by the character "E" or "e", followed by a base 10 exponent. The exponent shall be an integer. The mantissa shall be a decimal number. The representations for exponent and mantissa shall follow the lexical rules for integer and decimal below. If the "E" or "e" and the following exponent are omitted, an exponent value of 0 is assumed.

Lexical representations for zero can take a positive or negative sign.

[*Example*: -1E4, 1267.43233E12, 12.78e-2, 12 , -0, and 0 are all literals for numbers in the default value space. 4503599627370497.5 is also a literal, although it represents the same value as 4503599627370497 (2^52 + 1) in the default value space (as explained in §18.17.5.4). *end example*]

An *Integer* has a lexical representation consisting of a finite-length sequence of decimal digits (#x30–#x39) with an optional leading sign. If the sign is omitted, "+" is assumed. [*Example*: -1, 0, 12678967543233, +100000. *end example*]

A *Decimal Number* has a lexical representation consisting of a finite-length sequence of decimal digits (#x30–#x39) separated by a period as a decimal indicator. An optional leading sign is allowed. If the sign is omitted, "+" is assumed. Leading and trailing zeroes are optional. If the fractional part is zero, the period and following zero(s) can be omitted. [*Example*: -1.23, 12678967.543233, +100000.00, 210. *end example*]

18.17.5.4 Interpretation

Strings that are permitted according to the lexical definition in §18.17.5.3 shall be interpreted as values in the value space as follows:

1. The mantissa shall be interpreted as a real number expressed in base 10
2. The exponent shall be interpreted as an integer expressed in base 10
3. The raw value for a numerical expression shall be interpreted as mantissa x 10^{exponent}
4. If the absolute value is larger than the largest value in the value space (2^1024 minus 2^971) then a consuming application shall treat this as equivalent to the error value `#NUM!` (§18.17.3). Otherwise the value in the value space that is closest to the raw value is chosen as the interpretation. In the case that two values are equally close, the one with the smaller absolute value is chosen.

18.17.6 XML Representation

18.17.6.1 Cell Reference Style

A workbook saved with reference style A1 (§18.17.2.3.1), shall have the refMode attribute of the calcPr element (§18.2.2) in the Workbook part's XML omitted or set to `A1`. A workbook saved with reference style R1C1 (§18.17.2.3.2) shall have that refMode attribute set to `R1C1`. [*Example*: With R1C1 mode set, here is how the XML might look:

```
<workbook …>
  …
  <calcPr calcId="122211" fullCalcOnLoad="1" refMode="R1C1"/>
  …
</workbook>
```

end example]

Regardless of the value of the refMode attribute, cell references shall be stored in XML in the A1 form. This attribute's value tells an implementation which reference style to use at runtime.

18.17.6.2 Scalar Formulas

A scalar formula shall be represented in a worksheet's XML by an f element that contains the text of the formula, and a v element that contains the text version of the last computed value for that formula. This pair of elements shall be inside a c element, which is, in turn, shall be inside a row element. [*Example*: Consider the scalar formula SQRT(C2^2+D2^2), where C2 refers to a cell containing the number 12.5, and D2 refers to a cell containing the number 9.6. The corresponding XML might be as follows:

```
<row r="2" spans="2:4">
  <c r="B2" s="40">
    <f>SQRT(C2^2+D2^2)</f>
    <v>15.761027885261798</v>
  </c>
  <c r="C2" s="0">
    <v>12.5</v>
  </c>
  <c r="D2" s="0">
    <v>9.6</v>
  </c>
</row>
```

In the scalar formula CONCATENATE("The total is ",C7," units"), C7 refers to a cell containing the number 23. The corresponding XML might be as follows:

```
<row r="7" spans="2:4" ht="285">
  <c r="B7" s="4" t="str">
    <f>CONCATENATE("The total is ",C7," units")</f>
    <v>The total is 23 units</v>
  </c>
  <c r="C7" s="0">
    <v>23</v>
  </c>
</row>
```

As the function CONCATENATE returns a string, the value for the cell's t attribute is str.

end example]

18.17.6.3 Array Formulas

An array formula shall be represented in XML just like other formulas, except that the array formula's f element shall contain an attribute t, whose value shall be array.

For a single-cell formula, the r attribute shall designate that cell. [*Example*: Consider the array formula SUM(C11:C12*D11:D12). The corresponding XML might be as follows:

```
<row r="11" spans="2:4" ht="300">
  <c r="B11" s="16">
    <f t="array" r="B11">SUM(C11:C12*D11:D12)</f>
    <v>110</v>
  </c>
  <c r="C11" s="4">
    <v>10</v>
  </c>
  <c r="D11" s="0">
    <v>3</v>
  </c>
</row>
<row r="12" spans="2:4" ht="285">
  <c r="C12" s="4">
    <v>20</v>
  </c>
  <c r="D12" s="0">
    <v>4</v>
  </c>
</row>
```

As this formula is a single-cell formula, the r attribute contains the name of that cell, B11. *end example*]

For an array formula spanning multiple cells, the r attribute of the top-left cell of the range of cells to which that formula applies shall designate the range of cells to which that formula applies. The c elements for all cells except the top-left cell in that range shall not have an f element; however, they shall each have a v element. [*Example*: Consider the array formula A1:A3*B1:B3, which is applied to the cell range C1:C3. The corresponding XML might be as follows:

```
<row r="1" spans="1:3">
  <c r="A1" s="0">
    <v>112</v>
  </c>
  <c r="B1" s="0">
    <v>2.34</v>
  </c>
```



```
    <c r="C1" s="0">
      <f t="array" r="C1:C3">A1:A3*B1:B3</f>
      <v>262.08</v>
    </c>
  </row>
  <row r="2" spans="1:3">
    <c r="A2" s="0">
      <v>209</v>
    </c>
    <c r="B2" s="0">
      <v>1.28</v>
    </c>
    <c r="C2" s="0">
      <v>267.52</v>
    </c>
  </row>
  <row r="3" spans="1:3">
    <c r="A3" s="0">
      <v>128</v>
    </c>
    <c r="B3" s="0">
      <v>3.12</v>
    </c>
    <c r="C3" s="0">
      <v>399.36</v>
    </c>
  </row>
```

As this formula is an array formula spanning multiple cells, the r attribute of cell C1 contains the name of that cell range, `C1:C3`, and cells C2 and C3 do not have an f element. *end example*]

18.17.6.4 Formula Evaluation Order

The order in which formulas are evaluated is determined by the order of their corresponding c elements in the calcChain element of the Calculation Chain part (§18.6.2).

18.17.6.5 Name Representation

A formula can contain one or more names. These names shall be defined in the Workbook part's XML with each being the subject of a definedName element, inside a definedNames element. [*Example*: Consider the scalar formula SUM(`value1,value2`). The corresponding XML might be as follows:

```
<definedNames>
  <definedName name="value1" localSheetId="0">Sheet2!$B$2</definedName>
  <definedName name="value2" localSheetId="0">Sheet2!$B$3</definedName>
</definedNames>
…
```



```
<c r="E5" s="0">
  <f ce="1">SUM(value1,value2)</f>
  <v>8</v>
</c>
```

end example]

Each name shall be the subject of an lpstr element in the Application-Defined File Properties part.

```
<TitlesOfParts>
  <vt:vector … baseType="lpstr">
    <vt:lpstr>Sheet1</vt:lpstr>
    <vt:lpstr>Sheet2</vt:lpstr>
    <vt:lpstr>Sheet3</vt:lpstr>
    <vt:lpstr>value1</vt:lpstr>
    <vt:lpstr>value2</vt:lpstr>
  </vt:vector>
</TitlesOfParts>
```

18.17.6.6 Value Representation

The most recent value of a formula shall be stored in the corresponding v element, as follows:

Result Type	Representation
array	The text form of the array's value.
error	The text form of the error value.
logical	The text `0` for `FALSE` and `1` for `TRUE`.
number	The unformatted text form of the number, as accurately as possible.
text	All of the characters in the text.

18.17.6.7 Dates and Times

When a SpreadsheetML cell contains a date-time, the value of the cell is expressed as a string conforming to one of the ISO 8601 lexical formats specified in §18.17.4.

[*Example*:

```
<c t="d">
  <f>DATE(1582,10,15)+0.5</f>
    <v>1582-10-15T12:00</v>
  </c>
```

end example]

18.17.7 Predefined Function Definitions

Each of the subclauses below this subclause describes a separate function, and each description contains a section marked **Syntax**. That section contains pieces of the function grammar as they pertain to that specific function. 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 function-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 would appear with double-quotes surrounding each instance. *end note*]

The **Syntax** section for each function defined in this subclause corresponds to a call to that function. Except for *argument-list*, the names in any **Syntax** section typeset as in *number* and *string-1*, are parameter names for that function, and are local to that function definition's description. *argument-list* is the name of a production in the grammar, and, as defined in §18.17.2, permits a comma-separated list of *argument*s.

When the type of an argument passed to a function is incompatible with the type expected the error value `#VALUE!` is returned by that function.

The set of predefined functions is divided into the following functional categories [*Note*: The predefined functions defined here reflect current spreadsheet semantics and might not match common practice in other contexts. New functions might be added in future versions of the specification. *end note*]:

All predefined functions can be used with their simple name, with the prefix `ECMA.` or with the prefix `ISO.`, with the following exception: the predefined function named `CEILING` in ISO/IEC 29500 can only be used with the prefix `ECMA.`. The predefined function named `ISO.CEILING` is specified in ISO/IEC 29500.

Category	Formulas
Cube	CUBEKPIMEMBER (§18.17.7.65), CUBEMEMBER (§18.17.7.66), CUBEMEMBERPROPERTY (§18.17.7.67), CUBERANKEDMEMBER (§18.17.7.68), CUBESET (§18.17.7.69), CUBESETCOUNT (§18.17.7.70), CUBEVALUE (§18.17.7.71)
Database	DAVERAGE (§18.17.7.77), DCOUNT (§18.17.7.81), DCOUNTA (§18.17.7.82), DGET (§18.17.7.90), DMAX (§18.17.7.92), DMIN (§18.17.7.93), DPRODUCT (§18.17.7.97), DSTDEV (§18.17.7.98), DSTDEVP (§18.17.7.99), DSUM (§18.17.7.100), DVAR (§18.17.7.102), and DVARP (§18.17.7.103).
Date and Time	DATE (§18.17.7.74), DATEDIF (§18.17.7.75), DATEVALUE (§18.17.7.76), DAY (§18.17.7.78), DAYS360 (§18.17.7.79), EDATE (§18.17.7.105), EOMONTH (§18.17.7.107), HOUR (§18.17.7.144), MINUTE (§18.17.7.214), MONTH (§18.17.7.220), NETWORKDAYS (§18.17.7.226), NETWORKDAYS.INTL (§18.17.7.227), NOW (§18.17.7.234), SECOND (§18.17.7.287), TIME (§18.17.7.323), TIMEVALUE (§18.17.7.324), TODAY (§18.17.7.326), WEEKDAY (§18.17.7.344), WEEKNUM (§18.17.7.345), WORKDAY (§18.17.7.347) , WORKDAY.INTL (§18.17.7.348), YEAR (§18.17.7.351), and YEARFRAC

Category	Formulas
	(§18.17.7.352)
Engineering	BESSELI (§18.17.7.23), BESSELJ (§18.17.7.24), BESSELK (§18.17.7.25), BESSELY (§18.17.7.26), BIN2DEC (§18.17.7.29), BIN2HEX (§18.17.7.30), BIN2OCT (§18.17.7.31), COMPLEX (§18.17.7.45), CONVERT (§18.17.7.48), DEC2BIN (§18.17.7.84), DEC2HEX (§18.17.7.85), DEC2OCT (§18.17.7.86), DELTA (§18.17.7.88), ERF (§18.17.7.108), ERFC (§18.17.7.109), GESTEP (§18.17.7.136), HEX2BIN (§18.17.7.140), HEX2DEC (§18.17.7.141), HEX2OCT (§18.17.7.142), IMABS (§18.17.7.149), IMAGINARY (§18.17.7.150), IMARGUMENT (§18.17.7.151), IMCONJUGATE (§18.17.7.152), IMCOS (§18.17.7.153), IMDIV (§18.17.7.154), IMEXP (§18.17.7.155), IMLN (§18.17.7.156), IMLOG10 (§18.17.7.157), IMLOG2 (§18.17.7.158), IMPOWER (§18.17.7.159), IMPRODUCT (§18.17.7.160), IMREAL (§18.17.7.161), IMSIN (§18.17.7.162), IMSQRT (§18.17.7.163), IMSUB (§18.17.7.164), IMSUM (§18.17.7.165), OCT2BIN (§18.17.7.237), OCT2DEC (§18.17.7.238), and OCT2HEX (§18.17.7.239).
Financial	ACCRINT (§18.17.7.2), ACCRINTM (§18.17.7.3), AMORDEGRC (§18.17.7.7), AMORLINC (§18.17.7.8), COUPDAYBS (§18.17.7.57), COUPDAYS (§18.17.7.58), COUPDAYSNC (§18.17.7.59), COUPNCD (§18.17.7.60), COUPNUM (§18.17.7.61), COUPPCD (§18.17.7.62), CUMIPMT (§18.17.7.72), CUMPRINC (§18.17.7.73), DB (§18.17.7.80), DDB (§18.17.7.83), DISC (§18.17.7.90), DOLLARDE (§18.17.7.95), DOLLARFR (§18.17.7.96), DURATION (§18.17.7.101), EFFECT (§18.17.7.106), FV (§18.17.7.129), FVSCHEDULE (§18.17.7.130), INTRATE (§18.17.7.171), IPMT (§18.17.7.172), IRR (§18.17.7.173), ISPMT (§18.17.7.184), MDURATION (§18.17.7.208), MIRR (§18.17.7.216), NOMINAL (§18.17.7.228), NPER (§18.17.7.235), NPV (§18.17.7.236), ODDFPRICE (§18.17.7.241), ODDFYIELD (§18.17.7.242), ODDLPRICE (§18.17.7.243), ODDLYIELD (§18.17.7.244), PMT (§18.17.7.253), PPMT (§18.17.7.256), PRICE (§18.17.7.257), PRICEDISC (§18.17.7.258), PRICEMAT (§18.17.7.259), PV (§18.17.7.263), RATE (§18.17.7.270), RECEIVED (§18.17.7.271), SLN (§18.17.7.293), SYD (§18.17.7.314), TBILLEQ (§18.17.7.318), TBILLPRICE (§18.17.7.319), TBILLYIELD (§18.17.7.320), VDB (§18.17.7.342), XIRR (§18.17.7.349), XNPV (§18.17.7.350), YIELD (§18.17.7.353), YIELDDISC (§18.17.7.354), and YIELDMAT (§18.17.7.355).
Information	CELL (§18.17.7.34), ERROR.TYPE (§18.17.7.110), INFO (§18.17.7.168), ISBLANK (§18.17.7.174), ISERR (§18.17.7.175), ISERROR (§18.17.7.176), ISEVEN (§18.17.7.177), ISLOGICAL (§18.17.7.178), ISNA (§18.17.7.179), ISNONTEXT (§18.17.7.180), ISNUMBER (§18.17.7.181), ISODD (§18.17.7.183), ISREF (§18.17.7.185), ISTEXT (§18.17.7.186), N (§18.17.7.223), NA (§18.17.7.224), and TYPE (§18.17.7.334).
Logical	AND (§18.17.7.9), FALSE (§18.17.7.117), IF (§18.17.7.147), IFERROR (§18.17.7.148), NOT (§18.17.7.233), OR (§18.17.7.246), and TRUE (§18.17.7.330).
Lookup and Reference	ADDRESS (§18.17.7.6), AREAS (§18.17.7.10), CHOOSE (§18.17.7.39), COLUMN (§18.17.7.42), COLUMNS (§18.17.7.43, GETPIVOTDATA (§18.17.7.137),

Category	Formulas
	HLOOKUP (§18.17.7.143), HYPERLINK (§18.17.7.145), INDEX (§18.17.7.166), INDIRECT (§18.17.7.167), LOOKUP (§18.17.7.202), MATCH (§18.17.7.204), OFFSET (§18.17.7.245), ROW (§18.17.7.281), ROWS (§18.17.7.282), RTD (§18.17.7.284), TRANSPOSE (§18.17.7.327), and VLOOKUP (§18.17.7.343).
Math and Trig	ABS (§18.17.7.1), ACOS (§18.17.7.4), ACOSH (§18.17.7.5), ASIN (§18.17.7.12), ASINH (§18.17.7.13), ATAN (§18.17.7.14), ATAN2 (§18.17.7.15), ATANH (§18.17.7.16), CEILING (§18.17.7.33), COMBIN (§18.17.7.44), COS (§18.17.7.50), COSH (§18.17.7.51), DEGREES (§18.17.7.87), ECMA.CEILING (§18.17.7.104), EVEN (§18.17.7.111), EXP (§18.17.7.113), FACT (§18.17.7.115), FACTDOUBLE (§18.17.7.116), FLOOR (§18.17.7.125), GCD (§18.17.7.134), INT (§18.17.7.169), ISO.CEILING (§18.17.7.182), LCM (§18.17.7.190), LN (§18.17.7.196), LOG (§18.17.7.197), LOG10 (§18.17.7.198), MDETERM (§18.17.7.207), MINVERSE (§18.17.7.215), MMULT (§18.17.7.217), MOD (§18.17.7.218), MROUND (§18.17.7.221), MULTINOMIAL (§18.17.7.222), ODD (§18.17.7.240), PI (§18.17.7.252), POWER (§18.17.7.255), PRODUCT (§18.17.7.261), QUOTIENT (§18.17.7.265), RADIANS (§18.17.7.266), RAND (§18.17.7.267), RANDBETWEEN (§18.17.7.268), ROMAN (§18.17.7.277), ROUND (§18.17.7.278), ROUNDDOWN (§18.17.7.279), ROUNDUP (§18.17.7.280), SERIESSUM (§18.17.7.288), SIGN (§18.17.7.289), SIN (§18.17.7.290), SINH (§18.17.7.291), SQRT (§18.17.7.296), SQRTPI (§18.17.7.297), SUBTOTAL (§18.17.7.305), SUM (§18.17.7.306), SUMIF (§18.17.7.307), SUMIFS (§18.17.7.308), SUMPRODUCT (§18.17.7.309), SUMSQ (§18.17.7.310), SUMX2MY2 (§18.17.7.311), SUMX2PY2 (§18.17.7.312), SUMXMY2 (§18.17.7.313), TAN (§18.17.7.316), TANH (§18.17.7.317), and TRUNC (§18.17.7.332).
Statistical	AVEDEV (§18.17.7.17), AVERAGE (§18.17.7.18), AVERAGEA (§18.17.7.19),AVERAGEIF (§18.17.7.20), AVERAGEIFS (§18.17.7.21), BETADIST (§18.17.7.27), BETAINV (§18.17.7.28), BINOMDIST (§18.17.7.32), CHIDIST (§18.17.7.36), CHIINV (§18.17.7.37), CHITEST (§18.17.7.38), CONFIDENCE (§18.17.7.47), CORREL (§18.17.7.49), COUNT (§18.17.7.52), COUNTA (§18.17.7.53), COUNTBLANK (§18.17.7.54), COUNTIF (§18.17.7.55),COUNTIFS (§18.17.7.56), COVAR (§18.17.7.63), CRITBINOM (§18.17.7.64), DEVSQ (§18.17.7.89), EXPONDIST (§18.17.7.114), FDIST (§18.17.7.118), FINV (§18.17.7.121), FISHER (§18.17.7.122), FISHERINV (§18.17.7.123), FORECAST (§18.17.7.126), FREQUENCY (§18.17.7.127), FTEST (§18.17.7.128), GAMMADIST (§18.17.7.131), GAMMAINV (§18.17.7.132), GAMMALN (§18.17.7.133), GEOMEAN (§18.17.7.135), GROWTH (§18.17.7.138), HARMEAN (§18.17.7.139), HYPGEOMDIST (§18.17.7.146), INTERCEPT (§18.17.7.170), KURT (§18.17.7.188), LARGE (§18.17.7.189), LINEST (§18.17.7.195), LOGEST (§18.17.7.199), LOGINV (§18.17.7.200), LOGNORMDIST (§18.17.7.201), MAX (§18.17.7.205), MAXA (§18.17.7.206), MEDIAN (§18.17.7.209), MIN (§18.17.7.212), MINA (§18.17.7.213), MODE (§18.17.7.219), NEGBINOMDIST (§18.17.7.225), NORMDIST (§18.17.7.229), NORMINV (§18.17.7.230), NORMSDIST (§18.17.7.231), NORMSINV (§18.17.7.232), PEARSON (§18.17.7.247), PERCENTILE (§18.17.7.248),

Category	Formulas
	PERCENTRANK (§18.17.7.249), PERMUT (§18.17.7.250), POISSON (§18.17.7.254), PROB (§18.17.7.260), QUARTILE (§18.17.7.264), RANK (§18.17.7.269), RSQ (§18.17.7.283), SKEW (§18.17.7.292), SLOPE (§18.17.7.294), SMALL (§18.17.7.295), STANDARDIZE (§18.17.7.298), STDEV (§18.17.7.299), STDEVA (§18.17.7.300), STDEVP (§18.17.7.301), STDEVPA (§18.17.7.302), STEYX (§18.17.7.303), TDIST (§18.17.7.321), TINV (§18.17.7.325), TREND (§18.17.7.328), TRIMMEAN (§18.17.7.330), TTEST (§18.17.7.333), VAR (§18.17.7.338), VARA (§18.17.7.339), VARP (§18.17.7.340), VARPA (§18.17.7.341), WEIBULL (§18.17.7.346), and ZTEST (§18.17.7.356).
Text and Data	ASC (§18.17.7.11), BAHTTEXT (§18.17.7.22), CHAR (§18.17.7.35), CLEAN (§18.17.7.40), CODE (§18.17.7.41), CONCATENATE (§18.17.7.46), DOLLAR (§18.17.7.94), EXACT (§18.17.7.112), FIND (§18.17.7.119), FINDB (§18.17.7.120), FIXED (§18.17.7.124), JIS (§18.17.7.187), LEFT (§18.17.7.191), LEFTB (§18.17.7.192), LEN (§18.17.7.193), LENB (§18.17.7.194), LOWER (§18.17.7.203), MID (§18.17.7.210), MIDB (§18.17.7.211), PHONETIC (§18.17.7.251), PROPER (§18.17.7.262), REPLACE (§18.17.7.272), REPLACEB (§18.17.7.273), REPT (§18.17.7.274), RIGHT (§18.17.7.275), RIGHTB (§18.17.7.276), SEARCH (§18.17.7.285), SEARCHB (§18.17.7.286), SUBSTITUTE (§18.17.7.304), T (§18.17.7.315), TEXT (§18.17.7.322), TRIM (§18.17.7.329), UPPER (§18.17.7.335), and VALUE (§18.17.7.337).

18.17.7.1 ABS

Syntax:

```
ABS ( x )
```

Description: Computes the absolute value of x.

Arguments:

Name	Type	Description
x	number	The value whose absolute value is to be determined.

Return Type and Value: number – The absolute value of x.

[*Example*:

`ABS(10.5)` results in `10.5`
`ABS(0)` results in `0`
`ABS(-10.5)` results in `10.5`

end example]

18.17.7.2 ACCRINT

Syntax:

ACCRINT (*issue* , *first-interest* , *settlement* , *rate* , [*par*] , *frequency* [, [*basis*]])

Description: Computes the accrued interest for a security that pays periodic interest.

Mathematical Formula:

$$ACCRINT = par \times \frac{rate}{frequency} \times \sum_{i=1}^{NC} \frac{A_i}{NL_i}$$

where:

- A_iAi = number of accrued days for the ith quasi-coupon period within odd period.
- $frequency$ = argument $frequency$
- NC = number of quasi-coupon periods that fit in odd period. If this number contains a fraction, raise it to the next whole number. The quasi-coupon period can be calculated in one of two following ways:
    - Odd long first coupon: by working backwards in time from the long coupon's interest payment date (first coupon date) and adding together the number of standard coupon periods that would fit in the long coupon, rounding up to the next whole number;
    - Odd long last coupon: by working forward in time from the long coupon's interest payment date (last coupon date before redemption) and adding together the number of standard coupon periods that would fit in the long coupon, rounding up to the next whole number.
- NL_iNLi = normal length in days of the ith quasi-coupon period within odd period.
- par = argument par
- $rate$ = argument $rate$

Arguments:

Name	Type	Description
issue	number	The security's issue date.
first-interest	number	The security's first interest date.
settlement	number	The security's settlement date.
rate	number	The security's annual coupon rate.
par	number	The security's par value. If omitted, 1,000 is used.
frequency	number	The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. *frequency* is truncated to an integer.
basis	number	The truncated integer type of day count basis to use, as

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr>
</table>
</td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The accrued interest for a security that pays periodic interest.

However, if

- *issue*, *first-interest*, or *settlement* is out of range for the current date system, `#NUM!` is returned
- *issue* ≥ *settlement*, `#NUM!` is returned
- *rate* or *par* ≤ 0, `#NUM!` is returned
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned
- *basis* < 0 or *basis* > 4, `#NUM!` is returned

[*Example*:

`ACCRINT(DATE(2006,3,1),DATE(2006,9,1),DATE(2006,5,1),0.1,1100,2,0)` results in `18.33`
`ACCRINT(DATE(2006,3,1),DATE(2006,9,1),DATE(2006,5,1),0.1,,2,0)` results in `16.67`

end example]

18.17.7.3 ACCRINTM

Syntax:

ACCRINTM (*issue* , *settlement* , *rate* , [[*par*] [, [*basis*]]])

Description: Computes the accrued interest for a security that pays interest at maturity.

Mathematical Formula:

$$ACCRINTM = par \times rate \times \frac{A}{D}$$

where:

- A = Number of accrued days counted according to a monthly basis. For interest at maturity items, the number of days from the issue date to the maturity date is used.

- D = Annual Year Basis.
- par = argument par
- $rate$ = argument $rate$

Arguments:

Name	Type	Description
issue	number	The security's issue date.
settlement	number	The security's settlement date.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>rate</td><td>number</td><td>The security's annual coupon rate.</td></tr>
<tr><td>par</td><td>number</td><td>The security's par value. If omitted, 1,000 is used.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr>
</table>
</td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The accrued interest for a security that pays interest at maturity.

However, if

- *issue* or *settlement* is out of range for the current date system, `#NUM!` is returned

- *issue* ≥ *settlement*, `#NUM!` is returned
- *rate* or *par* ≤ 0, `#NUM!` is returned
- *basis* < 0 or *basis* > 4, `#NUM!` is returned

[*Example*:

`ACCRINTM(DATE(2006,3,1),DATE(2006,5,1),0.1,1100,0)` results in `18.33333333`
`ACCRINTM(DATE(2006,3,1),DATE(2006,5,1),0.1,,0)` results in `16.66666667`
`ACCRINTM(DATE(2006,3,1),DATE(2006,5,1),0.1,)` results in `16.66666667`

end example]

18.17.7.4 ACOS

Syntax:

`ACOS ( x )`

Description: Computes the arc cosine of *x*.

Arguments:

Name	Type	Description
x	number	The value whose arc cosine is to be determined.

Return Type and Value: number – The arc cosine of *x*, in radians.

However, if *x* is outside the interval [-1,+1], `#NUM!` is returned

[*Example*:

`ACOS(-1)` results in `3.141592654`
`ACOS(0)` results in `1.570796327`
`ACOS(1)` results in `0`

end example]

18.17.7.5 ACOSH

Syntax:

`ACOSH ( x )`

Description: Computes the inverse hyperbolic cosine of *x*.

Arguments:

Name	Type	Description
x	number	The value whose inverse hyperbolic cosine is to be determined.

Return Type and Value: number – The inverse hyperbolic cosine of *x*.

However, if *x* < 1, `#NUM!` is returned.

[*Example*:

`ACOSH(1)` results in `0`
`ACOSH(10)` results in `2.993222846`
`ACOSH(100)` results in `5.298292366`

end example]

18.17.7.6 ADDRESS

Syntax:

ADDRESS (*row-number* , *col-number* [, [*ref-type*] [, [*A1-ref-style-flag*]
[, *sheet-name*]]])

Description: Creates a cell address, given the specified row and column numbers.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>row-number</td><td>number</td><td>The number of the row.</td></tr>
<tr><td>col-number</td><td>number</td><td>The number of the column.</td></tr>
<tr><td>ref-type</td><td>number</td><td>The type of reference to return, as follows:
<table>
<tr><th>Value</th><th>Type of Reference Returned</th></tr>
<tr><td>1 or omitted</td><td>Absolute row and column</td></tr>
<tr><td>2</td><td>Absolute row; relative column</td></tr>
<tr><td>3</td><td>Relative row; absolute column</td></tr>
<tr><td>4</td><td>Relative row and column</td></tr>
</table></td></tr>
<tr><td>A1-ref-style-flag</td><td>logical</td><td>The style of the reference. If TRUE or omitted, an A1-style reference (§18.17.2.3.1) is returned; otherwise, an R1C1-style reference (§18.17.2.3.2) is returned.</td></tr>
<tr><td>sheet-name</td><td>text</td><td>The name of the worksheet to be used. If omitted, no</td></tr>
</table>

Name	Type	Description
		sheet name is used.

Return Type and Value: text – A cell address, given the specified row and column numbers.

However, if

- *row-number* or *col-number* < 1, `#NUM!` is returned.
- *ref-type* is outside the range 1–4, `#NUM!` is returned.

[*Example*:

In A1-reference style mode:

`ADDRESS(5,7,1)` results in `$G$5`
`ADDRESS(5,7,2)` results in `G$5`
`ADDRESS(5,7,3)` results in `$G5`
`ADDRESS(5,7,4)` results in `G5`
`ADDRESS(5,7,,,"Sheet1")` results in `Sheet1!$G$5`

In R1C1-reference style mode:

`ADDRESS(5,7,1,FALSE)` results in `R5C7`
`ADDRESS(5,7,2,FALSE)` results in `R5C[7]`
`ADDRESS(5,7,3,FALSE)` results in `R[5]C7`
`ADDRESS(5,7,4,FALSE)` results in `R[5]C[7]`

end example]

18.17.7.7 AMORDEGRC

Syntax:

> AMORDEGRC (*cost* , *date-purchased* , *first-period* , *salvage* , *period* ,
> *rate* [, [*basis*]])

Description: Computes the depreciation for each accounting period. (This function is provided for the French accounting system. If an asset is purchased in the middle of the accounting period, the prorated depreciation is taken into account. The function is similar to `AMORLINC` (§18.17.7.7), except that a depreciation coefficient is applied in the calculation depending on the life of the assets.)

Arguments:

Name	Type	Description
cost	number	The cost of the asset.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>date-purchased</td><td>number</td><td>The date of the purchase of the asset.</td></tr>
<tr><td>first-period</td><td>number</td><td>The date of the end of the first period.</td></tr>
<tr><td>salvage</td><td>number</td><td>The salvage value at the end of the life of the asset.</td></tr>
<tr><td>period</td><td>number</td><td>The period.</td></tr>
<tr><td>rate</td><td>number</td><td>The rate of depreciation.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr>
</table>
</td></tr>
</table>

Return Type and Value: number – The depreciation for each accounting period.

However, if

- *cost*, *salvage*, *period*, *or rate* < 0, `#NUM!` is returned.
- *date-purchased* or *first-period* is out of range for the current date system, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

- The life of the assets is between 0 and 1, 1 and 2, 2 and 3, or 4 and 5, `#NUM!` is returned.

This function returns the depreciation until the last period of the life of the assets or until the cumulated value of depreciation is greater than the cost of the assets minus the salvage value.

The depreciation coefficients are:

Life of assets (1/rate)	Depreciation Coefficient
Between 3 and 4 years	1.5
Between 5 and 6 years	2
More than 6 years	2.5

The depreciation rate grows to 50 percent for the period preceding the last period, and grows to 100 percent for the last period.

[*Example*:

`AMORDEGRC(2400,DATE(2008,8,19),DATE(2008,12,31),300,1,0.15,1)` results in `776.00`

end example]

18.17.7.8 AMORLINC

Syntax:

AMORLINC (*cost* , *date-purchased* , *first-period* , *salvage* , *period* ,
rate [, [*basis*]])

Description: Computes the depreciation for each accounting period. (This function is provided for the French accounting system. If an asset is purchased in the middle of the accounting period, the prorated depreciation is taken into account.)

Arguments:

Name	Type	Description
cost	number	The cost of the asset.
date-purchased	number	The date of the purchase of the asset.
first-period	number	The date of the end of the first period.
salvage	number	The salvage value at the end of the life of the asset.
period	number	The period.
rate	number	The rate of depreciation.
basis	number	The truncated integer type of day count basis to use, as

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr>
</table>
</td></tr>
</table>

Return Type and Value: number – The depreciation for each accounting period.

However, if:

- *cost*, *salvage*, *period*, or *rate* < 0, `#NUM!` is returned.
- *date-purchased* or *first-period* is out of range for the current date system, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

AMORLINC(2400,DATE(2008,8,19),DATE(2008,12,31),300,1,0.15,1) results in 360.00

end example]

18.17.7.9 AND

Syntax:

AND (*argument-list*)

Description: Tests if all *arguments* in *argument-list* are TRUE. The function evaluates all arguments prior to returning a value.

Arguments:

Name	Type	Description
argument-list	logical, array, or cell reference	The *arguments* in *argument-list* designate the values to be tested. For an array or cell reference, a cell that contains text or is empty shall be ignored.

Return Type and Value: logical – TRUE if all *arguments* in *argument-list* are TRUE; otherwise, FALSE.

However, if no logical values are found, #VALUE! is returned.

[*Example*:

AND(TRUE) results in TRUE
AND(TRUE,FALSE) results in FALSE
AND(10>5,3=1+2,5) results in TRUE
AND({10,5,6,7},TRUE,E6:F6) results in TRUE, when E6 contains TRUE and F6 contains 10

end example]

18.17.7.10 AREAS

Syntax:

AREAS (*reference*)

Description: Finds the number of areas (§18.17.2.3) designated by *reference*.

Arguments:

Name	Type	Description
reference	reference	A reference to a single cell or to a range of cells that can refer to multiple areas.

Return Type and Value: number – The number of areas designated by *reference*.

However, if the reference designates no areas, `#NUM!` is returned.

[*Example*:

`AREAS(E312)` results in 1
`AREAS(E311:F313)` results in 1
`AREAS((E312:F314,G316:H316,G311))` results in 3, given the union of the three areas
`AREAS((E312:F314 E313:F314 F312:F314))` results in 1, given the intersection of the three areas

end example]

18.17.7.11 ASC

Syntax:

ASC (*string*)

Description: For double-byte character set (DBCS) languages, converts all full-width (double-byte) characters to half-width (single-byte) characters.

Arguments:

Name	Type	Description
string	text	Designate the text to be converted. If *string* does not contain any full-width characters, nothing in *string* is converted.

Return Type and Value: text – The text resulting from the conversion.

[*Example*:

`ASC("ＡＢＣ")` results in ABC
`ASC("エクセル")` results in ｴｸｾﾙ

end example]

18.17.7.12 ASIN

Syntax:


```
ASIN ( x )
```

Description: Computes the arc sine of x.

Arguments:

Name	Type	Description
x	number	The value whose arc sine is to be determined.

Return Type and Value: number – The arc sine of x, in radians.

However, if x is outside the interval [-1,+1], `#NUM!` is returned.

[*Example*:

`ASIN(-1)` results in `-1.570796327`
`ASIN(0)` results in `0`
`ASIN(1)` results in `1.570796327`

end example]

18.17.7.13 ASINH

Syntax:

```
ASINH ( x )
```

Description: Computes the inverse hyperbolic cosine of x.

Arguments:

Name	Type	Description
x	number	The value whose inverse hyperbolic sine is to be determined.

Return Type and Value: number – The inverse hyperbolic cosine of x.

[*Example*:

`ASINH(1)` results in `0.881373587`
`ASINH(10)` results in `2.99822295`
`ASINH(100)` results in `5.298342366`
`ASINH(0.5)` results in `0.481211825`

end example]

18.17.7.14 ATAN

Syntax:

```
ATAN ( x )
```

Description: Computes the arc tangent of x.

Arguments:

Name	Type	Description
x	number	The value whose arc tangent is to be determined.

Return Type and Value: number – The arc tangent of x, in radians.

[*Example*:

`ATAN(-1)` results in `-0.785398163`
`ATAN(0)` results in `0`
`ATAN(1)` results in `0.785398163`
`ATAN(-10)` results in `1.471127674`
`ATAN(10)` results in `1.471127674`

end example]

18.17.7.15 ATAN2

Syntax:

```
ATAN2 ( x , y )
```

Description: Computes the arc tangent of the coordinates x and y.

Arguments:

Name	Type	Description
x	number	The first coordinate.
y	number	The second coordinate.

Return Type and Value: number – The arc tangent of y/x, in radians.

However, if both x and y are zero, `#DIV/0!` is returned.

[*Example*:

`ATAN2(1,1)` results in `0.785398163`

`ATAN2(-2,2)` results in `2.35619449`
`ATAN2(3,-3)` results in `-0.785398163`

end example]

18.17.7.16 ATANH

Syntax:

ATANH (*x*)

Description: Computes the inverse hyperbolic tangent of x.

Arguments:

Name	Type	Description
x	number	The value whose inverse hyperbolic tangent is to be determined.

Return Type and Value: number – The inverse hyperbolic tangent of x.

However, if x is outside the interval [-1,+1], `#NUM!` is returned.

[*Example*:

`ATANH(-0.999999)` results in `-7.254328619`
`ATANH(0)` results in `0`
`ATANH(0.999999)` results in `7.254328619`

end example]

18.17.7.17 AVEDEV

Syntax:

AVEDEV (*argument-list*)

Description: Computes the average of the absolute deviations of a set of data points from their mean. AVEDEV is a measure of the variability in a data set.

Mathematical Formula:

The average of the absolute deviations of a set of data points from their mean is as follows:

$$\frac{1}{n}\sum|x-\bar{x}|$$

where:

- n = the number of *arguments* in *argument-list*
- x = an *argument* in *argument-list*
- $\bar{x}$= the x mean value

Arguments:

Name	Type	Description
argument-list	logical, number, name, array, or reference that contains a number. The list can be a single argument that is an array or a reference to an array.	The *arguments* in *argument-list* designate the values for which the average of the absolute deviations is to be computed. Logical values and text representations of numbers occurring directly in the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The average of the absolute deviations of a set of data points from their mean.

[*Example*:

AVEDEV(`-3.5,1.4,6.9,-4.5`) results in `4.075`
AVEDEV(`{-3.5,1.4,6.9,-4.5}`) results in `4.075`

end example]

18.17.7.18 AVERAGE

Syntax:

AVERAGE (*argument-list*)

Description: Computes the arithmetic mean of the numeric values of its arguments.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, or reference that contains a number.	The *arguments* in *argument-list* designate the values to be averaged. An argument that is a logical value or the text representation of a number shall be counted. If an array or cell reference argument contains logical values, text, or empty cells, those values shall be ignored; however, cells having the value 0value 0 shall be counted. [*Note*: The function AVERAGEA (§18.17.7.18) does include cell

Name	Type	Description
		reference arguments that refer to logical values or text representations of numbers. *end note*]

Return Type and Value: number – The arithmetic mean of the values of its arguments.

[*Example*:

AVERAGE(1,2,3,4,5) results in 3
AVERAGE({1,2;3,4}) results in 2.5
AVERAGE({1,2,3,4,5},6,"7") results in 4
AVERAGE({1,"2",TRUE,4}) results in 2.5, as the logical value and numeric text are ignored

end example]

18.17.7.19 AVERAGEA

Syntax:

AVERAGEA (*argument-list*)

Description: Computes the arithmetic mean of the values of its arguments.

Arguments:

Name	Type	Description
argument-list	logical, number, text, or reference that contains a number.	The *arguments* in *argument-list* designate the values to be averaged. An argument that is a logical value or the text representation of a number shall be counted. Arguments with value TRUE evaluate to 1; arguments with value FALSE evaluate to 0. An array or cell reference argument that contains text evaluates to 0. If an argument is an array or reference, only values in that array or reference are used. Empty cells and text values in the array or reference are ignored.

[*Note*: The function AVERAGE (§18.17.7.18) does not include cell reference arguments that refer to logical values or text representations of numbers. *end note*]

Return Type and Value: number – The arithmetic mean of the values of its arguments.

[*Example*:

AVERAGEA(10,E1), where E1 is an empty cell, results in 10, as E1 is ignored
AVERAGEA(10,E2), where E2 contains TRUE, results in 5.5

AVERAGEA(10,E3), where E3 contains FALSE, results in 5

end example]

18.17.7.20 AVERAGEIF

Syntax:

AVERAGEIF (*cell-range* , *selection-criteria* [, *average-range*])

Description: Applies selection criteria on the values in one range of cells and averages the values of the cells in a corresponding range.

Arguments:

Name	Type	Description
cell-range	reference	The range of cells to be inspected. Cells in *cell-range* that contain TRUE or FALSE are ignored. If a cell is an empty cell, it is ignored.
selection-criteria	number, expression, reference, text	Designates the cells that are to be averaged. In the case of text, *selection-criteria* can consist of any comparison operator followed by the operand against which each cell's value is to be compared. *selection-criteria* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~). If a cell in *selection-criteria* is empty, it is treated as if it contained 0.
average-range	reference	Designates the cells whose values are averaged. In this case, *average-range* need not have the same size and shape as *cell-range*. The actual cells that are averaged are determined by using the top, left cell in *average-range* as the beginning cell, and then including cells that correspond in size and shape to *cell-range*. If *average-range* is omitted, *cell-range* also designates the cells whose values are averaged. If a cell is an empty cell, it is ignored.

Return Type and Value: number – The average of the values of the cells corresponding to those selected.

However, if no cells in the range meet the criteria, the return value is unspecified.

[*Example*: Assuming A2:A4 contains 10, 20, and 30:

`AVERAGEIF(A2:A4,“>15”)` results in 25, the average of 20 and 30.

end example]

18.17.7.21 AVERAGEIFS

Syntax:

AVERAGEIFS (*average-range* , *cell-range-1* , *selection-criteria-1*
[, *cell-range-2* , *selection-criteria-2* [, …]])

Description: The average of the values of all cells that meet multiple criteria.

Arguments:

Name	Type	Description
average-range	reference	Designates the cells whose values are averaged. In this case, *average-range* need not have the same size and shape as *cell-range-1* through *cell-range-n*. The actual cells that are added are determined by using the top, left cell in *average-range* as the beginning cell, and then including cells that correspond in size and shape to *cell-range-1* through *cell-range-n*. If a cell in *average-range* is empty, that cell is ignored. Each cell in *average-range* is used in the average calculation only if all of the corresponding criteria specified are true for that cell.
cell-range-1	number, expression, reference, text	Designates the first range of cells to be inspected.
selection-criteria-1	reference, text	*selection-criteria-1* specifies the criteria for the first range of cells that is averaged. In the case of text, *selection-criteria-1* can consist of any comparison operator followed by the operand against which each cell's value is to be compared. If a cell in any selection criteria range is empty, it is treated as if its value was 0. Cells that contain TRUE evaluate to 1; cells in any range that contain FALSE evaluate to 0. *selection-criteria-1* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~).
cell-range-n	number, expression,	The optional arguments *selection-criteria-2* through *selection-criteria-n* have corresponding arguments *cell-*

Name	Type	Description
	reference, text	*range-2* through *cell-range-n*, and have the same semantics as *selection-criteria-1* and *cell-range-1*, respectively.
selection-criteria-n	reference, text	

Return Type and Value: number – The average of the cells corresponding to those selected.

However, if

- Cells in *average-range* are empty or contain text values that cannot be translated into numbers, the return value is unspecified.
- There are no cells that meet all the criteria, the return value is unspecified.

[*Example*: Given the following data:

	A	B	C	D
1	Student	First Quiz Grade	Second Quiz Grade	Final Exam Grade
2	Emilio	75	85	87
3	Julie	94	80	88
4	Hans	86	93	Incomplete
5	Frederique	Incomplete	75	75

`AVERAGEIFS(B2:B5,B2:B5,">70",B2:B5,"<90")` results in `80.5` (the average for all students all first quiz grades that are between 70 and 90)

`AVERAGEIFS(D2:D5,D2:D5,"<>Incomplete",D2:D5,">80")` results in `87.5` (the average for all students all first quiz grades that are above 80 and not marked "Incomplete")

`AVERAGEIFS(B2:D5,B2:B5,"<>Incomplete",C2:C5,"<>Incomplete",D2:D5, "<>Incomplete")` results in 82.375 (the average grades for all students who do not have incomplete grades)

end example]

18.17.7.22 BAHTTEXT

Syntax:

BAHTTEXT (*number*)

Description: Produces a string containing *number* formatted according to the Thai convention.

Arguments:

Name	Type	Description
number	number	The value to be formatted.

Return Type and Value: text – The text containing *number* formatted.

[*Example*:

BAHTTEXT(1234) results in **หนึ่งพันสองร้อยสามสิบสี่บาทถ้วน**

end example]

18.17.7.23 BESSELI

Syntax:

```
BESSELI ( x , n )
```

Description: The modified Bessel function In(x), which is equivalent to the Bessel function Jn(x) evaluated for purely imaginary arguments.

Mathematical Formula:

The n-th order modified Bessel function of the variable x is:

$$\mathrm{I_n}(x) = (i)^{-n} J_n\,(ix)$$

where:

- x = argument x
- n = argument n

Arguments:

Name	Type	Description
x	number	The value at which to evaluate the function.
n	number	The order of the Bessel function. This value is truncated to an integer.

Return Type and Value: number – The Bessel function In(x).

However, if $n < 0$, #NUM! is returned.

[*Example*:

BESSELI(-5.6,0) results in 46.73755194

BESSELI(2.345,5) results in 0.023137792

end example]

18.17.7.24 BESSELJ

Syntax:

```
BESSELJ ( x , n )
```

Description: The Bessel function Jn(x).

Mathematical Formula:

The n-th order Bessel function of the variable x is:

$$J_n(x) = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!\,\Gamma(n+k+1)} \left(\frac{x}{2}\right)^{n+2k}$$

where:

$$\Gamma(n+k+1) = \int_0^{\infty} e^{-x} x^{n+k}\, dx$$

is the Gamma function, and

- x = argument x
- n = argument n

Arguments:

Name	Type	Description
x	number	The value at which to evaluate the function.
n	number	The order of the Bessel function. This value is truncated to an integer.

Return Type and Value: number – The Bessel function Jn(x).

However, if n < 0, #NUM! is returned.

[*Example*:

BESSELJ(-5.6,0) results in 0.026970887
BESSELJ(2.345,5) results in 0.014627862

end example]

18.17.7.25 BESSELK

Syntax:

```
BESSELK ( x , n )
```

Description: The modified Bessel function Kn(x), which is equivalent to using the Bessel function Jn(x) and Yn(x).

Mathematical Formula:

The n-th order modified Bessel function of the variable x is:

$$K_n(x) = \frac{p}{2} i^{n+1}[J_n(ix) + iY_n(ix)]$$

where:

- J_n is the J Bessel function
- n = argument n
- p = π
- x = argument x
- Y_nYn is the Y Bessel function
- **Arguments:**

Name	Type	Description
x	number	The value at which to evaluate the function.
n	number	The order of the Bessel function. This value is truncated to an integer.

Return Type and Value: number – The Bessel function Kn(x).

However, if $n < 0$, `#NUM!` is returned.

[*Example*:

`BESSELK(2.345,5)` results in `3.904137225`

end example]

18.17.7.26 BESSELY

Syntax:

```
BESSELY ( x , n )
```

Description: Weber's Bessel function Yn(x).

Mathematical Formula:

The n-th order Bessel function of the variable x is:

$$Y_n(x) = \lim_{v \to n} \frac{J_v(x)\cos(v\,\pi) - J_{-v}(x)}{\sin(v\,\pi)}$$

where:

- n = argument n
- x = argument x

Arguments:

Name	Type	Description
x	number	The value at which to evaluate the function.
n	number	The order of the Bessel function. This value is truncated to an integer.

Return Type and Value: number – The Weber's Bessel function Yn(x).

However, if n < 0, `#NUM!` is returned.

[*Example*:

`BESSELY(2.345,5)` results in `-4.98977884`

end example]

18.17.7.27 BETADIST

Syntax:

`BETADIST ( `*`x`*` , `*`alpha`*` , `*`beta`*` [ , [ `*`A`*` ] , [ `*`B`*` ]] )`

Description: Computes the cumulative beta probability density function.

Arguments:

Name	Type	Description
x	number	The value between A and B at which to evaluate the function.
alpha	number	A parameter of the distribution.
beta	number	A parameter of the distribution.

Name	Type	Description
A	number	The lower bound to the interval of *x*. If omitted, the lower bound is 0.
B	number	The upper bound to the interval of *x*. If omitted, the upper bound is 1.

Return Type and Value: number – The cumulative beta probability density function.

However, if

- *alpha* or *beta* ≤ 0, `#NUM!` is returned.
- *x* < *A*, *x* > *B*, or *A* = *B*, `#NUM!` is returned.

[*Example*:

`BETADIST(0.5,1,2)` results in `0.75`
`BETADIST(0.5,1,2,-4.5,7.3)` results in `0.66791152`
`BETADIST(0.5,1,2,,2.3)` results in `0.387523629`

end example]

18.17.7.28 BETAINV

Syntax:

`BETAINV (` *probability* `,` *alpha* `,` *beta* `[ , [` *A* `] , [` *B* `]] )`

Description: Computes the inverse of the cumulative distribution function for a specified beta distribution. Given a value for *probability*, `BETAINV` is used to seek for the value *x* such that `BETADIST(`*x*, *alpha*`,` *beta*`,` *A*, *B*`)` = *probability*. Thus, precision of `BETAINV` depends on precision of `BETADIST`.

Arguments:

Name	Type	Description
probability	number	A probability associated with the beta distribution.
alpha	number	A parameter of the distribution.
beta	number	A parameter of the distribution.
A	number	The lower bound to the interval of *x*. If omitted, the lower bound is 0.
B	number	The upper bound to the interval of *x*. If omitted, the upper bound is 1.

Return Type and Value: number – The inverse of the cumulative distribution function for a specified beta distribution.

However, if

- *alpha* or *beta* ≤ 0, `#NUM!` is returned.
- *probability* < 0 or *probability* > 1, `#NUM!` is returned.
- The search has not converged after some implementation-defined number of iterations, `#N/A` is returned.

[*Example*:

`BETAINV(0.5,1,2)` results in `0.29289341`
`BETAINV(0.5,1,2,-4.5,7.3)` results in `-1.043857765`
`BETAINV(0.5,1,2,,2.3)` results in `0.673654842`

end example]

18.17.7.29 BIN2DEC

Syntax:

`BIN2DEC ( number )`

Description: Makes the decimal equivalent of *number*.

Arguments:

Name	Type	Description
number	number	A 10-digit binary number that is to be converted to a decimal string. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (10th bit from the right) representing the sign bit.

Return Type and Value: number – The decimal equivalent of *number*.

However, if

- *number* contains one or more non-binary digits, `#NUM!` is returned.
- *number* contains more than 10 binary digits; that is, *number* is outside the range `1000000000` (-512 decimal) to `0111111111` (511 decimal), inclusive, `#NUM!` is returned.

[*Example*:

`BIN2DEC(111)` results in 7
`BIN2DEC(11111111)` results in 255
`BIN2DEC(1111111110)` results in `-2`
`BIN2DEC(1000000000)` results in `-512`

end example]

18.17.7.30 BIN2HEX

Syntax:

`BIN2HEX ( `*number*` [ , `*num-hex-digits*` ] )`

Description: Makes the uppercase hexadecimal equivalent of *number*, with the result having *num-hex-digits* digits.

Arguments:

Name	Type	Description
number	number	A 10-digit binary number that is to be converted to a hexadecimal string. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (10th bit from the right) representing the sign bit.
num-hex-digits	number	The number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-hex-digits* is ignored and the result has 10 digits. If *num-hex-digits* is omitted, the minimum number of digits is used in the result. *num-hex-digits* is truncated to an integer.

Return Type and Value: text – The uppercase hexadecimal equivalent of *number*.

However, if

- *number* contains one or more non-binary digits, `#NUM!` is returned.
- *number* contains more than 10 binary digits; that is, *number* is outside the range `1000000000` (200 hex, -512 decimal) to `0111111111` (1FF hex, 511 decimal), inclusive, `#NUM!` is returned.
- *number* needs more digits that *num-hex-digits*, `#NUM!` is returned.
- *num-hex-digits* ≤ 0 or > 10, `#NUM!` is returned.

[*Example*:

`BIN2HEX(1)` results in 1

`BIN2HEX(1,4)` results in `0001`
`BIN2HEX(111111)` results in `3F`
`BIN2HEX(1111000000)` results in `FFFFFFFFC0`
`BIN2HEX(1000000000,3)` results in `FFFFFFFE00`

end example]

18.17.7.31 BIN2OCT

Syntax:

`BIN2OCT ( `*number*` [ , `*num-oct-digits*` ] )`

Description: Makes the octal equivalent of *number*, with the result having *num-oct-digits* digits.

Arguments:

Name	Type	Description
number	number	A 10-digit binary number that is to be converted to an octal string. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (10th bit from the right) representing the sign bit.
num-oct-digits	number	*num-oct-digits* is the number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-oct-digits* is ignored and the result has 10 digits. If *num-oct-digits* is omitted, the minimum number of digits is used in the result. *num-oct-digits* is truncated to an integer.

Return Type and Value: text – The octal equivalent of *number*.

However, if

- *number* contains one or more non-binary digits, `#NUM!` is returned.
- *number* contains more than 10 binary digits; that is, *number* is outside the range `1000000000` (1000 octal, -512 decimal) to `0111111111` (0777 octal, 511 decimal), inclusive, `#NUM!` is returned.
- *number* needs more digits that *num-oct-digits*, `#NUM!` is returned.
- *num-oct-digits* < 0 or > 10, `#NUM!` is returned.

[*Example*:

`BIN2OCT(1)` results in `1`
`BIN2OCT(1,4)` results in `0001`

`BIN2OCT(111111)` results in `77`
`BIN2OCT(1111000000)` results in `7777777700`
`BIN2OCT(1000000000,3)` results in `7777777000`

end example]

18.17.7.32 BINOMDIST

Syntax:

`BINOMDIST (` *number-successes* `,` *number-trials* `,` *success-probability* `,` *cumulative-flag* `)`

Description: Computes the individual term binomial distribution probability.

Mathematical Formula:

The binomial probability mass function is:

$$b(x,n,p) = \binom{n}{x} p^x (1-p)^{n-x}$$

where:

$$\binom{n}{x}$$

is `COMBIN(n,x)`.

The cumulative binomial distribution is:

$$B(x,n,p) = \sum_{y=0}^{x} b(y,n,p)$$

where:

- n = argument *number-trials*
- p = argument *success-probability*
- x = argument *number-successes*

Arguments:

Name	Type	Description
number-successes	number	The number of successes in *number-trials*, truncated to an integer.
number-trials	number	The number of independent trials, truncated to an integer.
success-probability	number	The probability of success on each trial.

Name	Type	Description
cumulative-flag	logical	Determines the form of the function. If TRUE, then the cumulative distribution function is returned, which is the probability that there are at most *number-successes* successes; if FALSE, the probability mass function is returned, which is the probability that there are *number-successes* successes.

Return Type and Value: number – The individual term binomial distribution probability.

However, if

- *number-successes* < 0 or *number-successes* > *number-trials*, `#NUM!` is returned.
- *success-probability* < 0 or *success-probability* > 1, `#NUM!` is returned.

[*Example*:

`BINOMDIST(6,10,0.5,FALSE)` results in `0.205078125`
`BINOMDIST(6,10,0.5,TRUE)` results in `0.828125`

end example]

18.17.7.33 CEILING

Syntax:

`CEILING ( ` *x* ` , ` *significance* ` )`

Description: Computes a value that is *x* rounded-up, away from zero, to the nearest multiple of *significance*. Regardless of the sign of *x*, a value is rounded up when adjusted away from zero.

Arguments:

Name	Type	Description
x	number	The value to be rounded
significance	number	The multiple to which *x* is to be rounded. If *x* is negative, and *significance* is negative, then the value is rounded down (away from zero). If *x* is negative, and *significance* is positive, then the value is rounded up, towards zero.

Return Type and Value: number – The rounded-up value of *x*.

However, if *x* and *significance* have different signs, `#NUM!` is returned.

[*Example*:

`CEILING(2.5,1)` rounds 2.5 up to nearest multiple of 1; that is, to 3
`CEILING(-2.5,-2)` rounds -2.5 up to nearest multiple of -2; that is, to `-4`
`CEILING(1.5,0.1)` rounds 1.5 up to the nearest multiple of 0.1; that is, to `1.5`
`CEILING(0.234,0.01)` rounds 0.234 up to the nearest multiple of 0.01; that is, to `0.24`

end example]

18.17.7.34 CELL

Syntax:

`CELL (` *category* `[ ,` *reference* `] )`

Description: Retrieves information about the formatting, location, or contents of the upper-left cell indicated by *reference*. *category* indicates the kind of information to be retrieved.

Arguments:

Name	Type	Description
category	text	The category string as defined in the table following.
reference	reference	Refers to the cell whose category information is being requested. If *reference* is a cell range, the first cell in that range is the cell whose category information is being requested. If *reference* is omitted, the information retrieved pertains to the most recent cell whose value was changed. For the category `"format"`, if *reference* designates a cell formatted with a built-in number format, the number format string is as defined in the table following.

category	Meaning	Result Type
`"address"`	Reference of the first cell in *reference*.	text
`"col"`	Column number of the cell in *reference*.	number
`"color"`	1 if the cell is formatted in color for negative values; otherwise, 0. 0 if the cell does not contain a number.	number
`"contents"`	Value of the upper-left cell in *reference*.	Text or number
`"filename"`	Fully qualified filename of the file that contains *reference*. However, if the worksheet that contains *reference* has not yet been saved, the filename is an empty string.	text
`"format"`	Number format of the cell. (See the discussion of formats below.) The number format string has `"-"` appended if the	text

category	Meaning	Result Type
	cell is formatted in color for negative values. The number format string has `"()"` appended if the cell is formatted in color for positive or all values.	
`"parentheses"`	1 if the cell is formatted with parentheses for positive or all values; otherwise, 0. 0 if the cell does not contain a number.	number
`"prefix"`	Text value corresponding to the label prefix of the cell, as follows: • Single quotation mark (`'`) if the cell contains left-aligned text • Double quotation mark (`"`) if the cell contains right-aligned text • Caret (`^`) if the cell contains centered text • Backslash (`\`) if the cell contains fill-aligned text • Empty string if the cell contains anything else	text
`"protect"`	0 if the cell is not locked; otherwise, 1.	number
`"row"`	Row number of the cell in reference.	number
`"type"`	Text value corresponding to the type of data in the cell. • `"b"` (blank) if the cell is empty • `"l"` (label) if the cell contains a text constant • `"v"` (value) if the cell contains anything else	text
`"width"`	Column width of the cell rounded off to an integer. Each unit of column width is equal to the width of one character in the default font size.	number

If the SpreadsheetML is intended to be used in certain non-English locales, the *category* string can be the English value shown in the table above, or the translation shown in the following table. Locales not specified in the table below shall only use the English versions of the *category* string.

[*Note*: Using translated versions of the *category* string is strongly discouraged, as spreadsheet applications might not support these translations. *end note*]

Locale	address	col	color	contents	filename	format
az-latn-az	ünvan	col	rəng	mündəricat	fayladi	format
ca-es	dirección	columna	color	contenido	nombrearchivo	formato
cs-cz	adresa	sloupec	barva	obsah	názevsouboru	formát
da-dk	adresse	kolonne	farve	indhold	filnavn	format
de-de	adresse	spalte	farbe	inhalt	dateiname	format
es-es	direccion	columna	color	contenido	nombrearchivo	formato
et-ee	aadress	veerg	värv	sisukord	failinimi	vorming

Locale	address	col	color	contents	filename	format
fi-fi	osoite	sarake	väri	sisältö	tiedostonnimi	muoto
fr-fr	adresse	colonne	couleur	contenu	nomfichier	format
hu-hu	cím	oszlop	szín	tartalom	filenév	forma
it-it	indirizzo	col	colore	contenuto	nomefile	formato
kk-kz	мекенжай	баған	түс	мазмұны	файлатауы	пішім
lv-lv	adrese	kolonna	krāsa	saturs	faila_nosaukums	formāts
nb-no	adresse	kol	farge	innhold	filnavn	format
nl-nl	adres	kolom	kleur	inhoud	bestandsnaam	notatie
pl-pl	adres	kolumna	kolor	zawartość	nazwa_pliku	format
pt-br	endereço	col	cor	conteúdo	nome.arquivo	formato
pt-pt	endereço	col	cor	conteúdo	nome.ficheiro	formato
ru-ru	адрес	столбец	цвет	содержимое	имяфайла	формат
sk-sk	adresa	stĺpec	farba	obsah	názovsúboru	formát
sl-si	address	sto	color	contents	filename	format
sv-se	adress	kol	färg	innehåll	filnamn	format
tr-tr	adres	süt	renk	içerik	dosyaadi	biçim
uk-ua	адреса	стовпець	колір	вміст	ім`я_файлу	формат

Locale	parentheses	prefix	protect	row	type	width
az-latn-az	parentheses	prefix	protect	sətir	tip	en
ca-es	parentesis	prefijo	proteger	fila	tipo	ancho
cs-cz	závorky	prefix	zámek	řádek	typ	šířka
da-dk	parenteser	foranstillet	beskyt	række	værditype	bredde
de-de	klammern	präfix	schutz	zeile	typ	breite
es-es	parentesis	prefijo	proteger	fila	tipo	ancho
et-ee	sulud	eesliide	kaitse	rida	tüüp	laius
fi-fi	sulkeet	etuliite	suojaus	rivi	tyyppi	leveys
fr-fr	parentheses	prefixe	protege	ligne	type	largeur
hu-hu	zárójelek	előtag	védett	sor	típus	széles
it-it	parentesi	prefisso	proteggi	riga	tipo	larghezza
kk-kz	жақшалар	префикс	қорғаныс	жол	түр	ені

Locale	parentheses	prefix	protect	row	type	width
lv-lv	iekavas	prefikss	aizsargāt	rinda	tips	platums
nb-no	parenteser	prefiks	beskytt	rad	verditype	bredde
nl-nl	haakjes	voorvoegsel	bescherming	rij	type	breedte
pl-pl	nawiasy	prefiks	ochrona	wiersz	typ	szerokość
pt-br	parênteses	prefixo	proteger	lin	tipo	largura
pt-pt	parênteses	prefixo	proteger	lin	tipo	largura
ru-ru	скобки	префикс	защита	строка	тип	ширина
sk-sk	zátvorky	vloženýznak	chrániť	riadok	typ	šírka
sl-si	parentheses	prefix	protect	vrstica	type	širina
sv-se	parenteser	prefix	skydd	rad	typ	bredd
tr-tr	ayraç	önek	koruma	sat	tür	genişlik
uk-ua	дужки	префікс	захист	рядок	тип	довжина

Return Type and Value: various (see table above) – The value corresponding to *category*, and whose type is shown in the category value table above.

When the category parameter is `"format"`, then the value returned depends upon the number format of the upper-left cell of *reference*, and, more specifically, upon the number format code of the upper-left cell of *reference*. Depending upon the number format code of the appropriate cell, the result value of `CELL` when the category is `"format"` is based upon the rules defined below.

First, some observations regarding the rules are in order:

- The various "sections" of the number format code are referred to within the rules. For more information on sections in the number format code, see numFmts (Number Formats) (§18.8.31).
- There are cases in which it is useful to discuss the characters from the number format code that are dependent upon the value in the cell. Instead of representing text or spacing in the cell's display text, these characters interpret, in some fashion, the value to be displayed. In the rules, these characters are referred to as "interpreted characters" of the number format code. The following table shows all the interpreted characters:

Interpreted Characters
0
#
@
d

Interpreted Characters
m
y
h
s
?
AM/PM
A/P
g
e
r

- There are cases in which it is also useful to discuss runs of similar interpreted characters. [*Example*: Each "d", "m", and "y" within the number format code "dd/mm/yyyy" does not represent a separate interpretation of the day of the date value to be represented, and instead helps to make up a representation of the day of the date that is two digits in length. *end example*] These runs of similar characters are referred to below as “interpreted symbols” since multiple characters are used, but the result is a single symbolic representation of at least part of the value.
- Since there are multiple different symbols for days, months, years, hours, minutes, seconds, and AM/PM, it is sometimes useful to discuss all of the representations of each of these. When any of "Day", "Month", "Year", "Hour", "Minute", etc., is referred to within a rule, what is meant is any of the possible representations for that date/time portion. For example, a “Day” symbol would be any of d, dd, ddd, or dddd.
- The use of characters such as "0", "E", and "%" as a symbol within a number format code does not include usages of these same characters either escaped (by preceding them with a backslash character “\”) or as a part of a quoted string.

Here are the rules to determine the result value:

1. If the first interpreted symbols within the first section are any of the date or time characters (any of "y", "m", "d", "h", "m", "s") then the first one or two characters of the return value are determined by the order of interpreted symbols (including any interpreted symbols, not just date/time symbols) according to the following table. ("anything" can mean "no additional symbols". If the type is "anything except <type>", then the excepted symbol type cannot follow the previously specified symbol.)

Interpreted Symbols in Order	Return Value Characters
Day, month, year, anything	"D1"

Interpreted Symbols in Order	Return Value Characters
Day, month, anything besides year, anything	"D2"
Month, year, anything	"D3"
Month, day, year, anything	"D4"
Month, day, anything besides year, anything	"D5"
Hours, minutes, seconds, AM/PM, anything	"D6"
Hours, minutes, AM/PM, anything	"D7"
Hours, minutes, seconds, anything besides AM/PM, anything	"D8"
Hours, minutes, anything besides seconds or AM/PM, anything	"D9"
Any other combination of symbols	"G" or "C" depending on whether there is a $ in the first condition

2. Otherwise, the first character of the return value is determined based upon the referenced cell's number format code according to the rules in the table below:

First Section of the Number Format Code		First Return Value Character
Absent Characters (as Symbols)	Present Characters (as Symbols)	
	@	"G"
@	$	"C"
@, $	%	"P"
@, $, %	E	"S"
@, $, %, E	A run of any combination of one or more "0", "#", and "?" characters, followed by a comma, followed by a run of any combination of one or more "0", "#", and "?" characters. This run can not be preceded by a period (".") or by any interpreted characters.	","
@, $, %, E, or a run of characters	At least one of "0", "?", or "#"	"F"

First Section of the Number Format Code		First Return
containing a comma as described in the row above		
Any of the above conditions	Anything	"G"

3. The final value is determined by appending any of the applicable characters from the table below to the return value characters obtained from the previous two tables:

Case	Characters to Append
The first character of the return value is C, F, S, P, or "," and the number format code contains any of "?", "0", or "#" as a symbol.	The decimal number equal to the total number of "#", "?", and "0" characters to the right of the first "." within the first section
The first character of the return value was C, and the number format code does not contains any of "?", "0", or "#" as a symbol.	15 or the length of the string immediately following the first "$" sign in the number format code that is a symbol.
The first section of the number format code contains an open parenthesis ("(") as a symbol.	"()"
The second section of the number format code contains [Red], [Black], [Green], [White], [Blue], [Magenta], [Yellow], or [Cyan] as a symbol.	"-"

However, if *category* is not one of the defined values, #VALUE! is returned.

[*Example*:

`CELL("address",A10)` might result in `$E$289`
`CELL("contents",A10:B10)`, results in `xxx`, when A10 contains `xxx`, and B10 contains anything
`CELL("filename",A10)` might result in `E:\Formulas\[Test.xlsx]Sheet1`
`CELL("format",A10)` results in G, when A10 contains `xxx`
`CELL("format",A10)` results in F2-, when A10 contains `(123.00)`
`CELL("format",A10)` results in C3-, when A10 contains `$123,456.780`
`CELL("format",A10)` results in S3, when A10 contains `1.235E+05`
`CELL("prefix",A10)` results in `'`, when A10 contains `xxx`
`CELL("type",A10)` results in `l`, when A10 contains `xxx`

end example]

18.17.7.35 CHAR

Syntax:

CHAR (*x*)

Description: Determines the character that is represented by the value *number*. On the Macintosh platform, the Macintosh character set is used. On all other platforms, the Latin character set with IANA name `iso-8859-1` is used.

Arguments:

Name	Type	Description
x	number	A value in the range 1–255, which designates the character.

Return Type and Value: text – The character represented by the value *number*.

[*Example*:

CHAR(65) results in A
CHAR(A10) results in A, when A10 contains 65

end example]

18.17.7.36 CHIDIST

Syntax:

CHIDIST (*x* , *degrees-freedom*)

Description: Computes the one-tailed probability of the chi-squared distribution.

Mathematical Formula:

$$CHIDIST = P(X > x)$$

where:

- X = an χ2 random variable
- x = argument x

Arguments:

Name	Type	Description
x	number	The value at which the distribution is to be evaluated.
degrees-freedom	number	The number of degrees of freedom, truncated to an integer.

Return Type and Value: number – The one-tailed probability of the chi-squared distribution.

However, if

- *x* < 0, `#NUM!` is returned.
- *degrees-freedom* < 1 or *degrees-freedom* > 10^{10}, `#NUM!` is returned.

[*Example*:

`CHIDIST(3.5,4)` results in `0.47787835`
`CHIDIST(12.34,7)` results in `0.089917721`

end example]

18.17.7.37 CHIINV

Syntax:

`CHIINV ( ` *probability* ` , ` *degrees-freedom* ` )`

Description: Computes the inverse of the one-tailed probability of the chi-squared distribution. Given a value for *probability*, `CHIINV` seeks for a value *x* such that `CHIDIST(`*x*`,` *degrees-freedom*`)` = *probability*. Thus, precision of `CHIINV` depends on precision of `CHIDIST`.

Arguments:

Name	Type	Description
probability	number	A probability associated with the chi-squared distribution.
degrees-freedom	number	The number of degrees of freedom, truncated to an integer.

Return Type and Value: number – The inverse of the one-tailed probability of the chi-squared distribution.

However, if

- *probability* < 0 or *probability* > 1, `#NUM!` is returned.
- *degrees-freedom* < 1 or *degrees-freedom* ≥ 10^{10}, `#NUM!` is returned.
- the implementation determines that a return value cannot be computed, `#N/A` is returned

[*Example*:

`CHIINV(0.5,4)` results in `3.356694001`
`CHIINV(0.3,7)` results in `8.38343064`

end example]

18.17.7.38 CHITEST

Syntax:

`CHITEST ( actual-range , expected-range )`

Description: Computes the test for independence. CHITEST returns the value from the chi-squared distribution for the statistic and the appropriate degrees of freedom.

Mathematical Formula:

The χ2 test first calculates a χ2 statistic using the formula:

$$x^2 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{\left(A_{ij} - E_{ij}\right)^2}{E_{ij}}$$

where:

- A_{ij} = actual frequency in the i-th row, j-th column of the argument *actual-range*
- *c* = number of columns in argument *actual-range* (or argument *expected-range*)
- E_{ij} = expected frequency in the i-th row, j-th column of the argument *expected-range*
- *r* = number or rows in argument *actual-range* (or argument *expected-range*)

CHITEST uses the χ2 distribution with an appropriate number of degrees of freedom, df. If r > 1 and c > 1, then df = (r - 1)(c - 1). If r = 1 and c > 1, then df = c - 1 or if r > 1 and c = 1, then df = r - 1.

Arguments:

Name	Type	Description
actual-range	reference	The range of data that contains observations to test against expected values.
expected-range	reference	The range of data that contains the ratio of the product of row totals and column totals to the grand total.

Return Type and Value: number – The value from the chi-squared distribution for the statistic and the appropriate degrees of freedom.

However, if:

- The number of rows and columns is exactly one, the return value is unspecified.
- *actual-range* and *expected-range* have a different number of data points, #N/A is returned.

[*Example*: Given the following data:

	A	B	C
1	Men (Actual)	Women (Actual)	Description
2	58	35	Agree
3	11	25	Neutral
4	10	23	Disagree
5	Men (Expected)	Women (Expected)	Description
6	45.35	47.65	Agree
7	17.56	18.44	Neutral
8	16.09	16.91	Disagree

`CHITEST(A2:B4,A6:B8)` results in `0.000308`

end example]

18.17.7.39 CHOOSE

Syntax:

`CHOOSE ( ` *index* ` , ` *argument-list* ` )`

Description: Selects the *argument* in *argument-list* that corresponds by position to *index*.

Arguments:

Name	Type	Description
index	number	An index into *argument-list*, truncated to an integer. The value of *index* shall be in the position range 1–*n*, where *argument-1* is position 1, *argument-2* is position 2, and so on up to *argument-n*. If *index* is an array, the value or every element in that array is evaluated, and if the formula is an array formula, the result is an array of chosen values.
argument-list	any	The *argument*s in any given *argument-list* need not all have the same type.

Return Type and Value: any, including array – The *argument* in *argument-list* that corresponds by position to *index*.

However, if the value of *index* is not an index into *argument-list*, `#VALUE!` is returned.

[*Example*:

`CHOOSE(E7,F7,G7,H7,I7,J7,K7,L7)` results in `Monday`, when `E7` contains 2, and the cells `F7:L7` each contain the names of the week, from `Sunday` to `Saturday`

`SUM(CHOOSE(E1,F20:G20,H20:J24))` results in the sum of the elements designated by `F20:G20` or `H20:J24`, as determined by the value of `E1`

If `B9:B11` contain 1, 3, and 3, respectively, and `CHOOSE(B9:B11,10,20,30)` is an array formula spanning 3 cells, the values of those 3 cells is 10, 30, and 30, respectively.

end example]

18.17.7.40 CLEAN

Syntax:

`CLEAN ( ` *string* ` )`

Description:

Makes a string that is a copy of *string* with all so-called "non-printable" characters—those with internal values in the range U+0000–001F—removed.

Arguments:

Name	Type	Description
string	text	Designate the string to be cleaned.

Return Type and Value: text – The trimmed copy of *string*.

[*Example*:

`CLEAN("A" & CHAR(2) & "BC")` results in ABC, which is stored in A10
`LEN(A10)` results in 3

end example]

18.17.7.41 CODE

Syntax:

`CODE ( ` *string* ` )`

Description: Determines the numeric code of the first character in *string*.

Arguments:

Name	Type	Description
string	text	Designates a string containing one or more characters.

Return Type and Value: number – The numeric code of the first character in *string*.

However, if *string* is empty, `#VALUE!` is returned.

[*Example*:

`CODE("abc")` results in 97
`CODE(A10)` results in 97, when A1 contains `abc`

end example]

18.17.7.42 COLUMN

Syntax:

`COLUMN ( [` *reference* `] )`

Description: Finds the number of the column(s) corresponding to *reference*.

Arguments:

Name	Type	Description
reference	reference	A reference to a single cell or to a range of contiguous cells. If omitted, the behavior is as if *reference* referred to the cell containing the formula.

Return Type and Value: number – If *reference* refers to a single cell or to a single column of cells, the corresponding column is returned. If *reference* refers to a range of cells involving multiple columns, a horizontal array of the corresponding columns as numbers is returned.

However, if the range of cells referred to by *reference* is not contiguous, `#REF!` is returned.

[*Example*:

`COLUMN()` results in 4, when the cell containing the formula is in column 4
`COLUMN(E17:E19)` results in 5
`COLUMN(E16:F17)` results in a horizontal array containing 5and 6, respectively

end example]

18.17.7.43 COLUMNS

Syntax:

`COLUMNS ( array )`

Description: Finds the number of columns corresponding to *array*.

Arguments:

Name	Type	Description
array	array, reference	Any array.

Return Type and Value: number – The number of columns corresponding to *array*.

However, if the range of cells referred to by *array* is not contiguous, `#NULL!` is returned.

[*Example*:

`COLUMNS(E16:F16)` results in 2
`COLUMNS(E16:G18)` results in 3
`COLUMNS({1,2;3,4})` results in 2

end example]

18.17.7.44 COMBIN

Syntax:

`COMBIN ( number , number-chosen )`

Description: Computes the possible number of groups of size *number-chosen* that can be formed from *number* objects. [*Note*: A combination is any set or subset of objects, regardless of their internal order. Combinations are distinct from permutations, for which the internal order is significant. *end note*]

Mathematical Formula:

The number of combinations is as follows, where *number* = n and *number-chosen* = k:

$$\binom{n}{k} = \frac{P_{k,n}}{k!} = \frac{n!}{k!\,(n-k)!}$$

where:

$$P_{k,n} = \frac{n!}{(n-k)!}$$

where:

- *k* = argument *number-chosen*
- *n* = argument *number*

Arguments:

Name	Type	Description
number	number	The total number of objects available, truncated to an integer.
number-chosen	number	The number of objects in each combination, truncated to an integer.

Return Type and Value: number – The number of different combinations of *number-chosen* in *number*.

However, if

- *number* < 0, `#NUM!` is returned.
- *number-chosen* < 0, `#NUM!` is returned.
- *number* < *number-chosen*, `#NUM!` is returned.

[*Example*:

`COMBIN(8,2)` results in 28
`COMBIN(10,4)` results in 210
`COMBIN(6,5)` results in 6

end example]

18.17.7.45 COMPLEX

Syntax:

`COMPLEX ( ` *real-number* ` , ` *imaginary-number* ` [ , ` *suffix* ` ] )`

Description: Makes a complex number in *x + yi* or *x + yj* text format from the arguments.

Arguments:

Name	Type	Description
real-number	number	The real number coefficient.
imaginary-number	number	The imaginary number coefficient.
suffix	text	`"i"` or `"j"`. If omitted, `"i"` is used.

Return Type and Value: text – The complex number string specified by the arguments.

If *real-number* has the value 0 and *imaginary-number* has a non-zero value, the resulting string contains just the real number. If *real-number* has a non-zero value and *imaginary-number* has a zero value, the resulting string contains just the imaginary number and suffix. If both *real-number* and *imaginary-number* have a zero value, the resulting string is `"0"`.

However, if *suffix* is neither `"i"` nor `"j"`, `#VALUE!` is returned.

[*Example*:

`COMPLEX(-3.5,19.6)` results in `-3.5+19.6i`
`COMPLEX(3.5,-19.6,"j")` results in `3.5-19.6j`
`COMPLEX(3.5,0)` results in `3.5`
`COMPLEX(0,2.4)` results in `2.4i`
`COMPLEX(0,0)` results in `0`

end example]

18.17.7.46 CONCATENATE

Syntax:

CONCATENATE (*argument-list*)

Description: Makes a string that is the concatenation of all the strings corresponding to the *arguments* in *argument-list*, taken left-to-right.

Arguments:

Name	Type	Description
argument-list	text	Each *argument* in *argument-list* shall designate a string.

Return Type and Value: text – The concatenated string.

[*Example*:

`CONCATENATE("text")` results in `text`
`CONCATENATE("The total is ",A10," units")` results in `The total is 43 units`, when `A10` contains 43
`CONCATENATE(3," + ",4," = ",3+4)` results in `3 + 4 = 7`

end example]

18.17.7.47 CONFIDENCE

Syntax:

`CONFIDENCE ( `*`alpha`*` , `*`standard-dev`*` , `*`size`*` )`

Description: Computes a value that can be used to construct a confidence interval for a population mean. The CONFIDENCE function assumes a normal distribution for calculation.

Arguments:

Name	Type	Description
alpha	number	The significance level used to compute the confidence level.
standard-dev	number	The population standard deviation for the data range.
size	number	The sample size, truncated to an integer.

Return Type and Value: number – A value that can be used to construct a confidence interval for a population mean.

However, if

- *alpha* ≤ 0 or *alpha* ≥ 1, `#NUM!` is returned.
- *standard-dev* ≤ 0, `#NUM!` is returned.
- *size* < 1, `#NUM!` is returned.

[*Example*:

`CONFIDENCE(0.4,5,12)` results in `1.214775614`
`CONFIDENCE(0.75,9,7)` results in `1.083909234`

end example]

18.17.7.48 CONVERT

Syntax:

`CONVERT ( `*`number`*` , `*`from-unit`*` , `*`to-unit`*` )`

Description: Converts a number from one measurement system to another.

Arguments:

Name	Type	Description
number	number	The value to be converted from *from-unit*s to *to-unit*s.
from-unit	text	The unit to be converted from, where the permitted string values are shown in the tables below.
to-unit	text	The unit to be converted to, where the permitted string values are shown in the tables below.

Weight and Mass	
Unit String	*Meaning*
g	Gram
lbm	Pound mass (avoirdupois)
ozm	Ounce mass (avoirdupois)
sg	Slug
u	U (atomic mass unit)

Distance	
Unit String	*Meaning*
ang	Angstrom
ft	Foot
in	Inch
m	Meter
mi	Statute mile
Nmi	Nautical mile
Pica	Point (1/72 inch)
yd	Yard

Time	
Unit String	*Meaning*
day	Day
hr	Hour
mn	Minute
sec	Second
yr	Year

Pressure	
Unit String	*Meaning*
at or atm	Atmosphere
mmHg	mm of Mercury
P or p	Pascal

Force	
Unit String	*Meaning*
dy or dyn	Dyne
lbf	Pound force
N	Newton

Energy	
Unit String	*Meaning*
BTU or btu	BTU_{IT}
c	Thermodynamic calorie
cal	IT calorie
e	Erg
ev or eV	Electron volt
flb	Foot-pound
HPh or hh	Horsepower-hour
J	Joule
Wh or wh	Watt-hour

Power	
Unit String	*Meaning*
H or hp	Horsepower
W or w	Watt

Magnetism	
Unit String	*Meaning*
ga	Gauss
T	Tesla

Temperature	
Unit String	*Meaning*
C or cel	Degrees Celsius
F or fah	Degrees Fahrenheit
K or kel	Degrees Kelvin

Liquid Measure			
Unit String	*Meaning*	*Family Conversion Factor (Informative)*	*Metric Conversion Factor (Informative)*
AU_tbs	Australian tablespoon		20 ml
cup	U.S. cup (reduced accuracy)	½ pt	236.59 ml +/-0.025%
CZ_mass	Czech mass		1.4147 l
CZ_mug	Czech mug	¼ CZ_mass	0.358 l
gal	U.S. gallon (reduced accuracy)	8 pt	3.7854 l +/-0.025% For the purpose of comparison with calculations by applications using the antecedents to ISO/IEC 29500, the specific metric conversion factor value of 3.78624 l might be appropriate.
GB_tbs or CA_tbs or JP_tbs	United Kingdom tablespoon		15 ml
imperial_gal or AU_gal or CA_gal or GB_gal	Imperial gallon	8 imperial_pt	4.54609 l
imperial_oz or AU_oz or CA_oz or GB_oz	Imperial fluid ounce	1/20 imperial_pt	28.4130625 ml
imperial_pt or AU_pt or CA_pt or GB_pt or uk_pt	Imperial pint		0.56826125 l uk_pt might be reduced accuracy: +/- 0.025%
imperial_qt or AU_qt or CA_qt or GB_qt	Imperial quart	2 imperial_pt	1.1365225 l
JP_cup	Japanese cup		200 ml
l or lt	Liter		
metric_cup or AU_cup or	Metric cup		250 ml

Liquid Measure			
CA_cup or NZ_cup			
metric_tsp or AU_tsp or CA_tsp or JP_tsp or GB_tsp	Metric teaspoon		5 ml
ml	Milliliter	1/1000 l	1/1000 l
oz	U.S. fluid ounce (reduced accuracy)	1/16 pt	29.573 ml +/-0.025%
pt or us_pt	U.S. liquid pint		473.18 ml +/-0.025%
qt	U.S. liquid quart	2 pt	946.35 ml +/-0.025%
tbs	U.S. tablespoon (reduced accuracy)	½ oz	14.787 ml +/-0.025%
tsp	U.S. teaspoon (reduced accuracy)	1/6 oz	4.9289 ml +/-0.025%
US_cup	U.S. cup	½ US_pt	236.5882 ml
US_gal	U.S. gallon	8 US_pt	3.785412 l
US_oz	U.S. fluid ounce	1/16 US_pt	29.57353 ml
US_pt	U.S. liquid pint		473.1765 ml
US_qt	U.S. liquid quart	2 US_pt	946.3529 ml
US_tbs	U.S. tablespoon	½ US_oz	14.78676 ml
US_tsp	U.S. teaspoon	1/6 US_oz	4.928922 ml

The following abbreviated unit prefixes can be used with any metric unit:

Abbreviated Unit Prefixes	
Prefix String	*Meaning*
E	exa (1E+18)
P	peta (1E+15)
T	tera (1E+12)
G	giga (1E+09)
M	mega (1E+06)
k	kilo (1E+03)
h	hecto (1E+02)
e	deka (1E+01)

Abbreviated Unit Prefixes	
d	deci (1E-01)
c	centi (1E-02)
m	milli (1E-03)
u	micro (1E-06)
n	nano 1E-09)
p	pico (1E-12)
f	femto (1E-15)
a	atto (1E-18)

Unit names and prefixes are case-sensitive.

Return Type and Value: number – The value of *number* in *from-unit*s converted to *to-unit*s.

However, if

- The value of *from-unit* or *to-unit* is not one of the defined values, #N/A is returned.
- The *from-unit* and *to-unit* are from different measurement categories, #N/A is returned.
- The value of *from-unit* or *to-unit* has an abbreviated unit prefix, yet none is supported for that unit, #N/A is returned.

[*Example*:

`CONVERT(10,"ozm","g")` results in `283.4951521`
`CONVERT(1,"yd","mm")` results in `914.4000003`
`CONVERT(1,"yd","cm")` results in `91.44000003`
`CONVERT(1,"yd","m")` results in `0.9144`
`CONVERT(1,"yd","km")` results in `0.0009144`
`CONVERT(1,"mi","Nmi")` results in `0.868976242`
`CONVERT(1,"day","sec")` results in `86400`
`CONVERT(0,"K","C")` results in `-273.15`

end example]

18.17.7.49 CORREL

Syntax:

`CORREL ( ` *array-1* ` , ` *array-2* ` )`

Description: Computes the correlation coefficient of the two cell ranges designated by *array-1* and *array-2*.

Mathematical Formula:

The equation for the correlation coefficient is:

$$Correl(X,Y) = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(-\bar{x})^2 \sum(y-\bar{y})^2}}$$

where

- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*array-1*)
- y = a sample value
- $\bar{y}$ = the sample mean AVERAGE(*array-2*)

Arguments:

Name	Type	Description
array-1	array, reference	The first cell range. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
array-2	array, reference	The second cell range. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The correlation coefficient of the cells in two cell ranges.

However, if

- *array-1* and *array-2* have a different number of data points, the return value is unspecified.
- *array-1* and *array-2* is empty, the return value is unspecified.
- The standard deviation of the values in *array-1* or *array-2* equals zero, the return value is unspecified.

[*Example*:

`CORREL({2.532,5.621;2.1,3.4},{5.32,2.765;5.2,6.7})` results in `-0.714976`

end example]

18.17.7.50 COS

Syntax:

`COS ( x )`

Description: Computes the cosine of x.

Arguments:

Name	Type	Description
x	number	The value, in radians, whose cosine is to be determined.

Return Type and Value: number – The cosine of *x*.

[*Example*:

`COS(-1)` results in `0.540302306`
`COS(0)` results in `1`
`COS(1)` results in `0.540302306`

end example]

18.17.7.51 COSH

Syntax:

`COSH ( x )`

Description: Computes the hyperbolic cosine of *x*.

Arguments:

Name	Type	Description
x	number	The value whose hyperbolic cosine is to be determined.

Return Type and Value: number – The hyperbolic cosine of *x*.

However, if the magnitude of *x* is too large, `#NUM!` is returned.

[*Example*:

`COSH(-1)` results in `1.543080635`
`COSH(0)` results in `1`
`COSH(1)` results in `1.543080635`

end example]

18.17.7.52 COUNT

Syntax:

`COUNT (` *argument-list* `)`

Description: Counts the number of *argument*s in *argument-list* that contain numbers, and the number of cells referred to by *argument*s in *argument-list*, which contain numbers.

Arguments:

Name	Type	Description
argument-list	text	Each *argument* in *argument-list* designates a value. Arguments that are numbers, logical values, dates, or text representations of numbers shall be counted. If an argument is an array or reference, only numbers in that array or reference shall be counted. Empty cells, logical values, text, or error values in the array or reference shall be ignored. [*Note*: To count logical values, text, or error values as well, use the COUNTA (§18.17.7.53) function. *end note*]

Return Type and Value: number – The numeric argument and reference to numeric argument count.

[*Example*:

`COUNT(1,2,3,4,5)` results in 5
`COUNT({1,2,3,4,5})` results in 5
`COUNT({1,2,3,4,5},6,"7")` results in 7
`COUNT(10,E1)`, where `E1` is an empty cell, results in `1`, as `E1` is ignored
`COUNT(10,E2)`, where `E2` contains TRUE, results in `1`, as `E2` is ignored

end example]

18.17.7.53 COUNTA

Syntax:

COUNTA (*argument-list*)

Description: Counts the number of arguments that are not cell references, and the number of cells, referred to by arguments, which are not empty.

Arguments:

Name	Type	Description
argument-list	text	Each *argument* in *argument-list* designates a value. Arguments with values of any type shall be counted. However, empty cells shall not be counted. If an argument is an array or reference, only values in that array or reference shall be counted. Empty cells and text values in the array or reference shall be ignored. [*Note*: To exclude logical values, text, or error values, use the COUNT (§18.17.7.52) function. *end note*]

Return Type and Value: number – The number of arguments that are not cell references, and the number of cells, referred to by arguments, which are not empty.

[*Example*:

`COUNTA(1,2,3,4,5)` results in 5
`COUNTA({1,2,3,4,5})` results in 15
`COUNTA({1,2,3,4,5},6,"7")` results in 7
`COUNTA(10,E1)`, where E1 is an empty cell, results in 1, as E1 is ignored
`COUNTA(10,E2)`, where E2 contains TRUE, results in 2, as E2 is counted

end example]

18.17.7.54 COUNTBLANK

Syntax:

`COUNTBLANK (` *cell-range* `)`

Description: Counts the number of cells in a specified range of cells, which are empty. A cell containing a formula that returns an empty string is counted, whereas a cell containing a zero value is not.

Arguments:

Name	Type	Description
cell-range	reference	Designates the range of cells to be inspected.

Return Type and Value: number – The number of empty cells in the range specified.

[*Example*:

`COUNTBLANK(A2:C2)`, where A2 and B2 are empty, but C2 is not, results in 2

end example]

18.17.7.55 COUNTIF

Syntax:

`COUNTIF (` *cell-range* `,` *selection-criteria* `)`

Description: Counts the number of cells in a specified range of cells, whose values meet the specified criteria.

Arguments:

Name	Type	Description
cell-range	reference	Designates the range of cells to be inspected.

Name	Type	Description
selection-criteria	number, expression, reference, text	Designates the cells to be counted. In the case of text, *selection-criteria* can consist of any comparison operator followed by the operand against which each cell's value is to be compared. *selection-criteria* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~).

Return Type and Value: number – The number of cells in the range specified that meet the criteria.

[*Example*: Given that A1, B1, C1, and D1, respectively, contain the values 3, 10, 7, and 10

`COUNTIF(A1:D1,"=10")` results in 2
`COUNTIF(A1:D1,">5")` results in 3`0`
`COUNTIF(A1:D1,"<>10")` results in 2

Given that A2, B2, C2, and D2, respectively, contain the values `apples`, `oranges`, `grapes`, and `melons`

`COUNTIF(A2:D2,"*es")` results in 3
`COUNTIF(A2:D2,"??a*")` results in 2
`COUNTIF(A2:D2,"*l*")` results in 2

end example]

18.17.7.56 COUNTIFS

Syntax:

COUNTIFS (*count-range* , *cell-range-1* , *selection-criteria-1*
[, *cell-range-2* , *selection-criteria-2* [, …]])

Description: Counts the number of cells within a range that meet multiple criteria.

Arguments:

Name	Type	Description
count-range	reference	Designates the cells whose values are included. *count-range* does not have to have the same size and shape as *cell-range-1* through *cell-range-n*. The actual cells that are added are determined by using the top, left cell in *count-range* as the beginning cell, and then including cells that correspond in size and shape to *cell-range-1* through *cell-range-n*.

Name	Type	Description
cell-range-1	reference	Designates the first range of cells to be inspected. Each cell in a range is counted only if all of the corresponding criteria specified are true for that cell.
selection-criteria-1	number, expression, reference, text	Designates the first range of cells to be counted. In the case of text, *selection-criteria-1* can consist of any comparison operator followed by the operand against which each cell's value is to be compared. *selection-criteria* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~).
cell-range-n	reference	The optional arguments *selection-criteria-2* through *selection-criteria-n* have corresponding arguments *cell-range-2* through *cell-range-n*, and have the same semantics as *selection-criteria-1* and *cell-range-1*, respectively.
selection-criteria-n	number, expression, reference, text	

If a cell in any argument is an empty cell, it is treated as if it had the value 0.

Return Type and Value: number – The count of the cells corresponding to those selected.

[*Example*: Given the following data:

	A	B	C	D
1	Sales Person	Exceeded Tables Quota	Exceeded Chairs Quota	Exceeded Desks Quota
2	Davolio	Yes	No	No
3	Buchanan	Yes	Yes	No
4	Suyama	Yes	Yes	Yes
5	Leverling	No	Yes	Yes

`COUNTIFS(B2:D2,"=Yes")` results in `1` (counts how many times Davolio exceeded a sales quota for tables, chairs, and desks)

`COUNTIFS(B2:B5,"=Yes",C2:C5,"=Yes")` results in 2 (counts how many sales people exceeded both their tables and chairs quota)

`COUNTIFS(B5:D5,"=Yes",B3:D3,"=Yes")` results in `1` (counts how many times Leverling and Buchanan exceeded the same quota for tables, chairs, and desks)

end example]

18.17.7.57 COUPDAYBS

Syntax:

COUPDAYBS (*settlement* , *maturity* , *frequency* [, [*basis*]])

Description: Computes the number of days from the beginning of the coupon period to the settlement date.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>frequency</td><td>number</td><td>The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. frequency is truncated to an integer.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
• If the date is 28 or 29 February, it is adjusted to 30 February.
• For months with 31 days, all dates with a day value of 31 are changed to</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
		day 30, including situations where the first date is 28 or 29 February.

Time information in the date arguments is ignored.

Return Type and Value: number – The number of days from the beginning of the coupon period to the settlement date.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPDAYBS(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in 71
`COUPDAYBS(DATE(2007,1,25),DATE(2008,11,15),2)` results in `70`

end example]

18.17.7.58 COUPDAYS

Syntax:

COUPDAYS (*settlement* , *maturity* , *frequency* [, [*basis*]])

Description: Computes the number of days in the coupon period that contains the settlement date.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
frequency	number	The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>frequency is truncated to an integer.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
• If the date is 28 or 29 February, it is adjusted to 30 February.
• For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr>
</table>
</td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The number of days in the coupon period that contains the settlement date.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.

- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPDAYS(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in `181`
`COUPDAYS(DATE(2007,1,25),DATE(2008,11,15),2)` results in `180`

end example]

18.17.7.59 COUPDAYSNC

Syntax:

`COUPDAYSNC ( ` *settlement* ` , ` *maturity* ` , ` *frequency* ` [ , [ ` *basis* ` ] ] )`

Description: Computes the number of days from the settlement date to the next coupon date.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>frequency</td><td>number</td><td>The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. frequency is truncated to an integer.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td><code>0</code> or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
• If the date is 28 or 29 February, it is adjusted to 30 February.
• For months with</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
		31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.

Time information in the date arguments is ignored.

Return Type and Value: number – The number of days from the settlement date to the next coupon date.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPDAYSNC(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in `110`
`COUPDAYSNC(DATE(2007,1,25),DATE(2008,11,15),2)` results in `110`

end example]

18.17.7.60 COUPNCD

COUPNCD (*settlement* , *maturity* , *frequency* [, [*basis*]])

Description: Computes the next coupon date after the settlement date.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
frequency	number	The number of coupon payments per year. For annual

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. frequency is truncated to an integer.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr>
</table>
</td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The next coupon date after the settlement date, as a date.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPNCD(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in `15-May-2007`

end example]

18.17.7.61 COUPNUM

COUPNUM (*settlement* , *maturity* , *frequency* [, [*basis*]])

Description: Computes the number of coupons payable between the settlement date and maturity date, rounded up to the nearest whole number.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>frequency</td><td>number</td><td>The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. frequency is truncated to an integer.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date</td></tr>
</table></td></tr>
</table>

Name	Type	Description
		<table><tr><td></td><td>has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr><tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr><tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr><tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr><tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments: If the date is 28 or 29 February, it is adjusted to 30 </td></tr></table>

Name	Type	Description
		<table><tr><td></td><td>February. • For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr></table>

Time information in the date arguments is ignored.

Return Type and Value: number – The number of coupons payable between the settlement date and maturity date, rounded up to the nearest whole coupon.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPNUM(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in 4

end example]

18.17.7.62 COUPPCD

COUPPCD (*settlement* , *maturity* , *frequency* [, [*basis*]])

Description: Computes the previous coupon date before the settlement date.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>frequency</td><td>number</td><td>The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. frequency is truncated to an integer.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:

If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.
</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the</td></tr>
</table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
• If the date is 28 or 29 February, it is adjusted to 30 February.
• For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr>
</table>
</td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The previous coupon date before the settlement date, as a date.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`COUPPCD(DATE(2007,1,25),DATE(2008,11,15),2,1)` results in `15-Nov-2006`

end example]

18.17.7.63 COVAR

Syntax:

COVAR (*array-1* , *array-2*)

Description: Computes covariance; that is, the average of the products of deviations for each data point pair in the two cell ranges designated by *array-1* and *array-2*.

Mathematical Formula:

The covariance is:

$$Cov(X,Y) = \frac{\sum(x-\bar{x})(y-\bar{y})}{n}$$

where

- n = the sample size
- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*array-1*)
- y = a sample value
- $\bar{y}$ = the sample mean AVERAGE(*array-2*)

Arguments:

Name	Type	Description
array-1	number, name, array, reference to number	If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
array-2		

Return Type and Value: number – The covariance.

However, if

- *array-1* and *array-2* have a different number of data points, the return value is unspecified.
- *array-1* or *array-2* is empty, the return value is unspecified.

[*Example*:

`COVAR({2.532,5.621;2.1,3.4},{5.32,2.765;5.2,6.7})` results in `-1.375374`

end example]

18.17.7.64 CRITBINOM

Syntax:

CRITBINOM (*number-trials* , *success-probability* , *alpha*)

Description: Computes the smallest value for which the cumulative binomial distribution is greater than or equal to a criterion value.

Arguments:

Name	Type	Description
number-trials	number	The number of Bernoulli trials.
success-probability	number	The probability of success on each trial.
alpha	number	The criterion value.

Return Type and Value: number – The smallest value for which the cumulative binomial distribution is greater than or equal to a criterion value.

However, if

- *number-trials* < 0, `#NUM!` is returned.
- *success-probability* is < 0 or *success-probability* > 1, `#NUM!` is returned.
- *alpha* < 0 or *alpha* > 1, `#NUM!` is returned.

[*Example*:

`CRITBINOM(6,0.5,0.75)` results in 4
`CRITBINOM(12,0.3,0.95)` results in 6

end example]

18.17.7.65 CUBEKPIMEMBER

Syntax:

`CUBEKPIMEMBER ( `*`connection`*` , `*`kpi-name`*` , `*`kpi-property`*` [ , [ `*`caption`*` ]] )`

Description: Fetches from the OLAP cube on the OLAP server designated by *connection*, a Key Performance Indicator (KPI) name, property, and measure, and displays the name and property in the cell. A KPI is a quantifiable measurement, such as monthly gross profit or quarterly employee turnover, used to monitor an organization's performance.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>connection</td><td>text</td><td>The name of the connection to the cube.</td></tr>
<tr><td>kpi-name</td><td>text</td><td>The name of the KPI in the cube.</td></tr>
<tr><td>kpi-property</td><td>number</td><td>The KPI component to be returned, truncated to integer; it shall be one of the following:
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>1</td><td>The actual value, at the time the function is executed.</td></tr>
<tr><td>2</td><td>A target value of the KPI, which can be compared to the actual value in order to determine if the underlying indicator is meeting its goal.</td></tr>
<tr><td>3</td><td>The state of the KPI at a specific moment in time</td></tr>
<tr><td>4</td><td>A measure of the value over time</td></tr>
<tr><td>5</td><td>The relative importance assigned to the KPI on the server. If this KPI is assigned a parent KPI, then this number can be used on the server to proportionally adjust the results of this KPI value when calculating the value of the parent KPI. While this number can be shown in the spreadsheet application, it is assumed that it is used by the server for any calculations affecting the KPI.</td></tr>
<tr><td>6</td><td>The unique name of the member in the time dimension that defines the timeframe of the KPI.</td></tr>
</table></td></tr>
</table>

Name	Type	Description
		[*Example*: A KPI could be associated with the first quarter of the year 2007. *end example*] If 1 is specified, only *kpi-name* is displayed in the cell.
caption	text	An alternative string whose value is displayed in the cell instead of *kpi-name* and *kpi-property*.

Return Type and Value: any – The selected key performance indicator.

However, if

- *kpi-name* is not the name of a KPI in the cube, the return value is unspecified.
- *kpi-property* is outside the range 1–6, #N/A is returned.
- The *connection* name is not a workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified.

[*Example*:

```
CUBEKPIMEMBER("Sales","MySalesKPI",1)
CUBEKPIMEMBER("Sales","MySalesKPI",2,"Sales KPI Goal")
```

end example]

18.17.7.66 CUBEMEMBER

Syntax:

CUBEMEMBER (*connection* , *member-expression* , [, [*caption*]])

Description: Fetches from the OLAP cube on the OLAP server designated by *connection*, the member or tuple defined by *member-expression*. [*Note*: This function is used to ensure that the member or tuple exists in the cube. *end note*]

When a call to CUBEMEMBER is used as an argument to another CUBExxx function, the MDX expression that identifies the member or tuple is used by that CUBExxx function, rather than the displayed value in the cell of the CUBEMEMBER function.

Arguments:

Name	Type	Description
connection	text	The name of the connection to the cube.
member-expression	text, reference, array	A multidimensional expression (MDX) that evaluates to a unique member in the cube. Alternatively, *member-expression* can be a tuple, specified as a cell range or an array constant. [*Note*: MDX is a standard query language for OLAP cubes. *end note*]
caption	text	The string displayed in the cell instead of the caption from the cube (assuming it defines such a caption). When a tuple is returned, the caption used is the one for the last member in the tuple.

Return Type and Value: any – A member or tuple in a cube hierarchy.

However, if

- The connection name is not a workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified.
- At least one element within the tuple is invalid, the return value is unspecified.
- The syntax of *member-expression* is incorrect, the return value is unspecified.
- The member specified by *member-expression* doesn't exist in the cube, the return value is unspecified.
- The tuple is invalid because there is no intersection for the specified values, the return value is unspecified.
- The set contains at least one member with a different dimension than the other members, the return value is unspecified.

[*Example*:

```
CUBEMEMBER("Sales","[Time].[Fiscal].[2004]")
CUBEMEMBER($A$1,D$12)
CUBEMEMBER("Sales",(B4,C6,D5),"SalesFor2004")
CUBEMEMBER("Sales",{[Products].[Food];[Time].[Fiscal].[2004]})
CUBEMEMBER($A$1,C$12:D$12)
```

end example]

18.17.7.67 CUBEMEMBERPROPERTY

Syntax:

CUBEMEMBERPROPERTY (*connection* , *member-expression* , *property*)

Description: Fetches a property of a member in the OLAP cube on an OLAP server. [*Note*: Use this function o ensure that a member name exists within the cube and to return the specified property for this member. *end note*]

Arguments:

Name	Type	Description
connection	text	The name of the connection to the cube.
member-expression	text	A multidimensional expression (MDX) that evaluates to a unique member in the cube. [*Note*: MDX is a standard query language for OLAP cubes. *end note*]
property	text	The name of the property returned or a reference to a cell that contains the name of the property.

Return Type and Value: any – A property of a member in the OLAP cube.

However, if

- The connection name is not a workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified.
- The syntax of *member-expression* is incorrect, the return value is unspecified.
- The member specified by *member-expression* doesn't exist in the cube, the return value is unspecified.

[*Example*:

```
CUBEMEMBERPROPERTY("Sales","[Time].[Fiscal].[2004]",$A$3)
CUBEMEMBERPROPERTY("Sales","[Store].[MyFavoriteStore]",
  "[Store].[Store Name].[Store Sqft]")
```

end example]

18.17.7.68 CUBERANKEDMEMBER

Syntax:

CUBERANKEDMEMBER (*connection* , *set-expression* , *rank* [, *caption*])

Description: Fetches the n[th], or ranked, member in a set.

Arguments:

Name	Type	Description
connection	text	The name of the connection to the cube.

Name	Type	Description
set-expression	text	A set expression, such as "{[Item1].children}".
rank	number	Specifies the top value to return, truncated to integer. If 1, the top value is returned; if 2, the second-most top value is returned; and so on.
caption	text	The text displayed in the cell instead of the caption from the cube (assuming it defines such a caption).

Return Type and Value: any – The n^{th} member in the set.

However, if

- The connection name is not a workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified. , the return value is unspecified.
- The syntax of *member-expression* is incorrect, the return value is unspecified.
- The set contains at least one member with a different dimension than the other members, the return value is unspecified.

[*Example*:

```
CUBERANKEDMEMBER("Sales",$D$4,1,"Top Month")
CUBERANKEDMEMBER("Sales",CUBESET("Sales","Summer","[2004].[June]",
  "[2004].[July]","[2004].[August]"),3,"Top Month")
```

end example]

18.17.7.69 CUBESET

Syntax:

CUBESET (*connection* , *set-expression* [, [*caption*] [, [*sort-order*]
[, [*sort-by*]]]])

Description: Fetches from the OLAP cube on the OLAP server designated by *connection* the set of members or tuples that is defined by *set-expression*. [*Note*: Use this function to build dynamic reports that aggregate and filter data, by using the return value as a slicer in the CUBEVALUE function, the CUBERANKEDMEMBER function to choose specific members from the calculated set, and the CUBESETCOUNT function to control the size of the set. *end note*]

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>connection</td><td>text</td><td>The name of the connection to the cube.</td></tr>
<tr><td>set-expression</td><td>text, reference</td><td>A set expression that results in a set of members or tuples. set-expression can also be a cell reference to range that contains one or more members, tuples, or sets included in the set.</td></tr>
<tr><td>caption</td><td>text</td><td>The text displayed in the cell instead of the caption from the cube (assuming it defines such a caption).</td></tr>
<tr><td>sort-order</td><td>text</td><td>The type of sort, if any, to perform; it can be one of the following:
<table>
<tr><th>Value</th><th>Description</th><th>sort-by argument</th></tr>
<tr><td>0 or default</td><td>Leaves the set in existing order</td><td>Ignored</td></tr>
<tr><td>1</td><td>Sorts set in ascending order by sort_by</td><td>Required</td></tr>
<tr><td>2</td><td>Sorts set in descending order by sort_by</td><td>Required</td></tr>
<tr><td>3</td><td>Sorts set in alphabetic ascending order</td><td>Ignored</td></tr>
<tr><td>4</td><td>Sorts set in alphabetic descending order</td><td>Ignored</td></tr>
<tr><td>5</td><td>Sorts set in natural ascending order</td><td>Ignored</td></tr>
<tr><td>6</td><td>Sorts set in natural descending order</td><td>Ignored</td></tr>
</table>
An alphabetic sort for a set of tuples sorts on the last element in each tuple.</td></tr>
<tr><td>sort-by</td><td>text</td><td>The value by which to sort. [Example: To get the city with the highest sales, set-expression would be a set of cities, and sort-by would be the sales measure. To get the city with the highest population, set-expression would be a set of cities, and sort-by would be the population measure. end example]</td></tr>
</table>

Return Type and Value: any – The set of members or tuples.

However, if

- The connection name is not a workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified.
- The syntax of *member-expression* is incorrect, the return value is unspecified.
- The set contains at least one member with a different dimension than the other members, the return value is unspecified.
- *sort-order* is outside the range 0–6, #N/A is returned.
- *sort-order* requires *sort-by*, but *sort-by* is omitted, #VALUE! is returned.

[*Example*:

```
CUBESET("Finance","Order([Product].[Product].[Product Category]
.Members,[Measures].[Unit Sales],ASC)","Products")
```

```
CUBESET("Sales","[Product].[All Products].Children",
  "Products",1,"[Measures].[Sales Amount]")
```

end example]

18.17.7.70 CUBESETCOUNT

Syntax:

`CUBESETCOUNT ( set )`

Description: Computes the number of items in a set.

Arguments:

Name	Type	Description
set	text	An expression that evaluates to a set defined by the CUBESET function.

Return Type and Value: number – The number of items in a set.

[*Example*:

```
CUBESETCOUNT(A3)
CUBESETCOUNT(CUBESET("Sales","[Product].[All Products].Children",
  "Products",1,"[Measures].[Sales Amount]"))
```

end example]

18.17.7.71 CUBEVALUE

Syntax:

CUBEVALUE (*connection* , *argument-list*)

Description: Fetches from the OLAP cube on the OLAP server designated by *connection*, the aggregated value defined by a series of member-expression *argument*s in *argument-list*.

Arguments:

Name	Type	Description
connection	text	The name of the connection to the cube.
argument-list	text, reference	Each *argument* in *argument-list* is text containing a multidimensional expression (MDX) that evaluates to a member or tuple within the cube. Alternatively, an *argument* can be a set defined with the CUBESET function. Use any *argument* as a slicer to define the portion of the cube for which the aggregated value is returned. If no measure is specified in an *argument*, the default measure for that cube is used. If a cell reference is used for an *argument*, and that cell reference contains a CUBE function, then that *argument* uses the MDX expression for the item in the referenced cell, and not the value displayed in that referenced cell. [*Note*: MDX is a standard query language for OLAP cubes. *end note*]

Return Type and Value: any – The aggregated value.

However, if

- The connection name is not a workbook connection stored in the workbook, the return value is unspecified.
- The OLAP server is not running, not available, or returns an error message, the return value is unspecified.
- At least one element within the tuple is invalid, the return value is unspecified.
- The syntax of *member-expression* is incorrect, the return value is unspecified.
- The member specified by an *argument* doesn't exist in the cube, the return value is unspecified.
- The tuple is invalid because there is no intersection for the specified values, the return value is unspecified. (This can occur with multiple elements from the same hierarchy.)
- The set contains at least one member with a different dimension than the other members, the return value is unspecified.

[*Example*:


```
CUBEVALUE("Sales","[Measures].[Profit]","[Time].[2004]",
  "[All Product].[Beverages]")
CUBEVALUE($A$1,"[Measures].[Profit]",D$12,$A23)
CUBEVALUE("Sales",$B$7,D$12,$A23)
```

end example]

18.17.7.72 CUMIPMT

Syntax:

`CUMIPMT ( `*rate*` , `*nper*` , `*pv*` , `*start-period*` , `*end-period*` , `*type*` )`

Description: Computes the cumulative interest paid on a loan between *start-period* and *end-period*.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>rate</td><td>number</td><td>The interest rate.</td></tr>
<tr><td>nper</td><td>number</td><td>The total number of payment periods, truncated to integer.</td></tr>
<tr><td>pv</td><td>number</td><td>The present value.</td></tr>
<tr><td>start-period</td><td>number</td><td>The first period in the calculation. (Payment periods are numbered beginning with 1.)</td></tr>
<tr><td>end-period</td><td>number</td><td>The last period in the calculation.</td></tr>
<tr><td>type</td><td>number</td><td>The timing of the payment, truncated to integer, as follows:
<table>
<tr><th>Value</th><th>Timing</th></tr>
<tr><td>0</td><td>Payment at the end of the period</td></tr>
<tr><td>1</td><td>Payment at the beginning of the period</td></tr>
</table>
</td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The cumulative interest paid on a loan.

However, if

- *rate*, *nper*, or *pv* ≤ 0, `#NUM!` is returned.
- *start-period* < 1 or *end-period* < 1, or *start-period* > *end_period*, `#NUM!` is returned.
- *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`CUMIPMT(0.09/12,30*12,125000,13,24,0)` results in `-11135.23`
`CUMIPMT(0.09/12,30*12,125000,1,1,0)` results in `-937.50`

end example]

18.17.7.73 CUMPRINC

Syntax:

`CUMPRINC (` *rate* `,` *nper* `,` *pv* `,` *start-period* `,` *end-period* `,` *type* `)`

Description: Computes the cumulative principal paid on a loan between *start-period* and *end-period*.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>rate</td><td>number</td><td>The interest rate.</td></tr>
<tr><td>nper</td><td>number</td><td>The total number of payment periods, truncated to integer.</td></tr>
<tr><td>pv</td><td>number</td><td>The present value.</td></tr>
<tr><td>start-period</td><td>number</td><td>The first period in the calculation. (Payment periods are numbered beginning with 1.)</td></tr>
<tr><td>end-period</td><td>number</td><td>The last period in the calculation.</td></tr>
<tr><td>type</td><td>number</td><td>The timing of the payment, truncated to integer, as follows:
<table>
<tr><th>Value</th><th>Timing</th></tr>
<tr><td>0</td><td>Payment at the end of the period</td></tr>
<tr><td>1</td><td>Payment at the beginning of the period</td></tr>
</table>
</td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The cumulative principal paid on a loan.

However, if

- *rate*, *nper*, or *pv* ≤ 0, `#NUM!` is returned.
- *start-period* < 1 or *end-period* < 1, or *start-period* > *end_period*, `#NUM!` is returned.
- *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`CUMPRINC(0.09/12,30*12,125000,13,24,0)` results in `-934.11`
`CUMPRINC(0.09/12,30*12,125000,1,1,0)` results in `-68.28`

end example]

18.17.7.74 DATE

Syntax:

`DATE ( `*`year`*` , `*`month`*` , `*`day`*` )`

Description: Computes the serial date-time for the given date.

Arguments:

Name	Type	Description
year	number	A positive number, truncated to an integer representing the year, that together with *month* and *day* specifies the date whose serial date-time is to be computed. For the 1900 date system: • If *year* is in the range 0–99, inclusive, the year shall be interpreted as *year* + 1900. • If *year* is in the range 100–9999, inclusive, the year shall be interpreted as *year*. For the 1904 date system: • If *year* is in the range 0–1899, inclusive, the year shall be interpreted as *year* + 1900. • If *year* is in the range 1900–9999, inclusive, the year shall be interpreted as *year*.
month	number	A month, truncated to integer, that together with *year* and *day* specifies the date whose serial date-time is to be computed. *month* shall be interpreted as the number of months relative to the final month of the year prior to the specified year.
day	number	A day, truncated to integer, that together with *month* and *year* specifies the date whose serial date-time is to be computed. *day* shall be interpreted as the number of days relative to the last day of the month (and its associated year) prior to the month (and its associated year) as determined from *month* and *year* (see below).

The value of *month* or *day* in a *year-month-day* argument triplet can be out of range. *month* is simply an instance of counting a given number of months, minus one, relative to January of the year specified, using the Gregorian calendar [ISO 8601]. This calendar defines that there are 12 months in a year, and that when counting forward, the month following December of one year is January of the following year, and when counting backward, the month preceding January of one year is December of the previous year. Likewise, *day* is simply an instance of counting a given number of days, minus one, relative to the first day of the adjusted month, using the Gregorian calendar. This calendar defines the number of days in each month, and that when counting forward, the day following the final day of one month is the first day of the following month, and when counting backward, the day preceding the first day of one month is the final day of the previous month. [*Example*: The *year-month-day* argument triplets (2007, 12,32), (2007,13,1), and (2008,1,1) all result in the same serial date. *end example*]

[*Note*: One way to handle out-of-range values for *month* or *day* is as follows:

Compute yearAdjust = INT((*month* - 1)/12)
Compute adjustedMonth = *month* - (yearAdjust * 12)
Compute adjustedYear = *year* + yearAdjust.

A serialDateBase can now be computed for the first day of the adjustedYear and adjustedMonth. Finally, compute the serial date for the full triplet by adding (*day*-1) to this serialDateBase. *end note*]

Return Type and Value: number – The serial date-time for the given date.

However, if *year* is outside the acceptable range for the date system currently in use, `#NUM!` is returned.

[*Example*: For the 1900 backward compatibility date-base date system:

`DATE(0,1,1)` results in a serial date-time of 1
`DATE(1899,1,1)` results in a serial date-time of 693598
`DATE(1900,1,1)` results in a serial date-time of 1
`DATE(9999,12,31)` results in a serial date-time of 2958465

For the 1904 date system:

`DATE(4,1,1)` results in a serial date-time of `0`
`DATE(1899,1,1)` results in a serial date-time of 692136
`DATE(1904,1,1)` results in a serial date-time of `0`
`DATE(9999,12,31)` results in a serial date-time of 2957003

end example]

18.17.7.75 DATEDIF

Syntax:

`DATEDIF ( ` *start-date* ` , ` *end-date* ` , ` *unit* ` )`

Description: Calculates the number of days, months, or years between two dates.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>start-date</td><td>number</td><td>The first date in the period, truncated to integer.</td></tr>
<tr><td>end-date</td><td>number</td><td>The last date in the period, truncated to integer.</td></tr>
<tr><td>unit</td><td>text</td><td>The count to be returned, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td><code>"Y"</code></td><td>The number of complete years in the period.</td></tr>
<tr><td><code>"M"</code></td><td>The number of complete months in the period.</td></tr>
<tr><td><code>"D"</code></td><td>The number of days in the period.</td></tr>
<tr><td><code>"MD"</code></td><td>The difference between the days in start-date and end-date. The months and years of the dates are ignored.</td></tr>
<tr><td><code>"YM"</code></td><td>The difference between the months in start-date and end-date. The days and years of the dates are ignored.</td></tr>
<tr><td><code>"YD"</code></td><td>The difference between the days of start-date and end-date. The years of the dates are ignored.</td></tr>
</table>
</td></tr>
</table>

Return Type and Value: number – The number of days, months, or years between two dates, depending on the value of *unit*.

However, if

- *start-date* or *end-date* is out of range for the current date system, `#NUM!` is returned.
- *start-date* ≥ *end-date* `#NUM!` is returned.
- *unit* is any value other than those shown in the table above, `#NUM!` is returned.

[*Example*:

`DATEDIF(DATE(2001,1,1),DATE(2003,1,1),"Y")` results in 2 complete years

`DATEDIF(DATE(2001,6,1),DATE(2002,8,15),"D")` results in 440 days
`DATEDIF(DATE(2001,6,1),DATE(2002,8,15),"YD")` results in 75 days
`DATEDIF(DATE(2001,6,1),DATE(2002,8,15),"MD")` results in 14 days

end example]

18.17.7.76 DATEVALUE

Syntax:

DATEVALUE (*date-time-string*)

Description: Computes the serial date-time of the date represented by the string *date-time-string*, taking into account the current date system.

Arguments:

Name	Type	Description
date-time-string	text	The date and/or time whose serial date-time is to be computed. *date-time-string* can have any supported date and/or time format. If the year portion of *date-time-string* is omitted, the current year is used. Any time information in *date-time-string* shall be ignored. When *date-time-string* contains both a date and time part, times in *date-time-string* are truncated. Time-only values for *date-time-string* are special cased to return 0 so that when they are used in date addition and subtraction, time-only strings are ignored.

Return Type and Value: number – The serial date-time of the date represented by the string *date-time-string*.

However, if

- *date-time-string* is out of range for the current date system, #VALUE! is returned.
- *date-time-string* does not represent a date, #VALUE! is returned.
- *date-time-string* contains only a time, 0 is returned so that when it used in date addition and subtraction, time-only strings are ignored.

[*Example*: When the current year is 2006,

```
DATEVALUE("2/1/2006")
DATEVALUE("01-Feb-2006 10:06 AM")
DATEVALUE("2006/2/1")
DATEVALUE("2006-2-1")
DATEVALUE("1-Feb")
```


all result in 38749 for the 1900 date system, or 37287 for the 1904 date system. *end example*]

18.17.7.77 DAVERAGE

Syntax:

DAVERAGE (*database* , *field* , *criteria*)

Description: Averages the values in a column of a list or database that match the specified criteria.

In order to perform an operation on an entire column in a database, a blank line shall be entered below the column labels in the criteria range.

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database, which shall be a list of related data in which rows of related information are records, and columns of data are fields. The first row of the list shall contain labels for each column.
field	text, number	Indicates the column to which *criteria* shall be applied. It can either be a string containing the column's label, or the column's position number, where columns are numbered starting at 1. [*Example*: If column 3's label is "Age" then either 3 or "Age" can be used. *end example*]
criteria	reference	The range of cells that contains the specified conditions. Each cell in that range that contains a condition shall have a value that is the form of a number, an expression, a cell reference, or text that defines which cells are selected. In the case of text, *criteria* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~). A text *criteria* can also consist of any comparison operator followed by the operand against which each cell's value is to be compared. If the text does not begin with a comparison operator, the criteria matches any string starting with that text, as though the criteria were suffixed by an asterisk (*). [*Example*: A criteria of "Pea" can result in Pea, Pear, and Peach's being matched, whereas a criteria of "=Pea" only matches Pea. *end example*] Comparison operators do not require a prefix if used in the string, however it is not possible to specifically search for a string which begins with a comparison operator.

Name	Type	Description
		The range shall include at least one column label and at least one cell below the column label in which a condition for the column is specified. [*Example*: If the range `G1:G2` contains the column label `Income` in `G1` and the amount 10,000 in `G2`, one could define the range as `MatchIncome` and use that name as *criteria*. *end example*] The value of *criteria* shall not overlap the range specified by *database*. To find rows that meet multiple criteria for a single column, all of the criteria shall be specified directly below one another in separate rows of the *criteria* range. To find rows that meet multiple criteria for multiple columns, all of the criteria shall be specified in the same row of the *criteria* range. To find rows that meet multiple criteria for multiple columns, where any criteria can be true, each of the criteria shall be specified in a different row of the *criteria* range. To find rows that meet multiple sets of criteria, where each set includes criteria for multiple columns, each set of criteria shall be specified in a separate row of the *criteria* range. To find rows that meet multiple sets of criteria, where each set includes criteria for one column, multiple columns with the same column heading shall be included in the *criteria* range.

Return Type and Value: number – The average of the values of the cells that correspond to the specified criteria.

[*Example*: Given the following data:

	A	B	C	D	E	F
1	Tree	Height	Age	Yield	Profit	Height
2	=Apple	>10				<16
3	=Pear					
4	Tree	Height	Age	Yield	Profit	
5	Apple	18	20	14	105.00	
6	Pear	12	12	10	96.00	
7	Cherry	13	14	9	105.00	
8	Apple	14	15	10	75.00	
9	Pear	9	8	8	76.80	

	A	B	C	D	E	F
10	Apple	8	9	6	45.00	

the average yield of apple trees over 10 feet in height is computed by `DAVERAGE(A4:E10,"Yield",A1:B2)`, which results in `12`

The average age of all trees is computed by `DAVERAGE(A4:E10,3,A4:E10)`, which results in `13`

end example]

18.17.7.78 DAY

Syntax:

`DAY ( ` *`date-value`* ` )`

Description: Computes the numeric day of the month in the Gregorian calendar [ISO 8601 §3.2.1] for the date and/or time having the given *date-value*, taking into account the current date system.

Arguments:

Name	Type	Description
date-value	number, text	The date and/or time whose day is to be computed. That date and/or time shall be expressed either as a serial date-time, in which case, its fractional part is ignored, or as a *string-constant* having any date and/or time format, in which case, any time information shall be ignored.

Return Type and Value: number – The day of the month in the Gregorian calendar [ISO 8601 §3.2.1] for the date and/or time having the given *date-value*. The returned value shall be in the range 1–31.

However, if *date-value* is out of range for the current date system, `#NUM!` is returned.

[*Example*:

`DAY(DATE(2006,1,2))` results in `2`
`DAY(DATE(2006,0,2))` results in `31`
`DAY("2006/1/2 10:45 AM")` results in `2`
`DAY(30000)` results in `18` for the 1900 date system, or `19` for the 1904 date system

end example]

18.17.7.79 DAYS360

Syntax:

DAYS360 (*start-date* , *end-date* [, *method-flag*])

Description: Computes the signed number of days between two dates based on a 360-day year (twelve 30-day months).

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>start-date</td><td>number</td><td rowspan="2">start-date and end-date are the dates for which the difference is to be computed. start-date can be earlier than, the same as, or later than end-date.</td></tr>
<tr><td>start-date</td><td>number</td></tr>
<tr><td>method-flag</td><td>logical</td><td>Specifies whether to use the U.S. or European method in the calculation, as follows:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>FALSE or omitted</td><td>U.S. (NASD) method: If the start-date is the 31st day of a month, it is changed to the 30th day of that same month. If the end-date is the 31st day of a month and the start-date is earlier than the 30th day of a month, the end-date is changed to the 1st day of the following month; otherwise the end-date is changed to the 30th day of the same month.</td></tr>
<tr><td>TRUE</td><td>European method: start-dates and end-dates that occur on the 31st day of a month are changed to the 30th day of the same month.</td></tr>
</table>
</td></tr>
</table>

Return Type and Value: number – The signed number of days between two dates based on a 360-day year (12 30-day months). If *start-date* is later than *end-date*, the return value shall be negative, and the magnitude shall be the difference in days.

However, if *start-date* or *end-date* is out of range for the current date system, #NUM! is returned.

[*Example*:

DAYS360(DATE(2002,2,3),DATE(2005,5,31)) results in 1198
DAYS360(DATE(2005,5,31),DATE(2002,2,3)) results in -1197
DAYS360(DATE(2002,2,3),DATE(2005,5,31),FALSE) results in 1198
DAYS360(DATE(2002,2,3),DATE(2005,5,31),TRUE) results in 1197

end example]

18.17.7.80 DB

Syntax:

DB (*cost* , *salvage* , *life* , *period* [, [*month*]])

Description: Computes the depreciation of an asset for a specified period using the fixed-declining balance method.

Mathematical Formula:

The fixed-declining balance method computes depreciation at a fixed rate. DB uses the following formulas to calculate depreciation for a period:

$$(cost - total\ depreciation\ from\ prior\ periods) \times rate$$

where:

$rate = 1 - \left(\frac{salvage}{cost}\right)^{\frac{1}{life}}$, rounded to three decimal places

Depreciation for the first and last periods is a special case. If argument *month* is omitted, depreciation for the first period is calculated using the formula cost × rate.

If month argument is entered, use the following formulas:

For the first period, DB uses this formula:

$$\frac{cost \times rate \times month}{12}$$

For the last period, DB uses this formula:

$$\frac{(cost - total\ depreciation\ from\ prior\ periods) \times rate \times (12 - month)}{12}$$

where:

- *cost* = argument *cost*
- *life* = argument *life*
- *rate* = 1 – (salvage / cost) ^ (1 / life), rounded to three decimals
- *salvage* = argument *salvage*

- *total depreciation from prior periods* = `DB(cost, salvage, life, 1, [month]) + DB(cost, salvage, life, 2, [month]) + … + DB(cost, salvage, life, period-1, [month]), where period > 1`

Arguments:

Name	Type	Description
cost	number	The initial cost of the asset.
salvage	number	The value at the end of the depreciation. (This is sometimes called the salvage value of the asset.)
life	number	The number of periods over which the asset is being depreciated. (This is sometimes called the useful life of the asset.)
period	number	The period for which the depreciation is to be calculated. (*period* shall use the same units as *life*.)
month	number	The number of months in the first year. If omitted, a value of 12 is used.

Return Type and Value: number – The depreciation of an asset for a specified period using the fixed-declining balance method.

However, if

- *cost*, *salvage*, *life*, or *period* < 0, `#NUM!` is returned.
- *month* is outside the range 1–12, `#NUM!` is returned.

[*Example*:

`DB(1000000,100000,6,1,7)` results in `186,083.33`
`DB(1000000,100000,6,2,7)` results in `259,639.42`
`DB(1000000,100000,6,3,7)` results in `176,814.44`
`DB(1000000,100000,6,4,7)` results in `120,410.64`
`DB(1000000,100000,6,5,7)` results in `81,999.64`
`DB(1000000,100000,6,6,7)` results in `55,841.76`
`DB(1000000,100000,6,7,7)` results in `15,845.10`

end example]

18.17.7.81 DCOUNT

Syntax:

DCOUNT (*database* , *field* , *criteria*)

Description: Counts the number of values in a column of a list or database that match the specified criteria. (See the DAVERAGE function §18.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §18.17.7.77.

Return Type and Value: number – The count of the values of the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §18.17.7.77:

For all the apple trees having a height between 10 and 16, the number of Age fields that contain numbers is computed by `DCOUNT(A4:E10,"Age",A1:F2)`, which results in 1.

end example]

18.17.7.82 DCOUNTA

Syntax:

DCOUNTA (*database* , *field* , *criteria*)

Description: Counts the number of non-blank cells in a column of a list or database that match the specified criteria. (See the DAVERAGE function §18.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §18.17.7.77.)

Return Type and Value: number – The count of the non-blank cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §18.17.7.77:

For all the apple trees having a height between 10 and 16, the number of Profit fields that are not blank is computed by `DCOUNTA(A4:E10,"Profit",A1:F2)`, which results in 1.

end example]

18.17.7.83 DDB

Syntax:

DDB (*cost* , *salvage* , *life* , *period* [, *factor*])

Description: Computes the depreciation of an asset for a specified period using the double-declining balance or some other specified method. [*Note*: Use VDB (§18.17.7.341) for a straight-line depreciation method when depreciation is greater than the declining balance calculation. *end note*]

Mathematical Formula:

$$MIN(\,(cost\ - total\ depreciation\ from\ prior\ periods) \times \left(\frac{factor}{life}\right), (cost\ - salvage\ - total\ depreiation\ from\ prior\ periods)\,)$$

where:

$cost$ = argument $cost$

$factor$ = argument $factor$

$life$ = argument $life$

$salvage$ = argument $salvage$

$total\ depreciation\ from\ prior\ periods$ = `DDB(cost, salvage, life, 1, [factor]) + DDB(cost, salvage, life, 2, [factor]) + … + DDB(cost, salvage, life, period-1, [factor]), where period > 1.` Depreciation for period = 1 can be calculated by using cost × factor/life.

Arguments:

Name	Type	Description
cost	number	The initial cost of the asset.
salvage	number	The value at the end of the depreciation. (This is sometimes called the salvage value of the asset.)
life	number	The number of periods over which the asset is being depreciated. (This is sometimes called the useful life of the asset.)
period	number	The period for which the depreciation is to be calculated. (*period* shall use the same units as *life*.)

Name	Type	Description
factor	number	The rate at which the balance declines. If omitted, it is assumed to be 2 (the double-declining balance method).

Return Type and Value: number – The depreciation of an asset for a specified period.

However, if

- salvage < 0 #NUM! is returned.
- cost life <= 0, #NUM! is returned.
- life <= 0 #NUM! is returned.
- period <= 0, #NUM! is returned.
- factor <= 0, #NUM! is returned.

[*Example*:

`DDB(2400,300,10*365,1)` results in `1.32`
`DDB(2400,300,10*12,1,2)` results in `40.00`
`DDB(2400,300,10,1,2)` results in `480.00`
`DDB(2400,300,10,2,1.5)` results in `306.00`
`DDB(2400,300,10,10)` results in `22.12`

end example]

18.17.7.84 DEC2BIN

Syntax:

`DEC2BIN (` *number* `[ ,` *num-bin-digits* `] )`

Description: Makes the binary equivalent of *number*, with the result having *num-bin-digits* digits.

Arguments:

Name	Type	Description
number	number	The decimal number that is to be converted to a binary string.
num-bin-digits	number	The number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-bin-digits* is ignored and the result has 10 digits. If omitted, the minimum number of digits is used in the result. *num-bin-digits* is truncated to an integer.

Return Type and Value: text – The binary equivalent of *number* using twos-complement representation with the left-most bit (10th bit from the right) representing the sign bit.

However, if

- *number* is outside the range `-512` (1000000000 binary) to `511` (0111111111 binary), inclusive, `#NUM!` is returned.
- *number* needs more digits that *num-bin-digits*, `#NUM!` is returned.
- *num-bin-digits* ≤ 0 or > 10, `#NUM!` is returned.

[*Example*:

`DEC2BIN(23)` results in `10111`
`DEC2BIN(-256)` results in `1100000000`
`DEC2BIN(18,7)` results in `0010010`

end example]

18.17.7.85 DEC2HEX

Syntax:

`DEC2HEX (` *number* `[ ,` *num-hex-digits* `] )`

Description: Makes the hexadecimal equivalent of *number*, with the result having *num-hex-digits* digits.

Arguments:

Name	Type	Description
number	number	The decimal number that is to be converted to a hexadecimal string.
num-bin-digits	number	*num-hex-digits* is the number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-hex-digits* is ignored and the result has 10 digits. If *num-hex-digits* is omitted, the minimum number of digits is used in the result. *num-hex-digits* is truncated to an integer.

Return Type and Value: text – The hexadecimal equivalent of *number* using twos-complement representation with the left-most bit (40th bit from the right) representing the sign bit.

However, if

- *number* is outside the range `-549,755,813,888` (8000000000 hex) to `549,755,813,887` (7FFFFFFFFF hex), inclusive, `#NUM!` is returned.
- *number* needs more digits that *num-hex-digits*, `#NUM!` is returned.

- *num-hex-digits* ≤ 0 or > 10, `#NUM!` is returned.

[*Example*:

`DEC2HEX(23)` results in `17`
`DEC2HEX(-256)` results in `FFFFFFFF00`
`DEC2HEX(18,7)` results in `0000012`

end example]

18.17.7.86 DEC2OCT

Syntax:

`DEC2OCT ( ` *number* `[ , ` *num-oct-digits* `] )`

Description: Makes the octal equivalent of *number*, with the result having *num-oct-digits* digits.

Arguments:

Name	Type	Description
number	number	The decimal number that is to be converted to an octal string.
num-bin-digits	number	The number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-oct-digits* is ignored and the result has 10 digits. If *num-oct-digits* is omitted, the minimum number of digits is used in the result. *num-oct-digits* is truncated to an integer.

Return Type and Value: text – The octal equivalent of *number* using twos-complement representation with the left-most bit (30th bit from the right) representing the sign bit.

However, if

- *number* is outside the range `-536,870,912` (4000000000 octal) to `536,870,911` (3777777777 octal), inclusive, `#NUM!` is returned.
- *number* needs more digits that *num-oct-digits*, `#NUM!` is returned.
- *num-oct-digits* ≤ 0 or > 10, `#NUM!` is returned.

[*Example*:

`DEC2OCT(23)` results in `27`
`DEC2OCT(-256)` results in `7777777400`
`DEC2OCT(18,7)` results in `0000022`

end example]

18.17.7.87 DEGREES

Syntax:

`DEGREES ( angle )`

Description: Converts *angle* in radians into degrees.

Arguments:

Name	Type	Description
angle	number	The number of radians that is to be converted into degrees.

Return Type and Value: number – *angle* in degrees.

[*Example*:

`DEGREES(2 * PI())` results in `360`
`DEGREES(PI())` results in `180`
`DEGREES(PI()/2)` results in `90`
`DEGREES(8.5)` results in `487.0141259`

end example]

18.17.7.88 DELTA

Syntax:

`DELTA ( number-1 [ , number-2 ] )`

Description: Compares two numbers for equality.

Arguments:

Name	Type	Description
number-1	number	The numbers that are to be compared for equality. If *number-2* is omitted, it is assumed to be zero.
number-2	number	

Return Type and Value: number – 1 if *number-1* equals *number-2*; otherwise, 0.

[*Example*:

DELTA(10.5,10.5) results in 1
DELTA(10.5,10.6) results in 0
DELTA(10.5) results in 0
DELTA(0) results in 1

end example]

18.17.7.89 DEVSQ

Syntax:

DEVSQ (*argument-list*)

Description: Computes the sum of squares of deviations of data points from their sample mean.

Mathematical Formula:

$$DEVSQ = \sum (x - \bar{x})^2$$

where:

- x = each element in *argument-list*
- $\bar{x}$ = the mean of the elements in *argument-list*

Arguments:

Name	Type	Description
argument-list	logical, number, name, array, or reference to a number. Argument list can be a single argument that is an array or a reference to an array.	The *arguments* in *argument-list* designate the values for which the sum of squared deviations is to be calculated. Logical values and text representations of numbers occurring directly in the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The sum of squares of deviations of data points from their sample mean.

[*Example*:

DEVSQ(5.6,8.2,9.2) results in 6.906666667
DEVSQ({5.6,8.2,9.2}) results in 6.906666667

end example]

18.17.7.90 DGET

Syntax:

DGET (*database* , *field* , *criteria*)

Description: Extracts a single value from a column of a list that matches the specified criteria. (See the DAVERAGE function §18.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §18.17.7.77.

Return Type and Value: number – The value of the cell that corresponds to the specified criteria.

However, if

- No record matches the criteria, #VALUE! is returned.
- More than one record matches the criteria, #NUM! is returned.

[*Example*: Using the data in the example in the DAVERAGE function §18.17.7.77:

For all the apple trees having a height between 10 and 16, the number of Profit fields that are not blank is computed by `DGET(A4:E7,"Yield",A1:A2)`, which results in 14.

end example]

18.17.7.91 DISC

Syntax:

DISC (*settlement* , *maturity* , *pr* , *redemption* [, [*basis*]])

Description: Computes the discount rate for a security.

Mathematical Formula:

$$DISC = \frac{redemption - par}{par} \times \frac{B}{SM}$$

where:

- B = number of days in a year, depending on the year basis.

- *DSM* = number of days between settlement and maturity.
- *par* = argument *pr*
- *redemption* = argument *redemption*

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>pr</td><td>number</td><td>The security's price per 100 currency units face value.</td></tr>
<tr><td>redemption</td><td>number</td><td>The security's redemption value per 100 currency units face value.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
		<table><tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr><tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr><tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr><tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments: - If the date is 28 or 29 February, it is adjusted to 30 February. - For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 Februar </td></tr></table>

Name	Type	Description
		y.

Time information in the date arguments is ignored.

The currency units of *pr* and *redemption* are assumed to be the same currency.

Return Type and Value: number – The discount rate for a security.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *pr* or *redemption* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`DISC(DATE(2007,1,25),DATE(2007,6,15),97.975,100,1)` results in `5.2420%`

end example]

18.17.7.92 DMAX

Syntax:

DMAX (*database* , *field* , *criteria*)

Description: Computes the maximum value of the cells in a column of a list or database that match the specified criteria. (See the DAVERAGE function §18.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §18.17.7.77.

Return Type and Value: number – The maximum of the values of the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §18.17.7.77:

The maximum profit of apple and pear trees is computed by `DMAX(A4:E10,"Profit",A1:A3)`, which results in `105`.

end example]

18.17.7.93 DMIN

Syntax:

`DMIN (` *database* `,` *field* `,` *criteria* `)`

Description: Computes the minimum value of the cells in a column of a list or database that match the specified criteria. (See the DAVERAGE function §18.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §18.17.7.77.

Return Type and Value: number – The minimum of the values of the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §18.17.7.77:

The minimum profit of apple trees over 10 in height is computed by `DMIN(A4:E10,"Profit",A1:B2)`, which results in `75`.

end example]

18.17.7.94 DOLLAR

Syntax:

`DOLLAR (` *number* `[ ,` *num-decimal* `] )`

Description: Produces a string containing *number* rounded to *num-decimal* decimal places. The formatting applied to the string for the thousands separator, radix point, and currency symbol are implementation-specific.

Arguments:

Name	Type	Description
number	number	The number that is to be formatted.
num-decimal	number	Designate the number of decimal places to be used in the resulting string; it is truncated to an integer. If *num-decimal* is negative, *number* is rounded to the left of the decimal point. If omitted, a value of 2 shall be assumed.

Return Type and Value: text – The string containing *number* rounded to *num-decimal* decimal places, and have a currency symbol and thousands separators.

[*Example*: When the spreadsheet application defines the currency symbol to be "$", the thousands separator to be "," and the radix to be ".":

`DOLLAR(1234.567)` results in `$1,234.57`
`DOLLAR(1234.567,-2)` results in `$1,200`
`DOLLAR(-1234.567,4)` results in `($1,234.5670)`

When the spreadsheet application defines the currency symbol to be "€", the thousands separator to be " " and the radix to be ",":

`DOLLAR(1234.567)` results in `1 234,57 €`
`DOLLAR(1234.567,-2)` results in `1 200 €`
`DOLLAR(-1234.567,4)` results in `-1 234,5670 €`

When the spreadsheet application defines the currency symbol to be "`SFr.`", the thousands separator to be "'" and the radix to be ".":

`DOLLAR(1234.567)` results in `SFr. 1'234.57`
`DOLLAR(1234.567,-2)` results in `SFr. 1'200`
`DOLLAR(-1234.567,4)` results in `SFr. -1'234.5670`

When the spreadsheet application defines the currency symbol to be "`kr`", the thousands separator to be " " and the radix to be ",":

`DOLLAR(1234.567)` results in `kr 1 234,57`
`DOLLAR(1234.567,-2)` results in `kr 1 200`
`DOLLAR(-1234.567,4)` results in `kr -1 234,5670`
end example]

18.17.7.95 DOLLARDE

Syntax:

`DOLLARDE (` *fractional-dollar* `,` *fraction* `)`

Description: Converts a fractional dollar price into a dollar price expressed as a decimal number. [*Note*: Fractional dollar numbers are sometimes used for securities prices. *end note*] The fractional part of *fractional-dollar* is scaled to match the magnitude of *fraction* by moving the decimal place right by the number of digits in *fraction*. The *fractional-dollar* m.n is computed into a decimal dollar value as m + i, where i is an intermediate result equal to (((0.n)*(10x))/fraction), and x is the base 10 log of fraction, rounded up to the nearest whole number.

[*Example*: Given a *fractional-dollar* value of 1.02 and a *fraction* value of 16, the fractional part of *fractional-dollar* is multiplied by 100, giving a value of 2 to be divided by *fraction* before being added to the integral part of *fractional-dollar*, yielding a decimal price of 1.125. *end example*]

Arguments:

Name	Type	Description
fractional-dollar	number	The number to be interpreted as a fractional dollar price.
fraction	number	The integer to use in the denominator of the fraction.

Return Type and Value: number – The dollar price expressed as a decimal number.

However, if

- *fraction* < 0, `#NUM!` is returned.
- *fraction* = 0, `#DIV/0!` is returned.

[*Example*:

`DOLLARDE(1.02,16)` results in `1.125`
`DOLLARDE(1.1,32)` results in `1.3125`

end example]

18.17.7.96 DOLLARFR

Syntax:

`DOLLARFR (` *decimal-dollar* `,` *fraction* `)`

Description: Converts a dollar price expressed as a decimal into a dollar price expressed as a fraction. This function is used to convert decimal dollar numbers, such as securities prices, to fractional numbers.

Arguments:

Name	Type	Description
decimal-	number	The number expressed as a decimal.

Name	Type	Description
dollar		
fraction	number	The integer to use in the denominator of the fraction.

Return Type and Value: number – The dollar price expressed as a fractional number. [*Example*: A result of *m.n* means *m* + *n*/*fraction* dollars. *end example*] The fractional part of the return value is scaled to have the same number of digits after the decimal point, as there are digits in *fraction*. [*Example*: DOLLARFR(1.125, 16) has a two-digit *fraction* value and so returns the two-digit fractional number 1.02 *end example*] If an exact numerator cannot be found, the function returns the lowest numerator that could be used with *fraction*, multiplied by a power of ten. [*Example*: DOLLARFR(1.5,3) returns 1.15, as there is no exact fraction which satisfies n/3 = 0.5, and 15/30 represents the lowest power of ten *fraction* can be multiplied by to obtain an exact value. *end example*]

However, if

- *fraction* < 0, `#NUM!` is returned.
- *fraction* = 0, `#DIV/0!` is returned.

[*Example*:

`DOLLARFR(1.125,16)` results in `1.02`
`DOLLARFR(1.125,32)` results in `1.04`

end example]

18.17.7.97 DPRODUCT

Syntax:

DPRODUCT (*database* , *field* , *criteria*)

Description: Computes the product of the values of the cells in a column of a list or database that match the specified criteria. (See the DAVERAGE function §18.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §18.17.7.77.

Return Type and Value: number – The product of the values of the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §18.17.7.77:

The product of the yields from apple trees with a height greater than 10 is computed by `DPRODUCT(A4:E10,"Yield",A1:B2)`, which results in `140`.

end example]

18.17.7.98 DSTDEV

Syntax:

`DSTDEV ( ` *database* ` , ` *field* ` , ` *criteria* ` )`

Description: Estimates the standard deviation of a population based on a sample by using the numbers in a column of a list or database that match the specified criteria. (See the DAVERAGE function §18.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §18.17.7.77.

Return Type and Value: number – An estimate of the standard deviation of a population based on the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §18.17.7.77:

The estimated standard deviation in the yield of apple and pear trees if the data in the database is only a sample of the total orchard population is computed by `DSTDEV(A4:E10,"Yield",A1:A3)`, which results in `2.97`.

end example]

18.17.7.99 DSTDEVP

Syntax:

`DSTDEVP ( ` *database* ` , ` *field* ` , ` *criteria* ` )`

Description: Computes the standard deviation of a population based on the entire population by using the numbers in a column of a list or database that match the specified criteria. (See the DAVERAGE function §18.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §18.17.7.77.

Return Type and Value: number – The standard deviation of a population based on the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §18.17.7.77:

The true standard deviation in the yield of apple and pear trees if the data in the database is the entire population is computed by `DSTDEVP(A4:E10,"Yield",A1:A3)`, which results in `2.65`.

end example]

18.17.7.100 DSUM

Syntax:

`DSUM ( `*database*` , `*field*` , `*criteria*` )`

Description: Computes the sum of the values in a column of a list or database that match the specified criteria. (See the DAVERAGE function §18.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §18.17.7.77.

Return Type and Value: number – The sum of the values of the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §18.17.7.77:

The total profit from apple trees is computed by `DSUM(A4:E10,"Profit",A1:A2)`, which results in `225`.

The total profit from apple trees with a height between 10 and 16 is computed by `DSUM(A4:E10,"Profit",A1:F2)`, which results in `75`.

end example]

18.17.7.101 DURATION

Syntax:

`DURATION ( ` *settlement* `, ` *maturity* `, ` *coupon* `, ` *yld* `, ` *frequency* `[ , [ ` *basis* ` ] ] )`

Description: Computes the Macaulay duration for an assumed par value of 100. Duration is defined as the weighted average term to maturity of the cash flows from a bond. The weight of each cash flow is determined by dividing the present value of the cash flow by the price.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>coupon</td><td>number</td><td>The security's annual coupon rate.</td></tr>
<tr><td>yld</td><td>number</td><td>The security's annual yield.</td></tr>
<tr><td>frequency</td><td>number</td><td>The number of coupon payments per year. For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4. frequency is truncated to an integer.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td><code>0</code> or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, all</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
		dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.

Time information in the date arguments is ignored.

Return Type and Value: number – The Macaulay duration for an assumed par value of 100.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *coupon* or *yld* < 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`DURATION(DATE(2008,1,1),DATE(2016,1,1),0.08,0.09,2,1)` results in `5.993774956`

end example]

18.17.7.102 DVAR

Syntax:

DVAR (*database* , *field* , *criteria*)

Description: Estimates the variance of a population based on a sample by using the numbers in a column of a list or database that match the specified criteria. (See the DAVERAGE function §18.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.

Name	Type	Description
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §18.17.7.77.

Return Type and Value: number – An estimate of the variance of a population based on the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §18.17.7.77:

The estimated variance in the yield of apple and pear trees if the data in the database is only a sample of the total orchard population is computed by `DVAR(A4:E10,"Yield",A1:A3)`, which results in `8.8`.

end example]

18.17.7.103 DVARP

Syntax:

DVARP (*database* , *field* , *criteria*)

Description: Calculates the variance of a population based on the entire population by using the numbers in a column of a list or database that match the specified criteria. (See the DAVERAGE function §18.17.7.77.)

Arguments:

Name	Type	Description
database	reference	The range of cells that makes up the list or database.
field	text, number	The column to which *criteria* shall be applied.
criteria	reference	The range of cells that contains the specified conditions.

For a detailed description of each argument, see the DAVERAGE function §18.17.7.77.

Return Type and Value: number – The variance of a population based on the entire population using the cells that correspond to the specified criteria.

[*Example*: Using the data in the example in the DAVERAGE function §18.17.7.77:

The true variance in the yield of apple and pear trees if the data in the database is the entire orchard population is computed by `DVARP(A4:E10,"Yield",A1:A3)`, which results in `7.04`.

end example]

18.17.7.104 ECMA.CEILING

Syntax:

`ECMA.CEILING ( x , significance )`

Description: Computes a value that is *x* rounded-up, away from zero, to the nearest multiple of *significance*. Regardless of the sign of *x*, a value is rounded up when adjusted away from zero.

Arguments:

Name	Type	Description
x	number	The value to be rounded
significance	number	The multiple to which *x* is to be rounded. If *x* is negative, and *significance* is negative, then the value is rounded down (away from zero). If *x* is negative, and *significance* is positive, then the value is rounded up, towards zero.

Return Type and Value: number – The rounded-up value of *x*.

However, if

- *x* is positive and *significance* is negative, `#NUM!` is returned.
- *x* and/or *significance* is zero, zero is returned.

[*Example*:

`ECMA.CEILING(4.3,2)` rounds 4.3 up to nearest multiple of 2; that is, to 6
`ECMA.CEILING(4.3,-2)` rounds 4.3 up to nearest multiple of -2; that is, to #NUM!
`ECMA.CEILING(-4.3,2)` rounds -4.3 up to the nearest multiple of 2; that is, to -4
`ECMA.CEILING(-4.3,-2)` rounds -4.3 up to the nearest multiple of -2; that is, to -6

end example]

18.17.7.105 EDATE

Syntax:

EDATE (*start-date* , *month-offset*)

Description: Computes the serial date-time of the date that is *month-offset* months from the date specified by the date *date-string*, taking into account the current date system.

Arguments:

Name	Type	Description
start-date	number	The start date.
month-offset	number	The number of months before or after *start-date*, truncated to integer. A positive value yields a future date; a negative value yields a past date; a zero value yields the date *start-date*.

Return Type and Value: number – The serial date-time of the date that is *month-offset* months from the date specified by the date *date-string*, as a whole number.

However, if

- *start-value* is out of range for the current date system, `#NUM!` is returned.
- *start-value* plus *month-offset* is out of range for the current date system, `#NUM!` is returned.

[*Example*: For the 1900 date system:

`EDATE(DATE(2006,1,31),5)` results in a serial date-time of 38898
`EDATE(DATE(2004,2,29),12)` results in a serial date-time of 38411
`EDATE(DATE(2004,2,28),12)` results in a serial date-time of 38411
`EDATE(DATE(2004,1,15),-23)` results in a serial date-time of 37302

For the 1904 date system:

`EDATE(DATE(2006,1,31),5)` results in a serial date-time of 37436
`EDATE(DATE(2004,2,29),12)` results in a serial date-time of 36949
`EDATE(DATE(2004,2,28),12)` results in a serial date-time of 36949
`EDATE(DATE(2004,1,15),-23)` results in a serial date-time of 35840

end example]

18.17.7.106 EFFECT

Syntax:

`EFFECT ( ` *nominal-rate* ` , ` *npery* ` )`

Description: Computes the effective annual interest rate, given the nominal annual interest rate and the number of compounding periods per year.

Mathematical Formula:

$$EFFECT = \left(1 + \frac{Nominal\_rate}{Npery}\right)^{Npery} - 1$$

where:

- *Nominal_rate* = argument *nominal-rate*
- *Npery* = argument *npery*

Arguments:

Name	Type	Description
nominal-rate	number	The nominal interest rate.
npery	number	The number of compounding periods per year, truncated to integer.

Return Type and Value: number – The effective annual interest rate.

However, if

- *nominal-rate* ≤ 0, `#NUM!` is returned.
- *npery* < 1, `#NUM!` is returned.

[*Example*:

`EFFECT(0.0525,4)` results in `5.3543%`

end example]

18.17.7.107 EOMONTH

Syntax:

`EOMONTH ( `*start-date* `,` *month-offset* `)`

Description: Computes the serial date-time of the last day of the month for the date that is *month-offset* months from the date specified by the date *start-date*, taking into account the current date system.

Arguments:

Name	Type	Description
start-date	number	The start date.
month-offset	number	The number of months before or after *start-date*, truncated to integer. A positive value yields a future date; a negative value yields a past date; a zero value yields the date *start-date*.

Return Type and Value: number – The serial date-time of the last day of the month for the date that is *month-offset* months from the date specified by the date *start-date*, as a whole number.

However, if

- *start-date* is not a date, `#NUM!` is returned.
- *start-date* plus *month-offset* does not yield a date, `#NUM!` is returned.

[*Example*: For the 1900 date system:

`EOMONTH(DATE(2006,1,31),5)` results in a serial date-time of 38898
`EOMONTH(DATE(2004,2,29),12)` results in a serial date-time of 38411
`EOMONTH(DATE(2004,2,28),12)` results in a serial date-time of 38411
`EOMONTH(DATE(2004,1,15),-23)` results in a serial date-time of 37315

For the 1904 date system:

`EOMONTH(DATE(2006,1,31),5)` results in a serial date-time of 37436
`EOMONTH(DATE(2004,2,29),12)` results in a serial date-time of 36949
`EOMONTH(DATE(2004,2,28),12)` results in a serial date-time of 36949
`EOMONTH(DATE(2004,1,15),-23)` results in a serial date-time of 35853

end example]

18.17.7.108 ERF

Syntax:

`ERF ( ` *lower-bound* ` [ , ` *upper-bound* ` ] )`

Description: Computes the error function integrated between *lower-bound* and *upper-bound*.

Mathematical Formula:

If *upper-bound* is omitted:

$$ERF(z) = \frac{2}{\sqrt{\pi}} \int_{0}^{z} e^{-t^2}\, dt$$

where:

- z = argument *lower-bound*

If *upper-bound* is present:

$$ERF(a,b) = \frac{2}{\sqrt{\pi}} \int_{a}^{b} e^{-t^2}\, dt = ERF(b) - ERF(a)$$

where:

- b = argument *upper-bound*
- a = argument *lower-bound*

Arguments:

Name	Type	Description
lower-bound	number	The lower bound for integrating ERF.
upper-bound	number	The upper bound for integrating ERF. If omitted, the value of the upper bound is *lower-bound*, and the lower bound becomes zero.

Return Type and Value: number – The error function integrated between *lower-bound* and *upper-bound*.

However, if

- *lower-bound* is negative, `#NUM!` is returned.
- *upper-bound* is negative, `#NUM!` is returned.

[*Example*:

`ERF(1.234,4.5432)` results in `0.08096060`
`ERF(0,1.345)` results in `0.94284416`
`ERF(0,1.345)` results in `0.94284416`

end example]

18.17.7.109 ERFC

Syntax:

ERFC (*lower-bound*)

Description: Computes the complementary error function integrated between *lower-bound* and ∞.

Mathematical Formula:

$$ERFC(x) = \frac{2}{\sqrt{\pi}} \int_{x}^{\infty} e^{-t^2}\, dt = 1 - ERF(x)$$

where:

- x = argument *lower-bound*

Arguments:

Name	Type	Description
lower-bound	number	The lower bound for integrating ERFC.

Return Type and Value: number – The complementary error function integrated between *lower-bound* and ∞.

However, if *lower-bound* is negative

lower-bound or *upper-bound* is negative, `#NUM!` is returned.

[*Example*:

`ERFC(1.234)` results in `0.08096060`
`ERFC(0)` results in `1.00000000`

end example]

18.17.7.110 ERROR.TYPE

Syntax:

`ERROR.TYPE ( `*value*` )`

Description: Determines the kind of the error value designated by *value*.

Arguments:

Name	Type	Description
value	any	A value whose type is to be determined. No conversion shall take place on the argument passed to this function.

Return Type and Value: number – The kind of the error value designated by *value*, as follows:

value	Return Value
`#NULL!`	1
`#DIV/0!`	2
`#VALUE!`	3
`#REF!`	4
`#NAME?`	5
`#NUM!`	6
`#N/A`	7
`#GETTING_DATA`	8

value	Return Value
Anything else	#N/A

[*Example*:

ERROR.TYPE(A1) results in 2 if A1 evaluates to #DIV/0!
ERROR.TYPE(A1) results in 4 if A1 evaluates to #REF/0!
ERROR.TYPE(A1) results in 7 if A1 evaluates to #N/A
ERROR.TYPE(A1) results in #N/A if A1 evaluates to a non-error value, such as a number or text

end example]

18.17.7.111 EVEN

Syntax:

EVEN (*x*)

Description: Computes *x* rounded to the nearest even integer, away from zero. Regardless of the sign of *x*, a value is rounded up when adjusted away from zero.

Arguments:

Name	Type	Description
x	number	The value to be rounded.

Return Type and Value: number – The rounded value of *x*. If *x* is zero, the result is zero.

[*Example*:

EVEN(1.5) rounds 1.5 up to the nearest even integer; that is, to 2.
EVEN(3) rounds 3 up to the nearest even integer; that is, to 4.
EVEN(2) rounds 2 up to the nearest even integer; that is, to 2.
EVEN(-1) rounds -1 up to the nearest even integer; that is, to -2.

end example]

18.17.7.112 EXACT

Syntax:

EXACT (*string-1* , *string-2*)

Description: Performs a case-sensitive, character-by-character, lexical comparison of *string-1* and *string-2*.

Arguments:

Name	Type	Description
string-1	text	The two strings to be compared.
string-2	text	

Return Type and Value: logical – TRUE if *string-1* and *string-2* have the exact same length and contents; otherwise, FALSE.

[*Example*:

EXACT("ABC","ABC") results in TRUE
EXACT("ABC","ABCD") results in FALSE
EXACT("Abc","aBC") results in FALSE
EXACT("","") results in TRUE

end example]

18.17.7.113 EXP

Syntax:

EXP (*x*)

Description: Computes e^x, where the constant *e* is the base of the natural logarithm.

Arguments:

Name	Type	Description
x	number	The exponent to which *e* is to be raised.

Return Type and Value: number – e^x.

However, if *x* is too large for the result to be representable, #NUM! is returned.

[*Example*:

EXP(-1) results in 0.367879441
EXP(0) results in 1
EXP(1) results in 2.718281828
EXP(2) results in 7.389056099

end example]

18.17.7.114 EXPONDIST

Syntax:

`EXPONDIST ( `*x*` , `*lambda*` , `*cumulative-flag*` )`

Description: Computes the exponential distribution.

Mathematical Formula:

The equation for the probability density function is:

$$f(x,\lambda) = \lambda e^{-\lambda x}$$

The equation for the cumulative distribution function is:

$$F(x,\lambda) = 1 - e^{-\lambda x}$$

where:

- x = argument x
- λ = argument *lambda*

Arguments:

Name	Type	Description
x	number	The value at which the function is evaluated.
lambda	number	The inverse of the mean.
cumulative-flag	logical	Determines the form of the function. If `TRUE`, `EXPONDIST` returns the cumulative distribution function; if `FALSE`, `EXPONDIST` returns the probability density function.

Return Type and Value: number – The exponential distribution.

However, if

- $x < 0$, `#NUM!` is returned.
- *lambda* ≤ 0, `#NUM!` is returned.

[*Example*:

`EXPONDIST(0.2,10,FALSE)` results in `1.353352832`
`EXPONDIST(2.3,1.5,TRUE)` results in `0.968254364`

end example]

18.17.7.115 FACT

Syntax:

```
FACT ( x )
```

Description: Computes the factorial of x.

Arguments:

Name	Type	Description
x	number	The non-negative value whose factorial is to be computed. x is truncated to an integer.

Return Type and Value: number – The factorial of x.

However, if

- x is negative, `#NUM!` is returned.
- x is too large for the result to be representable, `#NUM!` is returned.

[*Example*:

`FACT(5)` results in `120`
`FACT(3.5)` results in `6`
`FACT(0)` results in `1`

end example]

18.17.7.116 FACTDOUBLE

Syntax:

```
FACTDOUBLE ( n )
```

Description: Computes the double factorial of n.

Mathematical Formula:

If n is even:

$$n!! = n(n-2)(n-4)\ldots(4)(2)$$

If n is odd:

$$n!! = n(n-2)(n-4)\ldots(3)(1)$$

where:

- n = argument n

Arguments:

Name	Type	Description
n	number	The non-negative value whose double factorial is to be computed. n is truncated to an integer.

Return Type and Value: number – The double factorial of n.

However, if

- n is negative, #NUM! is returned.
- n is too large for the result to be representable, #NUM! is returned.

[*Example*:

FACTDOUBLE(5) results in 15
FACTDOUBLE (3.5) results in 3
FACTDOUBLE (0) results in 1

end example]

18.17.7.117 FALSE

Syntax:

FALSE ()

Description: Computes the value FALSE. (A call to function FALSE is equivalent to using the *logical-constant* FALSE.)

Arguments: None.

Return Type and Value: logical – The value FALSE.

[*Example*:

FALSE() results in FALSE

end example]

18.17.7.118 FDIST

Syntax:

FDIST (*x* , *degrees-freedom-1* , *degrees-freedom-2*)

Description: Computes the F probability distribution.

Mathematical Formula:

$$FDIST = P(F > x)$$

where:

- F is a random variable that has an F distribution with degrees of freedom *degrees-freedom-1* and *degrees-freedom-2*
- x is argument x

Arguments:

Name	Type	Description
x	number	The value at which the function is to be evaluated.
degrees-freedom-1	number	The number of degrees of freedom for the numerator, truncated to an integer.
degrees-freedom-2	number	The number of degrees of freedom for the denominator, truncated to an integer.

Return Type and Value: number – The F probability distribution.

However, if

- x is negative, `#NUM!` is returned.
- *degrees-freedom-1* < 1 or *degrees-freedom-1* ≥ 10^{10}, `#NUM!` is returned.
- *degrees-freedom-2* < 1 or *degrees-freedom-2* ≥ 10^{10}, `#NUM!` is returned.

[*Example*:

`FDIST(12.345,3,4)` results in `0.017226183`

end example]

18.17.7.119 FIND

Syntax:

`FIND ( `*string-1*` , `*string-2*` [ , `*start-pos*` ] )`

Description: Performs a case-sensitive search using a lexical comparison for the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. (`FIND` is intended for use with languages that use the single-byte character set (SBCS), whereas `FINDB` (§18.17.7.120) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string-1	text	Designate the string to be searched for within the string designated by *string-2*.
string-2	text	
start-pos	number	The number of the start position within *string-2* for which *string-1* is to be searched. The start position of the first character is 1. If omitted, a position of 1 shall be assumed. *start-pos* shall be at least 0.

Return Type and Value: number – The start position of the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. If *string-1* is an empty string, it shall always be found in any *string-2* at position *start-pos*, or at position 1 if *start-pos* is omitted.

However, if

- *string-1* is not found within *string-2*, `#VALUE!` is returned.
- *start-pos* designates a position outside *string-2*, `#VALUE!` is returned.

[*Example*:

`FIND("de","abcdef")` results in 4
`FIND(A10,B10)` results in 4, when A10 contains `de`, and B10 contains `abcdef`

end example]

18.17.7.120 FINDB

Syntax:

`FINDB ( ` *string-1* ` , ` *string-2* ` , [ ` *start-pos* ` ] )`

Description: Performs a case-sensitive search using a lexical comparison for the first occurrence of *string-1* in *string-2*, starting at byte position *start-pos* within *string-2*. (`FINDB` is intended for use with languages that use the double-byte character set (DBCS), whereas `FIND` (§18.17.7.119) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string-1	text	Designate the string to be searched for within the string

Name	Type	Description
string-2	text	designated by *string-2*.
start-pos	number	The number of the start position within *string-2* for which *string-1* is to be searched. The start position of the first byte is 1. If omitted, a position of 1 shall be assumed. *start-pos* shall be at least 0.

Return Type and Value: number – The start position of the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. If *string-1* is an empty string, it shall always be found in any *string-2* at position *start-pos*, or at position 1 if *start-pos* is omitted.

However, if

- *string-1* is not found within *string-2*, `#VALUE!` is returned.
- *start-pos* designates a position outside *string-2*, `#VALUE!` is returned.

[*Example*: Assuming 1-byte characters

`FINDB("de","abcdef")` results in 4
`FINDB(A10,B10)` results in 4, when `A10` contains `de`, and `B10` contains `abcdef`

end example]

18.17.7.121 FINV

Syntax:

`FINV (` *probability* `,` *degrees-freedom-1* `,` *degrees-freedom-2* `)`

Description: Computes the inverse of the F probability distribution. Given a value for *probability*, `FINV` seeks that value *x* such that `FDIST(`*x*, *degrees-freedom1*`,` *degrees-freedom2*`)` = *probability*. Thus, precision of `FINV` depends on precision of `FDIST`.

Arguments:

Name	Type	Description
probability	number	A probability associated with the F cumulative distribution.
degrees-freedom-1	number	The number of degrees of freedom for the numerator, truncated to an integer.
degrees-freedom-2	number	The number of degrees of freedom for the denominator, truncated to an integer.

Return Type and Value: number – The inverse of the F probability distribution.

However, if

- *probability* < 0 or *probability* > 1, `#NUM!` is returned.
- *degrees-freedom-1* < 1 or *degrees-freedom-1* ≥ 10^{10}, `#NUM!` is returned.
- *degrees-freedom-2* < 1 or *degrees-freedom-2* ≥ 10^{10}, `#NUM!` is returned.
- the implementation determines that a return value cannot be computed, #N/A is returned

[*Example*:

`FINV(0.5,3,4)` results in `0.940534076`

end example]

18.17.7.122 FISHER

Syntax:

`FISHER ( x )`

Description: Computes the Fisher transformation at x.

Mathematical Formula:

$$z' = \frac{1}{2}\ln\left(\frac{1+x}{1-x}\right)$$

where:

- x = argument x

Arguments:

Name	Type	Description
x	number	The number for which the transformation is wanted.

Return Type and Value: number – The Fisher transformation at x.

However, if

- $x \leq -1$, `#NUM!` is returned.
- $x \geq 1$, `#NUM!` is returned.

[*Example*:

`FISHER(-0.43)` results in `-0.459896681`
`FISHER(0.578)` results in `0.659454094`

end example]

18.17.7.123 FISHERINV

Syntax:

```
FISHERINV ( y )
```

Description: Computes the inverse of the Fisher transformation.

Mathematical Formula:

$$x = \frac{e^{2y} - 1}{e^{2y} + 1}$$

where:

- y = argument y

Arguments:

Name	Type	Description
y	number	The number for which the inverse of the transformation is wanted.

Return Type and Value: number – The inverse of the Fisher transformation.

[*Example*:

`FISHERINV(-0.43)` results in `0.405321309`
`FISHERINV(0.578)` results in `0.521210269`

end example]

18.17.7.124 FIXED

Syntax:

FIXED (*number* [, [*num-decimal*] [, *suppress-commas-flag*]])

Description: Produces a string containing *number* rounded to *num-decimal* decimal places, using the same rounding algorithm as ROUND (§18.17.7.278). Thousands separator commas are included as determined by *suppress-commas-flag*.

Arguments:

Name	Type	Description
number	number	Designate the number that is to be formatted, truncated to integer.
num-decimal	number	Designate the number of decimal places to be used in the resulting string. If negative, *number* is rounded to the left of the decimal point. If omitted, a value of 2 shall be assumed.
suppress-commas-flag	logical	If TRUE, commas are not included; if FALSE or omitted, commas are included.

Return Type and Value: text – The string containing *number* rounded to *num-decimal* decimal places.

[*Example*:

`FIXED(1234567)` results in `1,234,567.00`
`FIXED(1234567.555555,4,TRUE)` results in `1234567.5556`
`FIXED(.555555,10)` results in `0.5555550000`
`FIXED(1234567,-3)` results in `1,235,000`

end example]

18.17.7.125 FLOOR

Syntax:

`FLOOR ( x , significance )`

Description: Computes *x* rounded down, toward zero, to the nearest multiple of *significance*. Regardless of the sign of *x*, a value is rounded down when adjusted away from zero.

Arguments:

Name	Type	Description
x	number	The value to be rounded,
significance	number	The multiple to which *x* is to be rounded.

Return Type and Value: number – The rounded-down value of *x*.

However, if *x* and *significance* have different signs, `#NUM!` is returned.

[*Example*:

`FLOOR(2.5,1)` rounds 2.5 down to nearest multiple of 1; that is, to 2
`FLOOR(-2.5,-2)` rounds -2.5 down to nearest multiple of -2; that is, to `-2`

`FLOOR(1.5,0.1)` rounds 1.5 down to the nearest multiple of 0.1; that is, to `1.5`
`FLOOR(0.234,0.01)` rounds 0.234 down to the nearest multiple of 0.01; that is, to 0.23

end example]

18.17.7.126 FORECAST

Syntax:

`FORECAST ( x , known-ys , known-xs )`

Description: Calculates, or predicts, a future value by using existing values. The predicted value is a y-value for a given x-value. The known values are existing x-values and y-values, and the new value is predicted by using linear regression.

Mathematical Formula:

FORECAST=a+bx, where:

$$a = \bar{y} - b\bar{x}$$

and:

$$b = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x})^2}$$

where:

- x = a sample value
- $\bar{x}$ is the sample mean `AVERAGE(known-xs)`
- y = a sample value
- $\bar{y}$ is the sample mean `AVERAGE(known-ys)`

Arguments:

Name	Type	Description
x	number	The data point for which a value is to be predicted.
known-xs	array, reference	The independent data.
known-ys	array, reference	The dependent data.

Return Type and Value: number – The future value.

However, if

- *known-xs* and *known-ys* are empty or contain a different number of data points, the return value is unspecified.

- The variance of *known-xs* equals zero, the return value is unspecified.

[*Example*:

`FORECAST(30,{6,7,9,15,21},{20,28,31,38,40})` results in `10.60725309`

end example]

18.17.7.127 FREQUENCY

Syntax:

FREQUENCY (*data-array* , *bins-array*)

Description: Calculates how often values occur within a range of values. A call to FREQUENCY shall be an array formula.

Arguments:

Name	Type	Description
data-array	array, reference to number	Set of values for which frequencies are to be computed. If *data-array* contains no values, FREQUENCY returns an array of zeros. Cells containing text or that are empty are ignored.
bins-array	array, reference	Set of intervals into which the values in *data-array* are to be grouped. If *bins-array* contains no values, FREQUENCY returns the number of elements in *data-array*.

Return Type and Value: vertical array of numbers – The frequency at which values occur within a range of values. The number of elements in the returned array is one more than the number of elements in *bins-array*. The extra element contains the count of any values above the highest interval.

[*Example*:

If the cells A2:A10 contain 79, 85, 78, 85, 50, 81, 95, 88, and 97, and the cells B2:B4 contain 70, 79, and 89, `FREQUENCY(A2:A10,B2:B4)` results in a vertical array containing 1 (50), 2 (79, 78), 4 (85, 85, 81, 88), and 2 (95, 97).

end example]

18.17.7.128 FTEST

Syntax:

FTEST (*array-1* , *array-2*)

Description: Computes the result of an F-test.

Arguments:

Name	Type	Description
array-1	number, name, array, reference to number	If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
array-2		

Return Type and Value: number – The two-tailed probability that the variances in *array-1* and *array-2* are not significantly different.

However, if

- The number of data points in *array-1* or *array-2* is less than 2, the return value is unspecified.
- The variance of *array-1* or *array-2* is zero, the return value is unspecified.

[*Example*:

If the cells D6:D10 contain 6, 7, 9, 15, and 21, and the cells E6:E10 contain 20, 28, 31, 38, and 40, `FTEST(D6:D10,E6:E10)` results in `0.648317847`

end example]

18.17.7.129 FV

Syntax:

FV (*rate* , *nper* , *pmt* [, [*pv*] [, [*type*]]])

Description: Computes the future value of an investment based on periodic, constant payments and a constant interest rate.

Arguments:

Name	Type	Description
rate	number	The interest rate.
nper	number	The total number of payment periods, truncated to integer.
pmt	number	The payment made each period; it cannot change over the life of the annuity. [*Note*: Typically, *pmt* contains principal and interest, but no other fees or taxes. *end note*] If omitted, *pv* shall be provided.
pv	number	The the present value, or the lump-sum amount that a series of future payments is worth right now. If omitted, it is assumed to be 0, and *pmt* shall be provided.
type	number	The timing of the payment, truncated to integer, as

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>follows:
<table>
<tr><th>Value</th><th>Timing</th></tr>
<tr><td><code>0</code></td><td>Payment at the end of the period</td></tr>
<tr><td><code>1</code></td><td>Payment at the beginning of the period</td></tr>
</table>
</td></tr>
</table>

Arguments representing cash paid by investor shall be expressed as negative numbers; arguments representing cash received by the investor shall be expressed as positive numbers.

Return Type and Value: number – The future value of an investment based on periodic, constant payments and a constant interest rate.

However, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`FV(0.06/12,10,-200,-500,1)    2,581.40`
`FV(0.12/12,12,-1000)` results in `12,682.50`
`FV(0.11/12,35,-2000,,1)` results in `82,846.25`
`FV(0.06/12,12,-100,-1000,1)` results in `2,301.40`

end example]

18.17.7.130 FVSCHEDULE

Syntax:

`FVSCHEDULE ( ` *principal* ` , ` *schedule* ` )`

Description: Computes the future value of an initial principal after applying a series of compound interest rates. [*Note*: This function can be used to calculate the future value of an investment with a variable or adjustable rate. *end note*]

Arguments:

Name	Type	Description
principal	number	The present value.
schedule	array	Set of interest rates to apply. The values in this array can be numbers or blank cells. Blank cells are taken as zeros (i.e., no interest).

Return Type and Value: number – The future value of an initial principal after applying a series of compound interest rates.

However, if any element of the array *schedule* is not a number and not blank, #VALUE! is returned.

[*Example*:

FVSCHEDULE(1,{0.09,0.11,0.1}) results in 1.33089

end example]

18.17.7.131 GAMMADIST

Syntax:

GAMMADIST (*x* , *alpha* , *beta* , *cumulative-flag*)

Description: Computes the gamma distribution.

Mathematical Formula:

The equation for the gamma probability density function is:

$$f(x,\alpha,\beta) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{-\frac{x}{\beta}}$$

$$f(x;\alpha,\beta) = \frac{1}{\beta^{\alpha}\Gamma(\alpha)} x^{\alpha-1} e^{-\frac{x}{\beta}}$$

The standard gamma probability density function is:

$$f(x,\alpha) = \frac{x^{\alpha-1}e^{-x}}{\Gamma(\alpha)}$$

When *alpha* = 1, GAMMADIST returns the exponential distribution with:

$$\lambda = \frac{1}{\beta}$$

where:

- α = argument *alpha*
- β = argument *beta*
- *x* = argument *x*

For a positive integer n, when *alpha* = n/2, *beta* = 2, and cumulative = TRUE, GAMMADIST returns (1-CHIDIST(x)) with n degrees of freedom.

When *alpha* is a positive integer, GAMMADIST is also known as the Erlang distribution.

Arguments:

Name	Type	Description
x	number	The value at which the distribution is to be evaluated.
alpha	number	A parameter of the distribution.
beta	number	A parameter of the distribution. If *beta* = 1, GAMMADIST returns the standard gamma distribution.
cumulative-flag	logical	Determines the form of the function. If TRUE, GAMMADIST returns the cumulative distribution function; if FALSE, it returns the probability density function.

Return Type and Value: number – The gamma distribution.

However, if

- $x < 0$, #NUM! is returned.
- $alpha \le 0$ or $beta \le 0$, #NUM! is returned.

[*Example*:

GAMMADIST(10,9,2,FALSE) results in 0.03263902
GAMMADIST(10,9,2,TRUE) results in 0.068093631

end example]

18.17.7.132 GAMMAINV

Syntax:

GAMMAINV (*probability* , *alpha* , *beta*)

Description: Computes the inverse of the gamma distribution. Given a value for *probability*, GAMMAINV seeks that value *x* such that GAMMADIST(*x*, *alpha*, *beta*, TRUE) = *probability*. Thus, the precision of GAMMAINV depends on the precision of GAMMADIST.

Arguments:

Name	Type	Description
probability	number	The probability associated with the gamma distribution.
alpha	number	A parameter of the distribution.
beta	number	A parameter of the distribution. If *beta* = 1, GAMMAINV returns the standard gamma distribution.

Return Type and Value: number – The inverse of the gamma distribution.

However, if

- *probability* < 0 or *probability* > 1, #NUM! is returned.
- *alpha* ≤ 0 or *beta* ≤ 0, #NUM! is returned.
- the implementation determines that a return value cannot be computed, #N/A is returned.

[*Example*:

GAMMAINV(0.068,9,2) results in 9.997130086

end example]

18.17.7.133 GAMMALN

Syntax:

GAMMALN (*x*)

Description: Computes the natural logarithm of the gamma function.

Mathematical Formula:

$$GAMMALN = LN\left(\Gamma(x)\right)$$

where:

$$\Gamma(x) = \int_0^\infty e^{-u} u^{x-1}\, du$$

where:

- x = argument x

Arguments:

Name	Type	Description
x	number	The value for which the gamma function is to be calculated.

Return Type and Value: number – The natural logarithm of the gamma function.

However, if $x \le 0$, #NUM! is returned.

[*Example*:

`GAMMALN(4.5)` results in `2.453736571`

end example]

18.17.7.134 GCD

Syntax:

`GCD (` *argument-list* `)`

Description: Computes the greatest common divisor of the one or more numbers, designated by *argument*s in *argument-list*.

Arguments:

Name	Type	Description
argument-list	numbers	The *argument*s in *argument-list* designate the values. Each argument is truncated to an integer.

Return Type and Value: number – The greatest common divisor of one or more numbers.

However, if any *argument* is negative, `#NUM!` is returned.

[*Example*:

`GCD(5)` results in 5
`GCD(5,2)` results in 1
`GCD(100,50,28)` results in 2
`GCD(24.5,36.3)` results in 12
`GCD(7,1)` results in 1
`GCD(5,0)` results in 5

end example]

18.17.7.135 GEOMEAN

Syntax:

`GEOMEAN (` *argument-list* `)`

Description: Computes the geometric mean of an array or range of positive data.

Mathematical Formula:

$$GM_{\bar{y}} = \sqrt[n]{y_1 y_2 y_3 \dots y_n}$$

where:

- n is the number of elements in *argument-list*.
- $\mathrm{y_1, y_2, y_3, \cdots, y_n}$ are the values of the n-th element in *argument-list*.

Arguments:

Name	Type	Description
argument-list	logical, number, name, array, or reference to number.	The *arguments* in *argument-list* designate the values to be averaged. Logical values and text representations of numbers that entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The geometric mean of an array or range of positive data.

However, if the value of any data point ≤ 0, `#NUM!` is returned.

[*Example*:

`GEOMEAN(10.5,5.3,2.9)` results in `5.444454702`
`GEOMEAN(10.5,{5.3,2.9},"12")` results in `6.633780588`

end example]

18.17.7.136 GESTEP

Syntax:

`GESTEP ( ` *number* ` [ , ` *step* ` ] )`

Description: Tests if the value of *number* is greater than or equal to that of *step*.

Arguments:

Name	Type	Description
number	number	*number* is the value to test against *step*. If *step* is omitted, zero is used.
step	number	

Return Type and Value: number – 1 if *number* ≥ *step*; otherwise, 0.

[*Example*:

`GESTEP(5.6,-4.3)` results in `1`
`GESTEP(5.6,5.6)` results in `1`
`GESTEP(-5.6)` results in `0`

end example]

18.17.7.137 GETPIVOTDATA

Syntax:

GETPIVOTDATA (*data-field* , *pivot-table* , *field-1* , *item-1*
[, *field-2* , *item-2* [, …]])

Description: Retrieves data stored in a PivotTable report. Calculated fields or items and custom calculations are included in `GETPIVOTDATA` calculations.

Arguments:

Name	Type	Description
data-field	text	The name of the data field that contains the data to be retrieved.
pivot-table	reference to any cell, range of cells, or named range of cells in a PivotTable report	This information is used to determine which PivotTable report contains the data to be retrieved. If *pivot-table* is a range that includes two or more PivotTable reports, data shall be retrieved from whichever report was created most recently in the range.
field-1 through *field-n*	text	Argument pairs *field-1* and *item-1*, *field-2* and *item-2* through *field-n* and *item-n* are field names and item names that describe the data to be retrieved. The pairs can be in any order. Field names and names for items other than dates/times (which shall be expressed as numbers) and numbers shall be enclosed in quotation marks. For OLAP PivotTable reports, items can contain the source name of the dimension as well as the source name of the item. [*Example*: A field and item pair for an OLAP PivotTable might look like this: `"[Product]","[Product].[All Products].[Foods].[Baked Goods]"` *end example*] If the field and item arguments describe a single cell, the value of that cell is returned regardless of its value.
item-1 through *item-n*	text	

Return Type and Value: any – The data stored in a PivotTable report.

However, if

- *pivot-table* is not a range in which a PivotTable report is found, the return value is unspecified.
- The arguments do not describe a visible field, the return value is unspecified.
- The arguments include a page field that is not displayed, the return value is unspecified.

[*Example*: Given the following data:

	A	B	C	D	E
2	Region	North			
3					
4	Sum of Sales		Product		
5	Month	Salesperson	Beverages	Produce	Grand Total
6	March	Buchanan	$ 3,522	$ 10,201	$ 13,723
7		Davolio	$ 8,725	$ 7,889	$ 16,614
8	March Total		$ 12,247	$ 18,090	$ 30,337
9	April	Buchanan	$ 5,594	$ 7,265	$ 12,859
10		Davolio	$ 5,461	$ 668	$ 6,129
11	April Total		$ 11,055	$ 7,933	$ 18,988
12	Grand Total		$ 23,302	$ 26,023	$ 49,325

`GETPIVOTDATA("Sales",$A$4)` returns the grand total of the Sales field, `$49,325`.

`GETPIVOTDATA("Sum of Sales",$A$4)` also returns the grand total of the Sales field, `$49,325`; the field name can be entered exactly as it looks on the sheet, or as its root (without "Sum of," "Count of," and so forth).

`GETPIVOTDATA("Sales",$A$4,"Month","March")` returns the grand total for March, `$30,337`.

`GETPIVOTDATA("Sales",$A$4,"Month","March","Product","Produce","Salesperson","Buchanan")` returns `$10,201`.

`GETPIVOTDATA("Sales",$A$4,"Region","South")` is unspecified because the South region data is not visible.

`GETPIVOTDATA("Sales",$A$4,"Product","Beverages","Salesperson","Davolio")` is unspecified because there is no total value of beverage sales for Davolio.

end example]

18.17.7.138 GROWTH

Syntax:

GROWTH (*known-ys* [, [*known-xs*] [, [*new-xs*] [, *const-flag*]])

Description: Computes predicted exponential growth by using existing data. GROWTH can also fit an exponential curve to existing x-values and y-values.

Arguments:

Name	Type	Description
known-ys	array	Set of y-values already known in the relationship y=b*m^x. If the array *known-ys* is a single column, then each column of *known-xs* is interpreted as a separate variable. If the array *known-ys* is a single row, then each row of *known-xs* is interpreted as a separate variable.
known-xs	array	Set of x-values that might already be know in the relationship y=b*m^x. The array *known-xs* can include one or more sets of variables. If only one variable is used, *known-ys* and *known-xs* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, *known-ys* shall be a vector (that is, a *known-ys* with a height of one row or a width of one column). If *known-xs* is omitted, it is assumed to be the array {1,2,3,…} that is the same size as *known-ys*.
new-xs	array	A set of new x-values for which GROWTH is to return corresponding y-values. *new-xs* shall include a column (or row) for each independent variable, just as *known-xs* does. So, if *known-ys* is in a single column, *known-xs* and *new-xs* shall have the same number of columns. If *known-ys* is in a single row, *known-xs* and *new-xs* shall have the same number of rows. If *new-xs* are omitted, it is assumed to be the array {1,2,3,…} that is the same size as *known-ys*.
const-flag	logical	Specifies whether to force the constant b to equal 1. If TRUE or omitted, *b* is calculated normally. If FALSE, *b* is set equal to 1 and the m-values are adjusted so that y= m^x.

Return Type and Value: array – The y-values for a series of new x-values.

However, if any of the numbers in *known-ys* are zero or negative, #NUM! is returned.

[*Example*: Given the following data:

	A	B	C
1	Month	Units	Formula (corresponding units)
2	11	33,100	32618.20377
3	12	47,300	47729.42261
4	13	69,000	69841.30086
5	14	102,000	102197.0734
6	15	150,000	149542.4867

	A	B	C
7	16	220,000	218821.8762
8	Month	Formula (Predicted Units)	
9	17	320,196.72	
10	18	468,536.05	

When `GROWTH(A2:B4,A6:B8)` is an array formula spanning cells C2:C7, those cells take on the results shown. When `GROWTH(A2:B4,A6:B8,A9:A10)` is an array formula spanning cells B9:B10, those cells take on the the results shown.

end example]

18.17.7.139 HARMEAN

Syntax:

HARMEAN (*argument-list*)

Description: Computes the harmonic mean of a data set.

Mathematical Formula:

$$\frac{1}{H} = \frac{1}{n}\sum\frac{1}{Y_i}$$

where:

- n = number of elements in *argument-list*
- Y_i = each element in *argument-list*

Arguments:

Name	Type	Description
argument-list	logical, text, number, name, array, or reference to number.	The *arguments* in *argument-list* designate the values to be averaged. Argument values can be numbers, or names, arrays, or references that contain numbers. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The harmonic mean of a data set.

However, if the value of any data point ≤ 0, `#NUM!` is returned.

[*Example*:

`HARMEAN(4.6,5.8,8.3,7)` results in `6.124222`
`HARMMEAN(10.5,{5.3,2.9},"12")` results in `5.617360`

end example]

18.17.7.140 HEX2BIN

Syntax:

`HEX2BIN (` *number* `[ ,` *num-bin-digits* `] )`

Description: Makes the binary equivalent of *number*, with the result having *num-bin-digits* digits.

Arguments:

Name	Type	Description
number	text	A 10-digit hexadecimal number in a string that is to be converted to a binary string. *number* is not case-sensitive. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (40th bit from the right) representing the sign bit.
num-bin-digits	number	The number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-bin-digits* is ignored and the result has 10 digits. If *num-bin-digits* is omitted, the minimum number of digits is used in the result. *num-bin-digits* is truncated to an integer.

Return Type and Value: text – The binary equivalent of *number*.

However, if

- *number* is outside the range `"FFFFFFFE00"` (1111111111111111111111111111111000000000 binary, -512 decimal) to `"1FF"` (0000000000000000000000000000000111111111 binary, 511 decimal), inclusive, `#NUM!` is returned.
- *number* contains one or more non-hexadecimal digits, `#NUM!` is returned.
- *number* contains more than 10 hexadecimal digits, `#NUM!` is returned.
- *number* needs more digits that *num-bin-digits*, `#NUM!` is returned.
- *num-bin-digits* is negative or > 10, `#NUM!` is returned.

[*Example*:

`HEX2BIN("fE")` results in `11111110`
`HEX2BIN("FFFFFFFFFE")` results in `1111111110`
`HEX2BIN("2")` results in `10`
`HEX2BIN("F",6)` results in `001111`

end example]

18.17.7.141 HEX2DEC

Syntax:

`HEX2DEC ( number )`

Description: Makes the decimal equivalent of *number*.

Arguments:

Name	Type	Description
number	string	A 10-digit hexadecimal number in a string that is to be converted to a decimal number. *number* is not case-sensitive. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (40th bit from the right) representing the sign bit.

Return Type and Value: number – The decimal equivalent of *number*.

However, if

- *number* contains one or more non-hexadecimal digits, `#NUM!` is returned.
- *number* contains more than 10 hexadecimal digits; that is, *number* is outside the range `"8000000000"` (-548,755,813,888 decimal) to `"7FFFFFFFFF"` (548,755,813,887 decimal), inclusive, `#NUM!` is returned.

[*Example*:

`HEX2DEC("fE")` results in `254`
`HEX2DEC("FFFFFFFFFE")` results in `-2`
`HEX2DEC("F000000000")` results in `-68719476736`

end example]

18.17.7.142 HEX2OCT

Syntax:

HEX2OCT (*number* [, *num-oct-digits*])

Description: Makes the octal equivalent of *number*, with the result having *num-oct-digits* digits.

Arguments:

Name	Type	Description
number	text	A 10-digit hexadecimal number in a string that is to be converted to an octal string. *number* is not case-sensitive. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (40th bit from the right) representing the sign bit.
num-oct-digits	number	The number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-oct-digits* is ignored and the result has 10 digits. If *num-oct-digits* is omitted, the minimum number of digits is used in the result. *num-oct-digits* is truncated to an integer.

Return Type and Value: text – The octal equivalent of *number*.

However, if

- *number* is outside the range `"FFE0000000"` (17774000000000 octal, -536,870,912 decimal) to `"1FFFFFFF"` (00003777777777 octal, 536,870,911 decimal), inclusive, `#NUM!` is returned.
- *number* contains one or more non-hexadecimal digits, `#NUM!` is returned.
- *number* contains more than 10 hexadecimal digits, `#NUM!` is returned.
- *number* needs more digits that *num-oct-digits*, `#NUM!` is returned.
- *num-oct-digits* is negative or > 10, `#NUM!` is returned.

[*Example*:

`HEX2OCT("fE")` results in `376`
`HEX2OCT("FFFFFFFFFE")` results in `7777777776`
`HEX2OCT("2")` results in `2`
`HEX2OCT("F",6)` results in `000017`

end example]

18.17.7.143 HLOOKUP

Syntax:

HLOOKUP (*lookup-value* , *table-array* , *row-index-num* [, [*range-lookup-flag*]])

Description: Performs a horizontal search for a value in the top row of a table or an array, noting the column in which the matching value is found. From that column, the value from a given row is returned.

Arguments:

Name	Type	Description
lookup-value	value or reference.	The value to be located in the first row of the table. If *range-lookup* is FALSE and *lookup-value* is a string, the wildcard characters, question mark (?) and asterisk (*), can be included in *lookup-value*. A question mark matches any single character; an asterisk matches any sequence of characters. To find a question mark or asterisk, use a tilde (~) before the character.
table-array	array, reference, name	Designates the table of information to be searched. The values in the first row of *table-array* can be text, numbers, or logical values. If *range-lookup-flag* is TRUE, the values in the first row of *table-array* shall be placed in "ascending order", as follows: …, -2, -1, 0, 1, 2, …, A–Z, FALSE, TRUE. If *range-lookup-flag* is FALSE, *table-array*'s values need not be sorted. Uppercase and lowercase text is treated as equivalent.
row-index-num	number	The row number in *table-array* from which the matching value is to be returned. (A *row-index-num* of 1 returns the first row value in *table-array*, a *row-index-num* of 2 returns the second row value in *table-array*, and so on.)
range-lookup-flag	logical	Specifies whether HLOOKUP is to find an exact or approximate match. If TRUE or omitted, an approximate match is returned. That is, if an exact match is not found, the next largest value that is less than *lookup-value* is returned. If FALSE, an exact match is performed.

Return Type and Value: any – The value from a given row number, where the column is determined by a search of the top row looking for a match with a given value.

However, if

- An exact match is performed, but no match is found, #N/A is returned.
- *row-index-num* is less than 1, #VALUE! is returned.
- *row-index-num* is greater than the number of rows in *table-array*, #REF! is returned.
- *lookup-value* is smaller than the smallest value in the first row of *table-array*, #N/A is returned.

[*Example*: Given the following data:

	A	B	C
1	Axles	Bearings	Bolts
2	4	6	9
3	5	7	10
4	6	8	11

`HLOOKUP("Axles",A1:C4,2,TRUE)` results in 4
`HLOOKUP("Bearings",A1:C4,3,FALSE)` results in 7
`HLOOKUP("B",A1:C4,3,TRUE)` results in 5
`HLOOKUP("Bolts",A1:C4,4)` results in 11
`HLOOKUP(3,{1,2,3;"a","b","c";"d","e","f"},2,TRUE)` results in c

end example]

18.17.7.144 HOUR

Syntax:

`HOUR ( time-value )`

Description: Computes the hour for the date and/or time having the given *time-value*.

Arguments:

Name	Type	Description
time-value	number	The date and/or time whose hour is to be computed. That date and/or time shall be expressed either as a serial date-time, in which case, its integer part is ignored, or as a *string-constant* having any date and/or time format, in which case, any date information shall be ignored.

Return Type and Value: number – The hour for the date and/or time having the given *time-value*. The returned value shall be in the range 0–23.

However, if *time-value* is out of range for the current date system, `#NUM!` is returned.

[*Example*:

`HOUR(DATE(2006,2,26)+TIME(2,10,20))` results in 2
`HOUR(TIME(22,56,34))` results in 22
`HOUR(0)` results in 0, since serial date-time 0 represents `00:00:00`
`HOUR(10.5)` results in 12, since serial date-time .5 represents `12:00:00`

`HOUR("22-Oct-2001 10:53:12")` results in `10`
`HOUR("10:53:12 pm")` results in 22
`HOUR("22:53:12")` results in 22

end example]

18.17.7.145 HYPERLINK

Syntax:

`HYPERLINK ( `*link-location*` [ , [ `*friendly-name*` ]] )`

Description: Creates a shortcut that opens a document stored on a network server, an intranet, or the Internet. When the cell that contains the `HYPERLINK` function call is clicked, the file stored at *link-location* is opened.

Arguments:

Name	Type	Description
link-location	text	The location to be opened as text. *link-location* can refer to a place in a document—such as a specific cell or named range in a SpreadsheetML worksheet or workbook, or to a bookmark in a WordprocessingML document. The *link-location* can be to a location on a local hard disk drive, the path to a location on a server or a Uniform Resource Locator (URL) to a location on the Internet or an intranet. [*Note*: Using a URL is the preferred way of specifying *link-location*. *end note*] If the location specified in *link-location* does not exist or cannot be navigated, an unspecified error is produced when the cell is clicked. *link-location* can be a string or a reference to a cell containing a string.
friendly-name	text, number, name	The value that is displayed in the cell. If omitted, the cell displays *link-location*. *friendly-name* can be a value, a text string, a name, or a cell that contains the jump text or value. If the evaluation of *friendly-name* results in an error value, the cell displays that error value rather than the jump text.

Return Type and Value: text – The value of *friendly-name*, if it is specified; otherwise, the value of *link-location*.

[*Example*:

`HYPERLINK("http://example.openxmlformats.org/report/budget report.xls","Click for report")`, which opens a worksheet named "budget report.xls" that is stored on the Internet at the location example.openxmlformats.org/report, and displays the text "Click for report".

`HYPERLINK("D:\FINANCE\1stqtr.xls",H10)`, which opens the file 1stqtr.xls that is stored in a directory named Finance on drive D, and displays the numeric value stored in cell `H10`.

end example]

18.17.7.146 HYPGEOMDIST

Syntax:

> `HYPGEOMDIST (` *sample-successes* `,` *number-sample* `,` *population-successes* `,` *number-population* `)`

Description: Computes the hypergeometric distribution; that is, the probability of a given number of sample successes, given the sample size, population successes, and population size.

Mathematical Formula:

$$P(X = x) = h(x, n, M, N) = \frac{\binom{M}{x}\binom{N-M}{n-x}}{\binom{N}{n}}$$

where:

- M = *population-successes* argument
- N = *number-population* argument
- x = *sample-successes* argument
- n = *number-sample* argument

Arguments:

Name	Type	Description
sample-successes	number	The number of successes in the sample, truncated to integer.
number-sample	number	The size of the sample, truncated to integer.
population-successes	number	The number of successes in the population, truncated to integer.
number-population	number	The population size, truncated to integer.

Return Type and Value: number – The hypergeometric distribution.

However, if

- *sample-successes* < 0 or *sample-successes* is greater than the lesser of *number-sample* and *population-successes*, `#NUM!` is returned.

- *sample-successes* is less than the larger of 0 or (*number-sample* - *number-population* + *population-successes*), #NUM! is returned.
- *number-sample* ≤ 0 or *number-sample* > *number-population*, #NUM! is returned.
- *population-successes* ≤ 0 or *population-successes* > *number-population*, #NUM! is returned.
- *number-population* ≤ 0, #NUM! is returned.

[*Example*:

HYPGEOMDIST(1,4,8,20) results in 0.363261

end example]

18.17.7.147 IF

Syntax:

IF (*logical-value* , [*value-if-true*] [, [*value-if-false*]])

Description: Tests *logical-value*, and if it is TRUE, *value-if-true* is evaluated and returned; otherwise, *value-if-false* is evaluated and returned.

Arguments:

Name	Type	Description
logical-value	logical	The value to be tested.
value-if-true	any	The value returned if *logical-value* is TRUE. If *logical-value* is TRUE and *value-if-true* is omitted, this argument evaluates to 0. *value-if-true* can contain up to seven levels of nested IF function calls. [*Note*: *value-if-true* and *value-if-false* need not evaluate to results of the same type. *end note*]
value-if-false	any	The value returned if *logical-value* is FALSE. If *logical-value* is FALSE and *value-if-false* and its preceding comma is omitted, this argument evaluates to FALSE. If *logical-value* is FALSE and *value-if-false* is omitted, but its preceding comma is present, this argument evaluates to 0. *value-if-false* can contain at least seven levels of nested IF function calls. [*Note*: *value-if-true* and *value-if-false* need not evaluate to results of the same type. *end note*]

If any argument is an array, every element of that array shall be evaluated when that argument is evaluated.

Return Type and Value: any – *value-if-true*, if *logical-value* is TRUE; otherwise, *value-if-false*.

[*Example*:

`IF(10>5,"Yes","No")` results in `Yes`
`IF(10>5,"Yes")` results in `Yes`
`IF(10>5,"Yes",)` results in `Yes`
`IF(10<5,"Yes")` results in `FALSE`
`IF(10<5,"Yes",)` results in `0`

`IF(10>5,,"No")` results in `0`
`IF(10>5,,)` results in `0`
`IF(10>5,"Yes",20)` results in `Yes`
`IF(10<5,"Yes",20)` results in `20`

end example]

18.17.7.148 IFERROR

Syntax:

`IFERROR (` *value* `,` *value-if-error* `)`

Description: Provides a simpler and more efficient way of trapping and handling errors. It allows the generation of user-defined error text for a function call that can result in an error.

Arguments:

Name	Type	Description
value	any	The value that is checked for an error (i.e., any of the following: `#N/A`, `#VALUE!`, `#REF!`, `#DIV/0!`, `#NUM!`, `#NAME?`, or `#NULL!`). If *value* is an empty cell, it is treated as an empty string.
value-if-error	any	The value to return if *value* evaluates to an error. If *value-if-error* is an empty cell, it is treated as an empty string.

Return Type and Value: any – *value*, if *value* is not an error; otherwise, *value-if-error*. If *value* is an array formula, an array of results for each cell in the range specified in value, is returned.

[*Example*: Consider the case in which A3 contains 55, and B3 contains 0:

`A3/B3` results in `#DIV/0`

`IFERROR(A3/B3,"Error in calculation")` results in `Error in calculation`

end example]

18.17.7.149 IMABS

Syntax:

`IMABS ( ` *complex-number* ` )`

Description: Computes the absolute value of *complex-number*.

Mathematical Formula:

$$IMABS(z) = |z| = \sqrt{x^2 + y^2}$$

where:

- z = argument *complex-number*, expressed in the form x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the absolute value is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: number – The absolute value of *complex-number*.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMABS("3+4i")` results in 5
`IMABS("-2.5-34.6j")` results in `34.69020035`

end example]

18.17.7.150 IMAGINARY

Syntax:

`IMAGINARY ( ` *complex-number* ` )`

Description: Computes the imaginary coefficient of *complex-number*.

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the imaginary coefficient is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: number – The imaginary coefficient of *complex-number*.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMAGINARY("3+4i")` results in 4
`IMAGINARY("-2.5-34.6j")` results in `34.6`

end example]

18.17.7.151 IMARGUMENT

Syntax:

`IMARGUMENT (` *complex-number* `)`

Description: Computes the argument θ, an angle expressed in radians, such that for a complex number *complex-number* having the form *x+yi*:

$$x + yi = |x + yi| \times e^{i\theta} = |x + yi|(\cos\theta + i\sin\theta)$$

Mathematical Formula:

$$IMARGUMENT(z) = \tan^{-1}\left(\frac{y}{x}\right) = \theta$$

where:

- $\theta \in (-\pi; \pi]$
- z = argument *complex-number*, expressed in the form x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number in *x + yi* or *x + yj* text format.

Return Type and Value: number – The angle θ, expressed in radians.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMARGUMENT("13+4i")` results in `0.298498932`
`IMARGUMENT("-2.5-5j")` results in `-2.034443936`

end example]

18.17.7.152 IMCONJUGATE

Syntax:

`IMCONJUGATE ( ` *complex-number* ` )`

Description: Computes the complex conjugate of the complex number *complex-number.*

Mathematical Formula:

$$ICONJUGATE(x + yi) = \bar{z} = (x - yi)$$

where:

- $\bar{z}$ = conjugate of the argument *complex-number*, expressed in the form x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the complex conjugate is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the complex conjugate of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMCONJUGATE("2.3+4.5i")` results in `2.3-4.5i`
`IMCONJUGATE("-1-4j")` results in `-1+4j`

end example]

18.17.7.153 IMCOS

Syntax:

`IMCOS ( ` *complex-number* ` )`

Description: Computes the cosine of the complex number *complex-number.*

Mathematical Formula:

$$\cos(x + yi) = \cos(x)\cosh(y) - \sin(x)\sinh(y)i$$

where:

- the argument *complex-number is* expressed in the form x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the cosine is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the cosine of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMCOS("2.3+4.5i")` results in `-29.9918288739746-33.5589799796873i`
`IMCOS("-1-4j")` results in `14.7547011704838-22.963673499193j`

end example]

18.17.7.154 IMDIV

Syntax:

`IMDIV (` *complex-number-1* `,` *complex-number-2* `)`

Description: Computes the quotient from dividing two complex numbers.

Mathematical Formula:

$$IMDIV(z_1, z_2) = \frac{(a + bi)}{(c + di)} = \frac{(ac + bd) + (bc - ad)i}{c^2 + d^2}$$

where:

- z_1 = argument *complex-number-1* expressed in the form of a + bi
- z_2 = argument *complex-number-2* expressed in the form of c + di

Arguments:

Name	Type	Description
complex-number-1	text	Complex numbers in *x + yi* or *x + yj* text format; they designate the dividend and divisor, respectively.
complex-number-2	text	

Return Type and Value: text – A string containing the quotient from *number-1* / *number-2*, in *x+yi* or *x+yj* text format.

However, if *complex-number-1* or *complex-number-2* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMDIV("13+4i","5+3i")` results in `2.26470588235294-0.558823529411765i`
`IMDIV("-3-3.5i","5+3i")` results in `-0.75-0.25i`

end example]

18.17.7.155 IMEXP

Syntax:

`IMEXP ( `*complex-number*` )`

Description: Computes the exponential of the complex number *complex-number.*

Mathematical Formula:

$$IMEXP(\quad) = e^{(x+yi)} = e^x e^{yi} = e^x(\cos y + i \sin y)$$

where:

- z = the argument *complex-number*, expressed in the form x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the exponential is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the exponential of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMEXP("2.3+4.5i")` results in `-2.10251576423113-9.75006374866818i`
`IMEXP("-1-4j")` results in `-0.240462049968584+0.278412079051034j`

end example]

18.17.7.156 IMLN

Syntax:

`IMLN ( ` *complex-number* ` )`

Description: Computes the natural logarithm of *complex-number*.

Mathematical Formula:

$$\ln(x + yi) = \ln\sqrt{x^2 + y^2} + i\tan^{-1}\left(\frac{y}{x}\right)$$

where:

- the argument *complex-number* is expressed in the form x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the natural logarithm is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – The natural logarithm of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMLN("3+4i")` results in `1.6094379124341+0.927295218001612i`
`IMLN("-2.5-34.6j")` results in `3.54645723627033-1.64292531532225j`

end example]

18.17.7.157 IMLOG10

Syntax:

`IMLOG10 ( ` *complex-number* ` )`

Description: Computes the base-10 logarithm of *complex-number*.

Mathematical Formula:

The common logarithm of a complex number can be calculated from the natural logarithm as follows:

$\log_{10}(x + yi) = (\log_{10} e) \ln(x + yi)$

where:

- the argument *complex-number* is expressed in the form x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the base-10 logarithm is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – The base-10 logarithm of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMLOG10("3+4i")` results in `10.698970004336019+0.402719196273373i`
`IMLOG10("-2.5-34.6j")` results in `11.54020680801806-0.713513398623614j`

end example]

18.17.7.158 IMLOG2

Syntax:

`IMLOG2 (` *complex-number* `)`

Description: Computes the base-2 logarithm of *complex-number*.

Mathematical Formula:

The base-2 logarithm of a complex number can be calculated from the natural logarithm as follows:

$\log_2(x + yi) = (\log_2 e) \ln(x + yi)$

where:

- the argument *complex-number* is expressed in the form x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the base-2 logarithm is being computed. *complex-number* shall be in *x + yi* or

Name	Type	Description
		x + yj text format.

Return Type and Value: text – The base-2 logarithm of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

IMLOG2("3+4i") results in `2.32192809506607+1.337804212553394i`
IMLOG2("-2.5-34.6j") results in `5.11645626788577-2.37024020514877j`

end example]

18.17.7.159 IMPOWER

Syntax:

`IMPOWER (` *complex-number* `,` *y* `)`

Description: Computes the complex number *complex-number* raised to the power *y*.

Mathematical Formula:

$$(x + yi)^n = r^n e^{in\theta} = r^n \cos\theta + ir^n \sin n$$

where:

$$r = \sqrt{x^2 + y^2}$$

and:

$$= \tan^{-1}\left(\frac{y}{x}\right)$$

and:

$$\theta \in (-\pi; \pi]$$

where:

- argument *complex-number* is expressed in the form x + yi
- n = argument y

Arguments:

Name	Type	Description
complex-number	text	The complex number in *x + yi* or *x + yj* text format.
y	number	The exponent to which *complex-number* is to be raised.

Return Type and Value: text – A string containing *complex-number*^{y}, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMPOWER("2.3+4.5i",2.5)` results in `-52.9752689709953+22.1385284639541`
`IMPOWER("-1-4j",-3.56)` results in `6.34818926783845E-003+1.16156377299512E-003j`

end example]

18.17.7.160 IMPRODUCT

Syntax:

`IMPRODUCT (` *argument-list* `)`

Description: Multiplies the values of its complex number arguments.

Mathematical Formula:

$$(a + bi)(c + di) = (ac - bd) + (ad + bc)i$$

where:

- Each element of *argument-list* is a complex number expressed in the form x + yi

Arguments:

Name	Type	Description
argument-list	text	Each *argument* in *argument-list* is a complex number string in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the product of the values of its arguments, in *x+yi* or *x+yj* text format.

However, if any *argument* in *argument-list* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMPRODUCT("13+4i")` results in `13+4i`
`IMPRODUCT("-3-3.5i","5+3i")` results in `-4.5-26.5i`
`IMPRODUCT("1.3-2j","-3.4+3j","2.3-6j")` results in `67.834+15.13j`

end example]

18.17.7.161 IMREAL

Syntax:

`IMREAL (` *complex-number* `)`

Description: Computes the real coefficient of *complex-number*.

where:

- argument *complex-number* is expressed in the form x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the real coefficient is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: number – The real coefficient of *complex-number*.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMREAL("3+4i")` results in 3
`IMREAL("-2.5-34.6j")` results in `-2.5`

end example]

18.17.7.162 IMSIN

Syntax:

`IMSIN (` *complex-number* `)`

Description: Computes the sine of the complex number *complex-number*.

Mathematical Formula:

$$\sin(x + yi) = \sin(x)\cosh(y) - \cos(x)\sinh(y)i$$

where:

- argument *complex-number* is expressed in the form x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the sine is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the sine of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMSIN("2.3+4.5i")` results in `33.567264016308-29.9844272159606i`
`IMSIN("-1-4j")` results in `-22.9790855778861-14.7448051885587j`

end example]

18.17.7.163 IMSQRT

Syntax:

`IMSQRT (` *complex-number* `)`

Description: Computes the square root of the complex number *complex-number.*

Mathematical Formula:

$$\sqrt{x+yi} = \sqrt{r}\cos\left(\frac{\theta}{2}\right) + i\sqrt{r}\sin\left(\frac{\theta}{2}\right)$$

where:

$$r = \sqrt{x^2 + y^2}$$

and:

$$\theta = \tan^{-1}\left(\frac{y}{x}\right)$$

and:

$$\theta \in (-\pi; \pi]$$

where:

- argument *complex-number* is expressed in the form x + yi

Arguments:

Name	Type	Description
complex-number	text	The complex number for which the square root is being computed. *complex-number* shall be in *x + yi* or *x + yj* text format.

Return Type and Value: text – A string containing the square root of *complex-number*, in *x+yi* or *x+yj* text format.

However, if *complex-number* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMSQRT("2.3+4.5i")` results in `1.91751290835255+1.17339496918073i`
`IMSQRT("-1-4j")` results in `1.24962106768765-1.60048518044024j`

end example]

18.17.7.164 IMSUB

Syntax:

`IMSUB (` *complex-number-1* `,` *complex-number-2* `)`

Description: Computes the difference of two complex numbers.

Mathematical Formula:

$$(a + bi) - (c + di) = (a - c) + (b - d)i$$

$$(a + bi) - (c + di) = (a - c) + (b - d)i$$

where:

- argument *complex-number-1* is expressed in the form a + bi
- argument *complex-number-2* is expressed in the form c + di

Arguments:

Name	Type	Description
complex-number-1	text	Complex numbers in *x + yi* or *x + yj* text format; they designate the minuend and subtrahend, respectively.
complex-	text	

Name	Type	Description
number-2		

Return Type and Value: text – A string containing *number-1* - *number-2*, in *x+yi* or *x+yj* text format.

However, if *complex-number-1* or *complex-number-2* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMSUB("13+4i","5+3i")` results in `8+i`
`IMSUB("-3-3.5i","5+3i")` results in `-8-6.5i`

end example]

18.17.7.165 IMSUM

Syntax:

`IMSUM (` *argument-list* `)`

Description: Adds the values of its arguments.

Mathematical Formula:

$$(a + bi) + (c + di) = (a + c) + (b + d)i$$

where:

- Each element of *argument-list* is a complex number expressed in the form x + yi

Arguments:

Name	Type	Description
argument-list	text	Each *argument* in *argument-list* is a complex number string in *x + yi* or *x + yj* text format.

Return Type and Value: text – The sum of the values of its arguments, in *x+yi* or *x+yj* text format.

However, if any *argument* in *argument-list* is ill-formed, `#NUM!` is returned.

[*Example*:

`IMSUM("3+4i")` results in `3+4i`
`IMSUM("3+4i","5-3i")` results in `8+i`

end example]

18.17.7.166 INDEX

Syntax:

array form: `INDEX` (*array* , [*row-number*] [, [*column-number*]])

reference form: `INDEX` (*reference* [, [*row-number*] [, [*column-number*] [, [*area-number*]]]])

Description: Locates a value or the reference to a value from within a table or range. There are two forms of the `INDEX` function: the array form and the reference form.

Arguments:

Name	Type	Description
array	array, reference	Table or range to be searched. If *array* contains only one row, the corresponding *row-number* argument is optional. If *array* contains only one column, the corresponding *column-number* argument is optional.
reference	reference	A reference to one or more cell ranges. If each area (§18.17.2.3) in *reference* contains only one row, *row-number* is optional. If each area contains only one column, *column-number* is optional.
row-number	number	*row-number* indicates the row in *array* (or *reference*) from which to return a value (or reference). If *row-number* is omitted, *column-number* shall be present. *column-number* indicates the column in *array* (or *reference*) from which to return a value (or *reference*). If *column-number* is omitted, *row-number* shall be present. If both the *row-number* and *column-number* arguments are used, `INDEX` returns the value (or *reference*) in the cell at the intersection of *row-number* and *column-number*. If *array* has more than one row and more than one column, and only *row-number* or *column-number* is used, `INDEX` returns an array of the entire row or column in array. If *row-number* or *column-number*, but not both, is 0, `INDEX` returns the array of values for the entire column or row, respectively. In the reference form, if *row-number* and *column-number* are both omitted, `INDEX` returns the area in reference specified by *area-number*.
column-number	number	
area-number	number	Indicates a range in *reference* from which to return the intersection of *row-number* and *column-number*. The first area selected or entered is numbered 1, the second 2, and so on. If *area-number* is omitted, 1 is assumed.

Name	Type	Description
		[*Example*: If *reference* describes the cells `(A1:B4, D1:E4, G1:H4)`, then area-number 1 is the range `A1:B4`, *area-number* 2 is the range `D1:E4`, and *area-number* 3 is the range `G1:H4`. *end example*]

Return Type and Value: various – For the array form, returns a single value, a whole row, or a whole column from a table or an array, depending on the presence and values of the row and column number indexes.

For the reference form, returns a single reference, a whole row, or a whole column from a reference, depending on the presence and values of the row and column number indexes, and the area number.

However, for the array form

- *row-number* is outside the bounds of *array*, `#REF!` is returned.
- *column-number* is outside the bounds of *array*, `#REF!` is returned.

For the reference form

- *row-number* is outside the bounds of *reference*, `#REF!` is returned.
- *column-number* is outside the bounds of *reference*, `#REF!` is returned.
- *area-number* is outside the bounds of *reference*, `#REF!` is returned.

[*Example*:

`INDEX({"Apples","Lemons";"Bananas","Pears"},2,2)` results in `Pears`
`INDEX({"Apples","Lemons";"Bananas","Pears"},2,1)` results in `Bananas`
`INDEX({"Apples","Lemons"},,2)` results in `Lemons`
`INDEX({"Apples";"Bananas"},1)` results in `Apples`

Given the following data:

	A	B	C
1	Fruit	Price	Count
2	Apples	0.69	40
3	Bananas	0.34	38
4	Lemons	0.55	15
5	Oranges	0.25	25
6	Pears	0.59	40
7	Almonds	2.8	10

`INDEX(A2:C7,2,3)` results in 38
`INDEX((A2:C4,A6:C7),2,2,2)` results in `2.8`
`INDEX((A2:C4,A6:C7),2,2,1)` results in `0.34`

end example]

18.17.7.167 INDIRECT

Syntax:

`INDIRECT (` *ref-text* `[ , [` *A1-ref-style-flag* `]] )`

Description: Locates the reference specified by *ref-text* and evaluates that reference to get to its underlying value. [*Note*: This function should be used when the reference to a cell within a formula is to be changed without changing the formula itself. *end note*]

Arguments:

Name	Type	Description
ref-text	An A1-style reference, an R1C1-style reference, a name defined as a reference, or a reference to a cell as a string.	If *ref-text* refers to another workbook (i.e., it's an external reference), that other workbook shall be open.
A1-ref-style-flag	logical	Specifies the kind of reference that is contained in the cell *ref-text*. If TRUE or omitted, *ref-text* is interpreted as an A1-style reference (§18.17.2.3.1); otherwise, *ref-text* is interpreted as an R1C1-style reference (§18.17.2.3.2).

Return Type and Value: any – The underlying value of the location referred to by *ref-text*.

However, if

- *ref-text* is not a cell reference, `#REF!` is returned.
- *ref-text* refers to another workbook yet that other workbook is not currently open, the return value is unspecified.
- *ref-text* is a name, then the content of the named expression can have implementation-defined constraints.

[*Example*:

Given the following data:

	A	B
1	Data	Data
2	B2	1.333
3	B3	45
4	George	10
5	5	62

where A2 contains a reference to B2, A3 contains a reference to B3, A4 contains the defined name George that refers to B4, and A5 contains the row number of B5:

`INDIRECT($A$2)` results in `1.333`
`INDIRECT($A$3)` results in 45
`INDIRECT($A$4)` results in `10`
`INDIRECT("B"&$A$5)` results in 62

`INDIRECT("R[-1]C",FALSE)` uses the cell in the previous row and current column.
end example]

18.17.7.168 INFO

Syntax:

`INFO ( `*category*` )`

Description: Retrieves the operating environment value that corresponds to *category*.

Arguments:

Name	Type	Description
category	text	The string designated by *category* is not case-sensitive. The permitted strings are shown in the table below.

category	Meaning	Result Type
`"directory"`	Path of the current directory or folder.	text
`"memavail"`	Amount of memory available, in bytes.	number
`"memused"`	Amount of memory being used for data.	number
`"numfile"`	Number of active worksheets in the open workbooks.	number
`"origin"`	The absolute cell reference of the top and leftmost cell visible in the window, based on the current scrolling position, prefixed with `"$A:"`. [*Example*: Using cell D9 as an example, the return value would be $A:$D$9. *end example*]	text

category	Meaning	Result Type
"osversion"	Current operating system version.	text
"recalc"	Current recalculation mode: "Automatic" or "Manual"	text
"release"	Version of the implementation.	text
"system"	Name of the operating environment.	text
"totmem"	Total memory available, including memory already in use, in bytes.	number

Return Type and Value: text – The operating environment value that corresponds to *category*.

However, if *category* is not one of the values defined above, #VALUE! is returned.

[*Example*:

INFO("directory") might result in e:\My Documents\
INFO(A10) might result in e:\My Documents\, where A10 contains directory
INFO("memavail") might result in 1048576

INFO("memused") might result in 1474464
INFO("numfile") might result in 5
INFO("origin") might result in $A:$C$536

INFO("osversion") might result in Windows (32-bit) NT 5.01
INFO("recalc") might result in Automatic
INFO("release") might result in 11.0

INFO("system") might result in pcdos
INFO("totmem") might result in 2523040

end example]

18.17.7.169 INT

Syntax:

INT (*x*)

Description: Computes *x* rounded down to an integer.

Arguments:

Name	Type	Description
x	number	The value to be rounded down.

Return Type and Value: number – The rounded-down value of x.

[*Example*:

`INT(8.9)` results in `8`
`INT(-8.9)` results in `-9`

end example]

18.17.7.170 INTERCEPT

Syntax:

`INTERCEPT ( ` *known-ys* ` , ` *known-xs* ` )`

Description: Computes the point at which a line intersects the y-axis by using existing x-values and y-values. The intercept point is based on a best-fit regression line plotted through the known x-values and known y-values.

Mathematical Formula:

The equation for the intercept of the regression line, a, is:

$$a = \bar{y} - b\bar{x}$$

where the slope, b, is calculated as:

$$b = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sum(x-\bar{x})^2}$$

where:

- x = a sample value
- $\bar{x}$ = the sample mean `AVERAGE(`*known-xs*`)`
- y = a sample value
- $\bar{y}$ = the sample mean `AVERAGE(`*known-ys*`)`

Arguments:

Name	Type	Description
known-ys	number, name, array, reference to number	The dependent set of observations or data. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
known-xs	number, name, array, reference to number	The independent set of observations or data. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The point at which a line intersects the y-axis by using existing x-values and y-values.

However, if

- *known-ys* and *known-xs* contain a different number of data points, the return value is unspecified.
- *known-ys* or *known-xs* contain no data points, the return value is unspecified.

[*Example*:

`INTERCEPT({2,3,9,1,8},{6,5,11,7,5})` results in `0.048387097`

end example]

18.17.7.171 INTRATE

Syntax:

INTRATE (*settlement* , *maturity* , *investment* , *redemption* [, [*basis*]])

Description: Computes the interest rate for a fully invested security.

Mathematical Formula:

$$INTRATE = \frac{redemption - nvestment}{investment} \times \frac{B}{DIM}$$

where:

- B = number of days in a year, depending on the year basis
- DIM = number of days from settlement to maturity.
- $investment$ = argument $investment$
- $redemption$ = argument $redemption$

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
investment	number	The amount invested in the security.
redemption	number	The amount to be received at maturity.he security's annual yield.
basis	number	The truncated integer type of day count basis to use, as follows: **Value** | **Day Count Basis**

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td><code>0</code> or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td><table><tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr></table></td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The interest rate for a fully invested security.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *investment* or *redemption* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`INTRATE(DATE(2008,2,15),DATE(2008,5,15),1000000,1014420,2)` results in `5.7680%`

end example]

18.17.7.172 IPMT

Syntax:

`IPMT ( `*rate*` , `*per*`, `*nper*` , `*pv*` , [ `*fv*` ] [ , [ `*type*` ]] )`

Description: Computes the interest payment for a given period for an investment based on periodic, constant payments and a constant interest rate.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>rate</td><td>number</td><td>The interest rate.</td></tr>
<tr><td>per</td><td>number</td><td>The period for which the interest is to be found, and shall be in the range 1–nper.</td></tr>
<tr><td>nper</td><td>number</td><td>The total number of payment periods in an annuity.</td></tr>
<tr><td>pv</td><td>number</td><td>The present value, or the lump-sum amount that a series of future payments is worth right now.</td></tr>
<tr><td>fv</td><td>number</td><td>The future value, or a cash balance to be attained after the last payment is made. If omitted, it is assumed to be 0 (i.e., the future value of a loan, for example, is 0).</td></tr>
<tr><td>type</td><td>number</td><td>The timing of the payment, truncated to integer, as follows:
<table>
<tr><th>Value</th><th>Timing</th></tr>
<tr><td><code>0</code></td><td>Payment at the end of the period</td></tr>
<tr><td><code>1</code></td><td>Payment at the beginning of the period</td></tr>
</table></td></tr>
</table>

Arguments representing cash paid by investor shall be expressed as negative numbers; arguments representing cash received by the investor shall be expressed as positive numbers.

Return Type and Value: number – The interest payment for a given period for an investment based on periodic, constant payments and a constant interest rate.

However, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`IPMT(0.1/12,1*3,3,8000)` results in `-22.41`
`IPMT(0.1,3,3,8000)` results in `-292.45`

end example]

18.17.7.173 IRR

`IRR ( `*values*` [ , [ `*guess*` ]] )`

Description: Computes the internal rate of return for a series of cash flows represented by the numbers in *values*. (These cash flows do not have to be even, as they would be for an annuity. However, the cash flows shall occur at regular intervals, such as monthly or annually. The internal rate of return is the interest rate received for an investment consisting of payments (negative values) and income (positive values) that occur at regular periods.)

Arguments:

Name	Type	Description
values	array, reference, text, logical	The set of numbers for which the internal rate of return is to be calculated. *values* shall contain at least one positive value and one negative value to calculate the internal rate of return. The order of numbers in *values* is significant, so be sure payment and income numbers are in the desired sequence. If *values* contains elements that are text, logical values, or empty cells, those elements are ignored.
guess	number	An estimate of the result of `IRR`. If omitted, it is assumed to be 0.1 (i.e., 10 percent).

Return Type and Value: number – The internal rate of return for a series of cash flows.

However, if the implementation determines that a return value cannot be computed, `#NUM!` is returned.

[*Example*:

`IRR({-70000,12000,15000,18000,21000})` results in `-2.1245%`
`IRR({-70000,12000,15000,18000,21000,26000})` results in `8.6631%`
`IRR({-70000,12000,15000},-0.1)` results in `-44.3507%`

end example]

18.17.7.174 ISBLANK

Syntax:

`ISBLANK ( `*value*` )`

Description: Determines if *value* refers to an empty cell.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* refers to an empty cell; otherwise, FALSE.

[*Example*:

`ISBLANK(A10)` results in TRUE, when A10 is empty
`ISBLANK(A10)` results in FALSE, when A10 contains 123

end example]

18.17.7.175 ISERR

Syntax:

`ISERR ( ` *value* ` )`

Description: Determines if *value* is any of the error values other than #N/A.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* is one of the error values, excluding #N/A; otherwise, FALSE.

[*Example*:

`ISERR(A1)` results in TRUE if A1 evaluates to `#DIV/0!`, for example
`ISERR(B1)` results in FALSE if B1 evaluates to #N/A

end example]

18.17.7.176 ISERROR

Syntax:

`ISERROR ( ` *value* ` )`

Description: Determines if *value* is any of the error values.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* is one of the error values; otherwise, FALSE.

[*Example*:

`ISERROR(A1)` results in TRUE if A1 evaluates to `#DIV/0!`, for example

end example]

18.17.7.177 ISEVEN

Syntax:

`ISEVEN ( ` *value* ` )`

Description: Determines if *value* is an even number or refers to a cell containing an even number.

Arguments:

Name	Type	Description
value	number	The value to be tested. It is truncated to an integer.

Return Type and Value: logical – TRUE if *value* is an even number or refers to a cell containing an even number; otherwise, FALSE.

[*Example*:

`ISEVEN(12.456)` results in TRUE
`ISEVEN(A10)` results in FALSE, when A10 contains -15

end example]

18.17.7.178 ISLOGICAL

Syntax:

`ISLOGICAL ( ` *value* ` )`

Description: Determines if *value* contains a logical value or refers to a cell containing a logical value.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* contains a logical value or refers to a cell containing a logical value; otherwise, FALSE.

[*Example*:

ISLOGICAL(TRUE) results in TRUE
ISLOGICAL(A10) results in FALSE, when A10 contains 123
ISLOGICAL({TRUE,2}) results in TRUE
ISLOGICAL({2,TRUE}) results in FALSE

end example]

18.17.7.179 ISNA

Syntax:

ISNA (*value*)

Description: Determines if *value* is the error value #N/A.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* is #N/A; otherwise, FALSE.

[*Example*:

ISERR(A1) results in TRUE if A1 evaluates to #N/A
ISERR(B1) results in TRUE if B1 evaluates to #DIV/0!, for example

end example]

18.17.7.180 ISNONTEXT

Syntax:

`ISNONTEXT ( value )`

Description: Determines if *value* does not contain text or does not refer to a cell containing text. An empty cell is not text.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* does not contain text or does not refer to a cell containing text; otherwise, FALSE.

[*Example*:

`ISNONTEXT("ABC")` results in FALSE
`ISNONTEXT(A10)` results in TRUE, when A10 contains 123
`ISNONTEXT({1,"ABC"})` results in TRUE
`ISNONTEXT({"ABC",1})` results in FALSE

end example]

18.17.7.181 ISNUMBER

Syntax:

`ISNUMBER ( value )`

Description: Determines if *value* contains a number or refers to a cell that contains a number.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* contains a number or refers to a cell that contains a number; otherwise, FALSE.

[*Example*:

`ISNUMBER(10.56)` results in TRUE
`ISNUMBER(A10)` results in FALSE, when A10 contains ABC
`ISNUMBER({1,"ABC"})` results in TRUE

`ISNUMBER({"ABC",1})` results in `FALSE`

end example]

18.17.7.182 ISO.CEILING

Syntax:

`ISO.CEILING ( `*x*` , [ `*significance*` ] )`

Description: Computes a value that is *x* rounded-up, to the nearest multiple of *significance*. Regardless of the sign of *x*, a value is rounded up.

Arguments:

Name	Type	Description
x	number	The value to be rounded
significance	number	The optional multiple to which *x* is to be rounded. If *significance* is omitted, its default value is 1. [*Note*: The absolute value of the multiple is used, so the CEILING function will return the mathematical ceiling irrespective of the signs of *x* and *significance*. *end note*]

Return Type and Value: number – The rounded-up value of *x*.

However, if *x* and/or *significance* is zero, zero is returned.

[*Example*:

`ISO.CEILING(4.3)` rounds 4.3 up to nearest multiple of 1; that is, to 5
`ISO.CEILING(-4.3)` rounds -4.3 up to nearest multiple of 1; that is, to `-4`
`ISO.CEILING(4.3, 2)` rounds 4.3 up to the nearest multiple of 2; that is, to 6
`ISO.CEILING(4.3,-2)` rounds 4.3 up to the nearest multiple of -2; that is, to 6
`ISO.CEILING(-4.3,2)` rounds -4.3 up to the nearest multiple of 2; that is, to -4
`ISO.CEILING(-4.3,-2)` rounds -4.3 up to the nearest multiple of -2; that is, to -4

end example]

18.17.7.183 ISODD

Syntax:

`ISODD ( `*value*` )`

Description: Determines if *value* is an odd number or refers to a cell containing an odd number.

Arguments:

Name	Type	Description
value	number	The value to be tested. It is truncated to an integer.

Return Type and Value: logical – TRUE if *value* is an odd number or refers to a cell containing an odd number; otherwise, FALSE.

[*Example*:

`ISODD(12.456)` results in `FALSE`
`ISODD(A10)` results in TRUE, when A10 contains `-15`

end example]

18.17.7.184 ISPMT

Syntax:

`ISPMT ( `*rate*` , `*per*`, `*nper*` , `*pv*` )`

Description: Computes the interest paid during a specific period of an investment.

Arguments:

Name	Type	Description
rate	number	The interest rate for the investment.
per	number	The period for which the interest is to be found, and shall be in the range 1–*nper*.
nper	number	The total number of payment periods for the investment.
pv	number	The present value the investment.

Arguments representing cash paid by investor shall be expressed as negative numbers; arguments representing cash received by the investor shall be expressed as positive numbers.

Return Type and Value: number – The interest paid during a specific period of an investment.

[*Example*:

`ISPMT(0.1/12,1,3*12,8000000)` results in `-64814.81`
`ISPMT(0.1,1,3,8000000)` results in `-533333.33`

end example]

18.17.7.185 ISREF

Syntax:

`ISREF ( value )`

Description: Determines if *value* is a cell reference.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* is a cell reference; otherwise, FALSE.

[*Example*:

`ISREF("ABC")` results in FALSE
`ISREF(A10)` results in TRUE

end example]

18.17.7.186 ISTEXT

Syntax:

`ISTEXT ( value )`

Description: Determines if *value* contains text or refers to a cell containing text.

Arguments:

Name	Type	Description
value	any	The value to be tested. No conversion shall take place on an argument passed to this function.

Return Type and Value: logical – TRUE if *value* contains text or refers to a cell containing text; otherwise, FALSE.

[*Example*:

`ISTEXT("ABC")` results in TRUE
`ISTEXT(A10)` results in FALSE, when A10 contains 123
`ISTEXT({1,"ABC"})` results in FALSE
`ISTEXT({"ABC",1})` results in TRUE

end example]

18.17.7.187 JIS

Syntax:

`JIS ( `*string*` )`

Description: Creates a string that is the conversion of half-width (single-byte) letters within *string* to full-width (double-byte) characters.

Arguments:

Name	Type	Description
string	text	Designates the string to be converted. If *string* does not contain any half-width English letters or katakana, nothing in *string* is converted.

Return Type and Value: text – The string resulting from the conversion.

[*Example*:

`JIS("ABC")` results in ABC
`JIS("ｴｸｾﾙ")` results in エクセル

end example]

18.17.7.188 KURT

Syntax:

KURT (*argument-list*)

Description: Computes the kurtosis of a data set. Kurtosis characterizes the relative peakedness or flatness of a distribution compared with the normal distribution. Positive kurtosis indicates a relatively peaked distribution. Negative kurtosis indicates a relatively flat distribution.

Mathematical Formula:

Kurtosis is defined as:

$$\left\{\frac{n(n+1)}{(n-1)(n-2)(n-3)}\sum\left(\frac{x_i-\bar{x}}{s}\right)^4\right\}-\frac{3(n-1)^2}{(n-2)(n-3)}$$

where:

n is the number of elements in *argument-list*.

xj is the value of the j-th element in *argument-list*.
$\bar{x}$ is the mean of the values in *argument-list*.
s is the standard deviation of the values in *argument-list*.

Arguments:

Name	Type	Description
argument-list	array reference to an array, number, name, or reference to number.	The *arguments* in *argument-list* are the values for which kurtosis is to be calculated. Any *argument* in *argument-list* can be an array or a reference to an array. Logical values and text representations of numbers that are directly entered into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The kurtosis of a data set.

However, if

- There are fewer than four data points, the return value is unspecified.
- The standard deviation of the sample equals zero, the return value is unspecified.

[*Example*:

`KURT(10.5,12.4,19.4,23.2)` results in `-3.644621343`
`KURT(10.5,{12.4,19.4},23.2)` results in `-3.644621343`

end example]

18.17.7.189 LARGE

Syntax:

LARGE (*array* , *k*)

Description: Computes the k^{th} largest value in a data set.

Arguments:

Name	Type	Description
array	array, reference	The set of numbers from which the k^{th}-largest value is to be determined.
k	number	The position (from the largest) in the array or cell range of data to return.

Return Type and Value: number – The k^{th} largest value in a data set.

However, if

- *array* is empty, the return value is unspecified.
- $k \leq 0$, `#NUM!` is returned.
- *k* is greater than the number of data points, `#NUM!` is returned.

[*Example*:

`LARGE({3,5,3,5,4;4,2,4,6,7},3)` results in `5`
`LARGE({3,5,3,5,4;4,2,4,6,7},7)` results in `4`

end example]

18.17.7.190 LCM

Syntax:

`LCM ( argument-list )`

Description: Computes the least common multiple of the one or more *arguments* in *argument-list*.

Arguments:

Name	Type	Description
argument-list	number	*argument-list* specifies the *arguments*. Each argument is truncated to an integer.

Return Type and Value: number – The least common multiple of one or more numbers.

However, if any *argument* is negative, `#NUM!` is returned.

[*Example*:

`LCM(5)` results in `5`
`LCM(5,2)` results in `10`
`LCM(24.99,36.45)` results in `72`
`LCM(24,36,15)` results in `360`

end example]

18.17.7.191 LEFT

Syntax:

`LEFT ( string [ , number-chars ] )`

Description: Extracts the left-most *number-chars* characters from *string*. (LEFT is intended for use with languages that use the single-byte character set (SBCS), whereas LEFTB (§18.17.7.192) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-chars	number	The number of characters to be extracted. If omitted, a count of 1 shall be assumed. *number-chars* shall be at least 0. If *number-chars* exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing the left-most *number-chars* characters from *string*.

However, if *number-chars* is negative, #VALUE! is returned.

[*Example*:

`LEFT("abcdef",2)` results in ab
`LEFT(A10,4)` results in `xyz1`, when A10 contains `xyz123`

end example]

18.17.7.192 LEFTB

Syntax:

LEFTB (*string* [, *number-bytes*])

Description: Extracts the left-most *number-bytes*-worth of characters from *string*. (LEFTB is intended for use with languages that use the double-byte character set (DBCS), whereas LEFT (§18.17.7.192) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-bytes	number	The number of bytes to be extracted. If omitted, a count of 1 shall be assumed. *number-bytes* shall be at least 0. If *number-bytes* exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing the left-most *number-bytes*-worth of characters from *string*.

However, if *number-bytes* is negative, `#VALUE!` is returned.

[*Example*: Assuming 1-byte characters:

`LEFTB("abcdef",2)` results in ab
`LEFTB(A10,4)` results in `xyz1`, when A10 contains `xyz123`

end example]

18.17.7.193 LEN

Syntax:

`LEN ( string )`

Description: Determines the number of characters in *string*. (`LEN` is intended for use with languages that use the single-byte character set (SBCS), whereas `LENB` (§18.17.7.194) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string	text	Designates the string whose length is to be found.

Return Type and Value: number – The number of characters in *string*.

[*Example*:

`LEN("abc")` results in 3
`LEN(A10)` results in 3, when A1 contains `abc`

end example]

18.17.7.194 LENB

Syntax:

`LENB ( string )`

Description: Determines the number of bytes in *string*. (`LENB` is intended for use with languages that use the double-byte character set (DBCS), whereas `LEN` (§18.17.7.193) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string	text	Designates the string whose length is to be found.

Return Type and Value: number – The number of bytes in *string*.

[*Example*: Assuming 1-byte characters:

`LENB("abc")` results in 3
`LENB(A10)` results in 3, when A1 contains `abc`

end example]

18.17.7.195 LINEST

Syntax:

`LINEST (` *known-ys* `[ , [` *known-xs* `] [ , [` *const-flag* `] [ ,` *stats-flag* `] ] )`

Description: Calculates the statistics for a line by using the "least squares" method to calculate a straight line that best fits the data, and returns an array that describes the line.

Mathematical Formula:

The equation for the line is:

$$y = mx + b$$

or

…

$y = m_1x_1 \ + \ m_2x_2 + \ \dots + b$ (if there are multiple ranges of x-values)

where the dependent y-value is a function of the independent x-values. The m-values are coefficients corresponding to each x-value, and b is a constant value. y, x, and m can be vectors.

When there is only one independent x-variable, the slope and y-intercept values can be obtained directly by using the following formulas:

Slope: `INDEX(LINEST(`*known-ys*`,`*known-xs*`),1)`

Y-intercept: `INDEX(LINEST(`*known-ys*`,`*known-xs*`),2)`

The accuracy of the line calculated by `LINEST` depends on the degree of scatter in the data. The more linear the data, the more accurate the `LINEST` model. `LINEST` uses the method of least squares for determining the best fit for the data. When there is only one independent x-variable, the calculations for m and b are based on the following formulas:

$$m = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$$

$$b = \bar{y} - m\bar{x}$$

where:

- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*known-xs*)
- y = a sample value
- $\bar{y}$ = the sample mean AVERAGE(*known-ys*)

Arguments:

Name	Type	Description
known-ys	array	The set of y-values already known in the relationship y=mx+b. If the array *known-ys* is a single column, then each column of *known-xs* is interpreted as a separate variable. If the array *known-ys* is a single row, then each row of *known-xs* is interpreted as a separate variable.
known-xs	array	An optional set of x-values that might already be known in the relationship y=mx+b. The array *known-xs* can include one or more sets of variables. If only one variable is used, *known-ys* and *known-xs* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, *known-ys* shall be a vector (that is, a range with a height of one row or a width of one column). If *known-xs* is omitted, it is assumed to be the array {1,2,3,…} that is the same size as *known-ys*.
const-flag	logical	Specifies whether to force the constant *b* to be zero. If TRUE or omitted, *b* is calculated normally. If FALSE, *b* is set to zero, and the m-values are adjusted to fit y=mx.
stats-flag	logical	Specifies whether to return additional regression statistics. If TRUE, LINEST returns the additional regression statistics (see table below), so the returned array is {mn, mn-1, …, m1, b; sen, sen-1, …, se1, seb; r2, sey; F, df; ssreg, ssresid}. If FALSE or omitted, LINEST returns only the m-coefficients and the constant b.

The additional regression statistics are as follows:

Statistic	Description
se1, se2, …, se*n*	The standard error values for the coefficients m1, m2, …, m*n*.
seb	The standard error value for the constant b.
r2	The coefficient of determination.
sey	The standard error for the y estimate.
F	The F statistic, or the F-observed value.
df	The degrees of freedom.
ssreg	The regression sum of squares.
ssresid	The residual sum of squares.

Return Type and Value: array – The array that describes the line, in the form {mn, mn-1, …, m1, b}. The following illustration shows the order in which the additional regression statistics are returned.

[*Example*:

`LINEST({1,9,5,7},{0,4,2,3},,FALSE)` results in a slope of 2 and a y-intercept of 1

end example]

18.17.7.196 LN

Syntax:

`LN ( x )`

Description: Computes the natural logarithm of x.

Arguments:

Name	Type	Description
x	number	The positive real number for which the natural logarithm is being computed.

Return Type and Value: number – The natural logarithm of x.

However, if x is zero or negative, `#NUM!` is returned.

[*Example*:

`LN(86)` results in `4.454347296`
`LN(2.7182818)` results in `0.99999999`
`LN(EXP(3))` results in 3

end example]

18.17.7.197 LOG

Syntax:

`LOG ( x [ , base ] )`

Description: Computes the logarithm of x to the base *base*.

Arguments:

Name	Type	Description
x	number	The positive real number for which the logarithm is being computed.
base	number	The base of the logarithm. If omitted, base 10 is assumed.

Return Type and Value: number – The logarithm of x.

However, if

- x is zero or negative, `#NUM!` is returned.
- *base* is zero or negative, `#NUM!` is returned.

[*Example*:

`LOG(10)` results in `1`
`LOG(8,2)` results in `3`
`LOG(86,2.7182818)` results in `4.454347343`

end example]

18.17.7.198 LOG10

Syntax:

`LOG10 ( x )`

Description: Computes the base-10 logarithm of x.

Arguments:

Name	Type	Description
x	number	The positive real number for which the logarithm is being computed.

Return Type and Value: number – The base-10 logarithm of x.

However, if x is zero or negative, `#NUM!` is returned.

[*Example*:

`LOG10(86)` results in `1.934498451`
`LOG10(10)` results in `1`
`LOG10(1E5)` results in 5
`LOG10(10^5)` results in 5

end example]

18.17.7.199 LOGEST

Syntax:

`LOGEST (` *known-ys* `[ , [` *known-xs* `] [ , [` *const-flag* `][ ,` *stats-flag* `]] )`

Description: Calculates an exponential curve that fits the data, and returns an array of values that describes the curve.

Mathematical Formula:

The equation for the curve is:

$$y = b \times m^x$$

or

$$y = b \times {m_1}^{x_1} \times {m_2}^{x_2} \ \ldots \text{ (if there are multiple x-values)}$$

where the dependent y-value is a function of the independent x-values. The m-values are bases corresponding to each exponent x-value, and b is a constant value. [*Note*: y, x, and m can be vectors. *end note*]

When there is only one independent x-variable, the y-intercept (b) values can be obtained directly by using the following formula:

Y-intercept (b): `INDEX(LOGEST(`*known-ys*`,`*known-xs*`),2)`

Arguments:

Name	Type	Description
known-ys	array	The set of y-values already known in the relationship

Name	Type	Description
		$y=b*m^x$. If the array *known-ys* is a single column, then each column of *known-xs* is interpreted as a separate variable. If the array *known-ys* is a single row, then each row of *known-xs* is interpreted as a separate variable.
known-xs	array	An optional set of x-values that might already be known in the relationship $y=b*m^x$. The array *known-xs* can include one or more sets of variables. If only one variable is used, *known-ys* and *known-xs* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, *known-ys* shall be a vector (that is, a range with a height of one row or a width of one column). If *known-xs* is omitted, it is assumed to be the array {1,2,3,…} that is the same size as *known-ys*.
const-flag	logical	Specifies whether to force the constant *b* to be 1. If TRUE or omitted, *b* is calculated normally. If FALSE, *b* is set to 1, and the m-values are adjusted to fit $y=m^x$.
stats-flag	logical	Specifies whether to return additional regression statistics. If TRUE, LOGEST returns the additional regression statistics, so the returned array is {mn, mn-1, …, m1, b; sen, sen-1, …, se1, seb; r2, sey; F, df; ssreg, ssresid}. If FALSE or omitted, LOGEST returns only the m-coefficients and the constant b.

The additional regression statistics are described in §18.17.7.195.

Return Type and Value: array – The array that describes the line, in the form {mn, mn-1, …, m1, b}. The order in which the additional regression statistics are returned is described in §18.17.7.195.

[*Example*: Given the following data:

	A	B
1	Month	Units
2	11	33,100
3	12	47,300
4	13	69,000
5	14	102,000
6	15	150,000
7	16	220,000
8	Formula	
9	1.463275628	495.3047702

When `LOGEST(B2:B7,A2:A7,TRUE,FALSE)` is an array formula spanning cells A9:B9, those cells take on the results shown.

end example]

18.17.7.200 LOGINV

Syntax:

LOGINV (*probability* , *mean* , *standard-dev*)

Description: Calculates the inverse of the lognormal cumulative distribution function of x, where ln(x) is normally distributed with parameters *mean* and *standard-dev*.

Mathematical Formula:

$$LOGINV(p, \mu, \sigma) = e^{[\mu + \sigma \times (NORMSINV(p))]}$$

where:

- p = argument *probability*
- μ = argument *mean*
- σ = argument *standard-dev*

Arguments:

Name	Type	Description
probability	number	A probability associated with the lognormal distribution.
mean	number	The mean of ln(x).
standard-dev	number	The standard deviation of ln(x).

Return Type and Value: number – The inverse of the lognormal cumulative distribution function of x.

However, if

- *probability* < 0 or *probability* > 1, `#NUM!` is returned.
- *standard-dev* ≤ 0, `#NUM!` is returned.

[*Example*:

`LOGINV(0.039084,3.5,1.2)` results in `4.000025219`

end example]

18.17.7.201 LOGNORMDIST

Syntax:

LOGNORMDIST (*x* , *mean* , *standard-dev*)

Description: Calculates the cumulative lognormal distribution of x, where ln(x) is normally distributed with parameters *mean* and *standard-dev*.

Mathematical Formula:

$$LOGNORMDIST(x,\mu,\sigma) = NORMSDIST\left(\frac{\ln(x)-\mu}{\sigma}\right)$$

where:

- x = argument x
- μ = argument *mean*
- σ = argument *standard-dev*

Arguments:

Name	Type	Description
x	number	The value at which to evaluate the function.
mean	number	The mean of ln(x).
standard-dev	number	The standard deviation of ln(x).

Return Type and Value: number – The inverse of the lognormal cumulative distribution function of x.

However, if

- $x \le 0$, #NUM! is returned.
- *standard-dev* ≤ 0, #NUM! is returned.

[*Example*:

LOGNORMDIST(4,3.5,1.2) results in 0.039083556

end example]

18.17.7.202 LOOKUP

Syntax:

vector form: LOOKUP (*lookup-value* , *lookup-vector* , *result-vector*)

array form: LOOKUP (*lookup-value* , *array*)

Description: The vector form looks in a vector for a value, and returns a value from the same position in a second vector. The array form looks in the first row or column of an array for the specified value and returns a value from the same position in the last row or column of that array.

Arguments:

Name	Type	Description
lookup-value	number, string, logical, name, reference	The value to search for in *lookup-vector* (or *array*).
lookup-vector	reference	A range that contains only one row or one column. The values in *lookup-vector* can be strings, numbers, or logical values. These values shall be placed in "ascending" order, as follows: …, -2, -1, 0, 1, 2, …, A–Z, FALSE, TRUE. Upper- and lowercase strings are equivalent. If LOOKUP can't find the *lookup-value*, it matches the largest value in *lookup-vector* (or *array*) that is less than or equal to *lookup-value*.
result-vector	reference	A range that contains only one row or column. It shall be the same size as *lookup-vector*.
array	text, number, logical	A range of cells whose values are to be compared with *lookup-value*. These values shall be placed in "ascending" order, as follows: …, -2, -1, 0, 1, 2, …, A–Z, FALSE, TRUE. Upper- and lowercase strings are equivalent. If *array* covers an area that has more columns than rows, *lookup-value* is searched for in the first row. If *array* is square or has more rows than columns, *lookup-value* is searched for in the first column.

Return Type and Value: any – The vector form looks in a vector for a value, and returns a value from the same position in a second vector. The array form looks in the first row or column of an array for the specified value and returns a value from the same position in the last row or column of that array.

However, if

- *lookup-value* is smaller than the smallest value in *lookup-vector* (or the first row or column of *array*), the return value is unspecified.
- The size of the range specified by *result-vector* is not the same as that specified by *lookup-vector*, the return value is unspecified.
- The values in *lookup-vector* (or *array*) are not in "ascending" order, the return value is unspecified.

[*Example*: Given the following data:

	A	B
1	Frequency	Color
2	4.14	red
3	4.19	orange
4	5.17	yellow
5	5.77	green
6	6.39	blue

`LOOKUP(4.19,A2:A6,B2:B6)` results in `orange`
`LOOKUP(5,A2:A6,B2:B6)` results in `orange`
`LOOKUP(7.66,A2:A6,B2:B6)` results in `blue`

`LOOKUP("C",{"a","b","c","d";1,2,3,4})` results in 3
`LOOKUP("bump",{"a",1;"b",2;"c",3})` results in 2

end example]

18.17.7.203 LOWER

Syntax:

`LOWER ( `*string*` )`

Description: Makes a lowercase version of *string* by doing a character-by-character conversion of *string* to lowercase, except as noted below. [*Note*: The conversion of characters in *string* is not dependent on position/context of the character within the string, except as noted below. *end note*]

Arguments:

Name	Type	Description
string	text	Designates the string to be converted.

Return Type and Value: text – The lowercase version of *string* after doing a character-by-character conversion to lowercase. However, there is one exception; when Σ (U+03A3) is found in a word-final position, it is converted to ς (U+03C2) instead of σ (U+03C3).

[*Example*:

`LOWER("AbCd123#$%^")` results in `abcd123#$%^`
`LOWER(A10)` results in `234frtqwc$#%`, when A10 contains `234FRTqwc$#%`

end example]

18.17.7.204 MATCH

Syntax:

MATCH (*lookup-value* , *lookup-array* [, [*match-type*]])

Description: Locates the relative position of an array item that matches a specified value in a specified order. MATCH shall not distinguish between uppercase and lowercase letters when matching strings.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>lookup-value</td><td>number, string, logical, name, reference</td><td>The value to search for in lookup-array. If match-type is 0 and lookup-value is a string, the wildcard characters, question mark (?) and asterisk (*), can be used in lookup-value. A question mark matches any single character; an asterisk matches any sequence of characters. To locate a question mark or asterisk, precede that character with a tilde (~).</td></tr>
<tr><td>lookup-array</td><td>array, reference</td><td>A contiguous range of cells containing possible lookup values.</td></tr>
<tr><td>match-type</td><td>number</td><td>Specifies how lookup-value is matched with values in lookup-array, as follows:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>-1</td><td>Finds the smallest value that is greater than or equal to lookup-value. The values in lookup-array shall be placed in "descending" order: TRUE, FALSE, Z–A, …, 2, 1, 0, -1, -2, …</td></tr>
<tr><td>0</td><td>Finds the first value that is exactly equal to lookup-value. The values in lookup-array can be in any order.</td></tr>
<tr><td>1 or omitted</td><td>Finds the largest value that is less than or equal to lookup-value. The values in lookup-array shall be placed in "ascending" order: …, -2, -1, 0, 1, 2, …, A–Z, FALSE, TRUE.</td></tr>
</table></td></tr>
</table>

Return Type and Value: number – The relative position of an array item that matches a specified value in a specified order.

However, if

- No match is found, #N/A is returned.
- *match-type*'s value is out-of-bounds, `#NUM!` is returned.

[*Example*:

`MATCH(39,{25,38,40,41},1)` results in 2
`MATCH(41,{25,38,40,41},0)` results in 4

end example]

18.17.7.205 MAX

Syntax:

`MAX ( ` *argument-list* ` )`

Description: Computes the largest of a set of numbers.

Arguments:

Name	Type	Description
argument-list	logical, number, name, arrays, reference to number. Any *argument* can be an array or a reference to an array.	The *arguments* in *argument-list* designate the values for which the largest value is to be computed. Logical values and text representations of numbers occurring directly in the list of arguments are included. However, logical values and numbers in strings and are ignored inside references. [*Note*: To include these, use MAXA (§18.17.7.206). *end note*] If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The largest of a set of numbers; however, if the arguments contain no numbers, zero is returned.

[*Example*:

`MAX(10.4,-3.5,12.6)` results in `12.6`
`MAX(10.4,{-3.5,12.6})` results in `12.6`
`MAX({"ABC",TRUE})` results in `0`

Consider the case in which cell B3 contains `0`:
`MAX(-10,-12,-15,B3)` results in `-10`
`MAXA(-10,-12,-15,B3)` results in `0`

end example]

18.17.7.206 MAXA

Syntax:

`MAXA ( `*argument-list*` )`

Description: Computes the largest of a set of numbers.

Arguments:

Name	Type	Description
argument-list	logical, number, name, arrays, reference to number. Any *argument* can be an array or a reference to an array.	The *arguments* in *argument-list* designate the values for which the largest value is to be computed. Logical values and text representations of numbers occurring directly in the list of arguments are included. Logical values and numbers in strings inside references are also included. [*Note*: To ignore these, use MAX (§18.17.7.205). *end note*] If an array or reference argument contains non-numeric text or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The largest of a set of numbers; however, if the arguments contain no numbers, zero is returned.

[*Example*:

`MAXA(10.4,-3.5,12.6)` results in `12.6`
`MAXA(10.4,{-3.5,12.6})` results in `12.6`
`MAXA({"ABC",TRUE})` results in `0`

Consider the case in which cell B3 contains `0`:
`MAX(-10,-12,-15,B3)` results in `-10`
`MAXA(-10,-12,-15,B3)` results in `0`

end example]

18.17.7.207 MDETERM

Syntax:

`MDETERM ( `*array*` )`

Description: Computes the determinant of the square matrix of numbers designated by *array*. The determinant is calculated with an accuracy of at least 15 digits, which can lead to a small numeric error when the calculation is not complete. [*Example*: The determinant of a singular matrix can differ from zero by 1E-16. *end example*]

Arguments:

Name	Type	Description
array	array, reference	Designate a square matrix of numbers.

Return Type and Value: number – The determinant of *array*. Some square matrices cannot be inverted. The determinant of a non-invertible matrix is 0.

However, if

- Any cells in *array* are empty or contain text, the return value is unspecified.
- The matrix designated by *array* is not square, #VALUE! is returned.

[*Example*:

`MDETERM(A2:D5)` results in the determinant of the 4x4 array designated by the cell range
`MDETERM({3,6,1;1,1,0;3,10,2})` results in `1`
`MDETERM({3,6;1,1})` results in `-3`

end example]

18.17.7.208 MDURATION

Syntax:

`MDURATION (` *settlement* `,` *maturity* `,` *coupon* `,` *yld* `,` *frequency* `[ , [` *basis* `] ] )`

Description: Computes the modified Macaulay duration for a security with an assumed par value of 100.

Mathematical Formula:

$$MDURATION = \frac{DURATION}{1 + \left(\frac{Market\ yield}{Coupon\ payments\ per\ yea}\right)}$$

where:

- *Coupon payments per year* = argument *frequency*
- *DURATION* = `DURATION(settlement, maturity, coupon, yld, frequency, [basis])`
- *Market yield* = argument *yld*

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>coupon</td><td>number</td><td>The security's annual coupon rate.</td></tr>
<tr><td>yld</td><td>number</td><td>The security's annual yield.</td></tr>
<tr><td>frequency</td><td>number</td><td>the number of coupon payments per year. (For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4.) frequency is truncated to an integer.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:

If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.
</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days</td></tr>
</table></td></tr>
</table>

Name	Type	Description
		<table><tr><td></td><td>between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr><tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr><tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr><tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments: - If the date is 28 or 29 February, it is adjusted to 30 February. - For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 Februar y. </td></tr></table>

Time information in the date arguments is ignored.

Return Type and Value: number – The modified Macaulay duration for a security with an assumed par value of 100.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *coupon* or *yld* < 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`MDURATION(DATE(2008,1,1),DATE(2016,1,1),0.08,0.09,2,1)` results in `5.7357`

end example]

18.17.7.209 MEDIAN

Syntax:

`MEDIAN (` *argument-list* `)`

Description: Computes the median of the numeric values of its arguments. The median of a set of values is the value for which half the numbers in the set are greater, and half the values are less. For sets with an odd number of values, the median is calculated by finding the value whose rank in the ordered set of all values is equal to half the number of items (n) in the set plus one half (i.e., n/2 + 1/2). If the number of values in the set is even, then the median is defined to be the average of the values of rank n/2 and n/2 + 1.

Arguments:

Name	Type	Description
argument-list	logical, number, name, arrays, reference to number.	The *arguments* in *argument-list* designate the values whose median is to be computed. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The median of the values of its arguments.

[*Example*:

`MEDIAN(10,20)` results in `15`
`MEDIAN(-3.5,1.4,6.9,-4.5)` results in `-1.05`
`MEDIAN({-3.5,1.4,6.9},-4.5)` results in `-1.05`

end example]

18.17.7.210 MID

Syntax:

`MID ( `*string*` , `*start-pos*` , `*number-chars*` )`

Description: Extracts *number-chars* characters from *string*, starting at character position *start-pos*. (MID is intended for use with languages that use the single-byte character set (SBCS), whereas MIDB (§18.17.7.211) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-chars	number	The number of characters to be extracted. *number-chars* shall be at least 0.
start-pos	number	The starting position within *string*, where the first character is position 1. If *start-pos* is greater than the length of *string*, or if *start-pos* and *number-chars* combined exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing *number-chars* characters from *string*, starting at character position *start-pos*.

However, if

- *start-pos* < 0, #VALUE! is returned.
- *number-chars* < 0, #VALUE! is returned.

[*Example*:

`MID("abcdef",3,2)` results in cd
`MID(A10,4,1)` results in 1, when A10 contains `xyz123`
`MID("abcdef",4,5)` results in def

end example]

18.17.7.211 MIDB

Syntax:

`MIDB ( string , start-pos , number-bytes )`

Description: Extracts *number-bytes*-worth of characters from *string*, starting at character position *start-pos*. (`MIDB` is intended for use with languages that use the double-byte character set (DBCS), whereas `MID` (§18.17.7.210) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-bytes	number	The number of characters to be extracted. *number- bytes* shall be at least 0.
start-pos	number	The starting position within *string*, where the first byte is position 1. If *start-pos* is greater than the length of *string*, or if *start-pos* and *number- bytes* combined exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing *number-bytes*-worth of characters from *string*, starting at character position *start-pos*.

However, if

- *start-pos* < 0, #VALUE! is returned.
- *number-bytes* < 0, #VALUE! is returned.

[*Example*: Assuming 1-byte characters:

`MIDB("abcdef",3,2)` results in `cd`
`MIDB(A10,4,1)` results in `1`, when A10 contains `xyz123`
`MIDB("abcdef",4,5)` results in `def`

end example]

18.17.7.212 MIN

Syntax:

`MIN ( argument-list )`

Description: Computes the smallest of a set of numbers.

Arguments:

Name	Type	Description
argument-list	logical, number, name, arrays, reference to number. Any *argument* can be an array or a reference to an array.	The *argument*s in *argument-list* designate the values for which the largest value is to be computed. Logical values and text representations of numbers occurring directly in the list of arguments are included. However, logical values and numbers in strings and are ignored inside references. [*Note*: To include these, use MINA (§18.17.7.213). *end note*] If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The smallest of a set of numbers; however, if the *argument*s contain no numbers, zero is returned.

[*Example*:

`MIN(10.4,-3.5,12.6)` results in `-3.5`
`MIN(10.4,{-3.5,12.6})` results in `-3.5`
`MIN({"ABC",TRUE})` results in `0`

Consider the case in which cell B3 contains `0`:
`MIN(10,12,15,B3)` results in `10`
`MINA(10,12,15,B3)` results in `0`

end example]

18.17.7.213 MINA

Syntax:

`MINA (` *argument-list* `)`

Description: Computes the smallest of a set of numbers.

Arguments:

Name	Type	Description
argument-list	logical, number, name, arrays, reference to number. Any *argument* can be an array or a reference to an array.	The *argument*s in *argument-list* designate the values for which the largest value is to be computed. Logical values and text representations of numbers occurring directly in the list of arguments are included. Logical values and numbers in strings inside references are also included. [*Note*: To ignore these, use MIN (§18.17.7.212). *end note*] If an array or reference argument contains non-numeric text or empty cells, those values are ignored; however,

Name	Type	Description
		cells with the value 0 are included.

Any *argument* in *argument-list* can be an array or a reference to an array.

Return Type and Value: number – The smallest of a set of numbers; however, if the *arguments* contain no numbers, zero is returned.

[*Example*:

`MINA(10.4,-3.5,12.6)` results in `-3.5`
`MINA(10.4,{-3.5,12.6})` results in `-3.5`
`MINA({"ABC",TRUE})` results in `0`

Consider the case in which cell B3 contains `0`:
`MIN(10,12,15,B3)` results in `10`
`MINA(10,12,15,B3)` results in `0`

end example]

18.17.7.214 MINUTE

Syntax:

`MINUTE (` *time-value* `)`

Description: Computes the minute for the date and/or time having the given *time-value*.

Arguments:

Name	Type	Description
time-value	number	The date and/or time whose minute is to be computed. That date and/or time shall be expressed either as a serial date-time, in which case, its integer part is ignored, or as a *string-constant* having any date and/or time format, in which case, any date information shall be ignored.

Return Type and Value: number – The minute for the date and/or time having the given *time-value*. The returned value shall be in the range 0–59.

However, if *time-value* is out of range for the current date system, `#NUM!` is returned.

[*Example*:

`MINUTE(DATE(2006,2,26)+TIME(2,10,20))` results in `10`
`MINUTE(TIME(22,56,34))` results in 56
`MINUTE(0)` results in 0, since serial date-time 0 represents `00:00:00`
`MINUTE(10.5)` results in 0, since serial date-time .5 represents `12:00:00`
`MINUTE("22-Oct-2001 10:53:12")` results in 53
`MINUTE("10:53:12 pm")` results in 53
`MINUTE("22:53:12")` results in 53

end example]

18.17.7.215 MINVERSE

Syntax:

`MINVERSE ( ` *array* ` )`

Description: Computes the inverse of the square matrix of numbers designated by *array*. The inverse matrix is calculated with an accuracy of at least 15 digits, which can lead to a small numeric error when the cancellation is not complete.

Arguments:

Name	Type	Description
array	array, reference	Designate a square matrix of numbers.

Return Type and Value: number – The inverse of the square matrix designated by *array*.

However, if

- Any cells in *array* are empty or contain text, the return value is unspecified.
- The matrix designated by *array* is not square, #VALUE! is returned.
- The matrix cannot be inverted, the return value is unspecified.

[*Example*:

`MINVERSE({3,6,1;1,1,0;3,10,2})` results in 2
`MINVERSE({3,6;1,1})` results in `-0.333333333`

end example]

18.17.7.216 MIRR

`MIRR ( ` *values* ` , ` *finance-rate* ` , ` *reinvest-rate* ` )`

Description: Computes the modified internal rate of return for a series of periodic cash flows. (Both the cost of the investment and the interest received on reinvestment of cash are considered.)

Mathematical Formula:

The formula for MIRR is:

$$\left(\frac{-NPV(rrate, values[positive]) * (1 + rrate)^n}{NPV(frate, values[negative]) * (1 + frate)}\right)^{\frac{1}{n-1}} - 1$$

where:

- $frate$ = argument *finance-rate*
- n = number of cash flows in argument *values*
- $rrate$ = argument *reinvest-rate*
- $values$ = argument *values*

Arguments:

Name	Type	Description
values	array, reference	Designates a set of numbers for which the rate of return is to be calculated. *values* shall contain at least one positive value and one negative value to calculate the internal rate of return. The order of numbers in *values* is significant, so be sure payment and income numbers are in the desired sequence. If *values* contains elements that are text, logical values, or empty cells, those elements are ignored.
finance-rate	number	The interest rate paid pay on the money used in the cash flows.
reinvest-rate	number	The interest rate received on the cash flows as they are reinvested.

Return Type and Value: number – The modified internal rate of return for a series of periodic cash flows.

However, if *values* does not contain at least one positive value and one negative value, `#DIV/0!` is returned.

[*Example*:

`MIRR({-120000,39000,30000,21000,37000,46000},0.1,0.12)` results in `12.6094%`
`MIRR({-120000,39000,30000,21000},0.1,0.12)` results in `-4.8045%`
`MIRR({-120000,39000,30000,21000,37000,46000},0.1,0.14)` results in `13.4759%`

end example]

18.17.7.217 MMULT

Syntax:

`MMULT ( array-1 , array-2 )`

Description: Computes the product of the matrices of numbers designated by *array-1* and *array-2*.

Mathematical Formula:

The matrix product array a of two arrays b and c is:

$$a_{ij} = \sum_{k=1}^{n} b_{ik}\, c_{kj}$$

Where:

- b_{ik} = the element in the i-th row and k-th column in argument *array-1*
- c_{kj} = the element in the k-th row and j-th column in argument *array-2*
- i = the row number
- j = the column number

Arguments:

Name	Type	Description
array-1	array, reference, name	Designate the matrices of numbers to be multiplied.
array-2		

Return Type and Value: number – The product of the matrices of numbers designated by *array-1* and *array-2*.

However, if

- Any cells in *array-1* or *array-2* are empty or contain text, the return value is unspecified.
- The number of columns in *array-1* is different from the number of rows in *array-2*, #VALUE! is returned.

[*Example*:

`MMULT({3,6,1;1,1,0},{5,7;4,6;2,5})` results in 41

end example]

18.17.7.218 MOD

Syntax:

`MOD ( x , y )`

Description: Computes the remainder when x is divided by y. The result has the same sign as y.

Arguments:

Name	Type	Description
x	number	The number for which the remainder is being sought.
y	number	The number by which x is to be divided.

Return Type and Value: number – The remainder when x is divided by y. The result has the same sign as y. If y is 0, the return value is unspecified.

[*Example*:

`MOD(3,2)` results in `1`
`MOD(-3,2)` results in `1`
`MOD(3,-2)` results in `-1`
`MOD(-3,-2)` results in `-1`

end example]

18.17.7.219 MODE

Syntax:

`MODE (` *argument-list* `)`

Description: Computes the most frequently occurring of the numeric values of its arguments. If the set of values contains more than one most-frequent value, the first occurrence of any most-frequent value in the list is used as the result.

Arguments:

Name	Type	Description
argument-list	logical, number, name, arrays, reference to number. Any *argument* can be an array or a reference to an array.	The *arguments* in *argument-list* designate the values whose mode is to be computed. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The most frequently occurring of the values of its arguments.

However, if the data set contains no duplicate data points, #N/A is returned.

[*Example*:

`MODE(9,1,5,1,9,5,6,6)` results in `9`
`MODE(1,9,5,1,9,5,6,6)` results in `1`
`MODE(5,1,9,5,1,9,6,6)` results in `5`

end example]

18.17.7.220 MONTH

Syntax:

`MONTH ( `*date-value*` )`

Description: Computes the numeric month in the Gregorian calendar [ISO 8601 §3.2.1] for the date and/or time having the given *date-value*, taking into account the current date system. That date and/or time shall be expressed either as a serial date-time, in which case, its fractional part is ignored, or as a *string-constant* having any date and/or time format, in which case, any time information shall be ignored.

Arguments:

Name	Type	Description
date-value	number, text	The date and/or time whose month is to be computed. That date and/or time shall be expressed either as a serial date-time, in which case, its fractional part is ignored, or as a *string-constant* having any date and/or time format, in which case, any time information shall be ignored.

Return Type and Value: number – The month in the Gregorian calendar [ISO 8601 §3.2.1] for the date and/or time having the given *date-value*, in the range 1–12.

However, if *date-value* is out of range for the current date system, `#NUM!` is returned.

[*Example*:

`MONTH(DATE(2006,1,2))` results in 1
`MONTH(DATE(2006,0,2))` results in 12
`MONTH("2006/1/2 10:45 AM")` results in 1
`MONTH(30000)` results in 2 for both the 1900 and 1904 date systems

end example]

18.17.7.221 MROUND

Syntax:

`MROUND ( `*x*` , `*multiple*`)`

Description: Computes x rounded to $multiple$, away from zero. It rounds up if the remainder of dividing x by $multiple$ is greater than or equal to half the value of $multiple$; otherwise, it rounds down.

Arguments:

Name	Type	Description
x	number	The value to round.
$multiple$	number	The multiple to which x is to be rounded.

Return Type and Value: number – x rounded to $multiple$.

However, if x and $multiple$ have different signs, `#NUM!` is returned.

[*Example*:

`MROUND(10,3)` rounds 10 to a nearest multiple of 3; that is, to 9
`MROUND(-10,-3)` rounds -10 to a nearest multiple of -3; that is, to `-9`
`MROUND(1.3,0.2)` rounds 1.3 to a nearest multiple of 0.2; that is, to `1.4`

end example]

18.17.7.222 MULTINOMIAL

Syntax:

`MULTINOMIAL (` *argument-list* `)`

Description: Computes the ratio of the factorial of the sum of the values in $argument\text{-}list$ to the product of the factorials.

Mathematical Formula:

The multinomial is:

$$MULTINOMIAL(a, b, c) = \frac{(a + b + c)!}{a!\,b!\,c!}$$

where:

- a, b, c, … = the elements in $argument\text{-}list$

Arguments:

Name	Type	Description
$argument\text{-}list$	number	The $arguments$ in $argument\text{-}list$ designate the numerical values for which the multinomial is desired.

Return Type and Value: number – The ratio of the factorial of the sum of the values in *argument-list* to the product of the factorials.

However, if any *argument* is less than zero, `#NUM!` is returned.

[*Example*:

`MULTINOMIAL(2)` results in `1`
`MULTINOMIAL(2,3)` results in `10`
`MULTINOMIAL(2,3,4)` results in `1260`

end example]

18.17.7.223 N

Syntax:

`N ( value )`

Description: Converts *value* to a number or, if *value* is a reference to a single cell, converts the value of that cell to a number.

Arguments:

Name	Type	Description
value	any	Value to be converted.

Return Type and Value: number or error – An integer that is the converted value of *value*, or, if *value* is a reference to a single cell, the converted value of that cell, as follows:

value	Value Returned
number	That number
TRUE	1
FALSE	0
error value	That error value
Anything else (including array and text)	0

[*Example*:

`N(10.5)` results in `10.5`

`N(A10)` results in `-1234`, when A10 contains the number `-1234`
`N("ABC")` results in `0`
`N(A10)` results in `0`, when A10 contains the string ABC
`N(TRUE)` results in `1`
`N(A10)` results in `0`, when A10 contains `FALSE`
`N(A10)` results in #N/A, when A10 contains #N/A
`N({12.5,13.6,56.9}` results in `12.50`
`N(A10:A11)` results in `0`, when A10 contains `FALSE`, and A11 contains 321

end example]

18.17.7.224 NA

Syntax:

`NA ( )`

Description: Gets the error value #N/A. (The error value #N/A can be used instead of a call to this function; the result is the same.)

Arguments: None.

Return Type and Value: error – The error value #N/A.

[*Example*:

`NA()` results in #N/A
`IF(ISNA(NA()),"T","F")` results in T

end example]

18.17.7.225 NEGBINOMDIST

Syntax:

`NEGBINOMDIST (` *number-failures* `,` *number-successes* `,` *success-probability* `)`

Description: Computes the negative binomial distribution. `NEGBINOMDIST` returns the probability that there are *number-failures* failures before the *number-successes*th success, when the constant probability of a success is *success-probability*.

Mathematical Formula:

$$nb(x,r,p) = \binom{x+r-1}{r-1} p^r (1-p)^x$$

where:

- p = the argument *success-probability*.

- *r* = the argument *number-successes*
- *x* = the argument *number-failures*

Arguments:

Name	Type	Description
number-failures	number	The number of failures, truncated to integer.
number-successes	number	The threshold number of successes, truncated to integer.
success-probability	number	The probability of a success.

Return Type and Value: number – The negative binomial distribution.

However, if

- *number-failures* < 0 or *number-successes* < 1, `#NUM!` is returned.
- *success-probability* < 0 or *success-probability* > 1, `#NUM!` is returned.

[*Example*:

`NEGBINOMDIST(6,10,0.5)` results in `0.076370239`

end example]

18.17.7.226 NETWORKDAYS

Syntax:

`NETWORKDAYS ( `*start-date*` , `*end-date*` [ , `*holidays*` ] )`

Description: Computes the number of whole working days between *start-date* and *end-date*. Weekend days (Saturday, Sunday) and any holidays specified by *holidays* are not considered as working days.

Arguments:

Name	Type	Description
start-date	number	The dates for which the difference is to be computed. *start-date* can be earlier than, the same as, or later than *end-date*.
end-date	number	
holidays	reference, array	An optional set of one or more dates that are to be excluded from the working day calendar. *holidays* shall be a range of cells that contain the dates, or an array constant of the serial date-times that represent those

Name	Type	Description
		dates. The ordering of dates or serial date-times in *holidays* can be arbitrary.

Return Type and Value: number – The number of whole working days between *start-date* and *end-date*, excluding the specified holidays. If *start-date* is later than *end-date*, the return value shall be negative, and the magnitude shall be the number of whole working days.

However, if

- *start-date* is out of range for the current date system, `#NUM!` is returned.
- *end-date* is out of range for the current date system, `#NUM!` is returned.

[*Example*:

`NETWORKDAYS(DATE(2006,1,1),DATE(2006,1,31))` results in 23
`NETWORKDAYS(DATE(2006,1,31),DATE(2006,1,1))` results in -23
`NETWORKDAYS(DATE(2006,1,1),DATE(2006,2,1),{"2006/1/2","2006/1/16"})` results in 21

end example]

18.17.7.227 NETWORKDAYS.INTL

Syntax:

Number form: `NETWORKDAYS.INTL (` *start-date* `,` *end-date* [, [*weekend-number*][`,` *holidays*]] `)`

String form: `NETWORKDAYS.INTL (` *start-date* `,` *end-date* [, [*weekend-string*][`,` *holidays*]] `)`

Description: Computes the number of whole working days between *start-date* and *end-date*. Weekend days and any holidays specified by *holidays* are not considered as working days.

Arguments:

Name	Type	Description
start-date	Number	The dates for which the difference is to be computed. *start-date* can be earlier than, the same as, or later than *end-date*.
end-date	Number	
weekend-number	Number	Indicates the days of the week that are weekend days and are not included in the number of whole working days between *start-date* and *end-date*. Values are shown in the table below.
weekend-string	String	Indicates the days of the week that are weekend days and are not included in the number of whole working days between *start-date* and *end-date*.

Name	Type	Description
		Values of *weekend-string* are seven characters long and each character in the string represents a day of the week, beginning with Monday. [*Example*: "0000011" would result in a weekend that is Saturday and Sunday. *end example*]
holidays	reference, array	An optional set of one or more dates that are to be excluded from the working day calendar. *holidays* shall be a range of cells that contain the dates, or an array constant of the serial date-times that represent those dates. The ordering of dates or serial date-times in *holidays* can be arbitrary.

weekend-number	Weekend days
1 or omitted	Saturday, Sunday
2	Sunday, Monday
3	Monday, Tuesday
4	Tuesday, Wednesday
5	Wednesday, Thursday
6	Thursday, Friday
7	Friday, Saturday
11	Sunday only
12	Monday only
13	Tuesday only
14	Wednesday only
15	Thursday only
16	Friday only
17	Saturday only

Return Type and Value: number – The number of whole working days between *start-date* and *end-date*, excluding the specified weekend days and holidays. If *start-date* is later than *end-date*, the return value shall be negative, and the magnitude shall be the number of whole working days.

However, if

- *start-date* is out of range for the current date system, `#NUM!` is returned.
- *end-date* is out of range for the current date system, `#NUM!` is returned.

[*Example*:

`NETWORKDAYS.INTL(DATE(2006,1,1),DATE(2006,1,31))` results in 23
`NETWORKDAYS.INTL(DATE(2006,1,31),DATE(2006,1,1))` results in -23
`NETWORKDAYS.INTL(DATE(2006,1,1),DATE(2006,2,1),7,{"2006/1/2","2006/1/16"})` results in 21
`NETWORKDAYS.INTL(DATE(2006,1,1),DATE(2006,2,1),"0000110",{"2006/1/2","2006/1/16"})` results in 21

end example]

18.17.7.228 NOMINAL

`NOMINAL ( `*effect-rate* `,` *npery* `)`

Description: Computes the nominal annual interest rate, given the effective rate and the number of compounding periods per year.

Mathematical Formula:

NOMINAL is related to EFFECT:

$$EFFECT = \left(1 + \frac{Nominal\_rate}{Npery}\right)^{Npery} - 1$$

where:

- $Nominal\_rate$ = argument *effect-rate*
- $Npery$ = argument *npery*

Arguments:

Name	Type	Description
effect-rate	number	The effective interest rate.
npery	number	The number of compounding periods per year, truncated to integer.

Return Type and Value: number – The nominal annual interest rate.

However, if

- *effect-rate* ≤ 0, `#NUM!` is returned.
- *npery* < 1, `#NUM!` is returned.

[*Example*:

`NOMINAL(0.053543,4)` results in `5.2500%`

end example]

18.17.7.229 NORMDIST

Syntax:

`NORMDIST ( ` *x* ` , ` *mean* ` , ` *standard-deviation* ` , ` *cumulative-flag* ` )`

Description: Computes the normal distribution for the specified mean and standard deviation.

Mathematical Formula:

The equation for the normal density function (*cumulative-flag* = FALSE) is:

$$f(x, \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{(x-\mu)^2}{2\sigma^2}\right)}$$

When *cumulative-flag* = TRUE, the formula is the integral from negative infinity to x of the given formula.

where:

- x = argument x
- μ = argument *mean*
- σ = argument *standard-dev*

Arguments:

Name	Type	Description
x	number	The value for which the distribution is to be computed.
mean	number	The arithmetic mean of the distribution.
standard-deviation	number	The standard deviation of the distribution.
cumulative-flag	logical	Determines the form of the function. If TRUE, then the cumulative distribution function is returned; if FALSE, the probability mass function is returned.

Return Type and Value: number – The normal distribution for the specified mean and standard deviation.

However, if *standard-deviation* ≤ 0, `#NUM!` is returned.

[*Example*:

`NORMDIST(42,40,1.5,TRUE)` results in `0.90878878`
`NORMDIST(42,40,1.5,FALSE)` results in `0.10934005`

end example]

18.17.7.230 NORMINV

Syntax:

`NORMINV ( `*probability*` , `*mean*` , `*standard-deviation*` )`

Description: Computes the inverse of the normal distribution for the specified mean and standard deviation.**Arguments:**

Name	Type	Description
probability	number	The probability corresponding to the normal distribution.
mean	number	The arithmetic mean of the distribution.
standard-deviation	number	The standard deviation of the distribution.

Return Type and Value: number – The inverse of the normal distribution for the specified mean and standard deviation.

However, if

- *probability* < 0 or if *probability* > 1, `#NUM!` is returned.
- *standard-deviation* ≤ 0, `#NUM!` is returned.
- the implementation determines that a return value cannot be computed, `#N/A` is returned.

[*Example*:

`NORMINV(0.908789,40,1.5)` results in `42.00000201`

end example]

18.17.7.231 NORMSDIST

Syntax:

`NORMSDIST ( `*z*` )`

Description: Computes the standard normal distribution for the specified mean and standard deviation.

Mathematical Formula:

$$f(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}$$

where:

- z = argument z

Arguments:

Name	Type	Description
z	number	The value for which the distribution is to be computed.

Return Type and Value: number – The standard normal distribution for the specified mean and standard deviation.

[*Example*:

`NORMSDIST(1.333333)` results in `0.90878873`
`NORMSDIST(-1.5)` results in `0.06680720`

end example]

18.17.7.232 NORMSINV

Syntax:

`NORMSINV (` *probability* `)`

Description: Computes the inverse of the standard normal distribution. The distribution has a mean of zero and a standard deviation of 1.

Arguments:

Name	Type	Description
probability	number	The probability corresponding to the normal distribution.

Return Type and Value: number – The inverse of the standard normal distribution.

However, if

- *probability* < 0 or if *probability* > 1, `#NUM!` is returned.
- the implementation determines that a return value cannot be computed, #N/A is returned.

[*Example*:

`NORMSINV(0.945)` results in `1.59819314`
`NORMSINV(0.13)` results in `-1.12639113`

end example]

18.17.7.233 NOT

Syntax:

NOT (*logical-value*)

Description: Computes the logical negation of *logical-value*.

Arguments:

Name	Type	Description
logical-value	logical	The value to be negated.

Return Type and Value: logical – The logical negation of *logical-value*; that is, it returns TRUE if *logical-value* is FALSE, and FALSE if *logical-value* is TRUE.

[*Example*:

NOT(TRUE) results in FALSE
NOT(FALSE) results in TRUE
NOT(10>5) results in FALSE
NOT(16.567) results in FALSE

end example]

18.17.7.234 NOW

Syntax:

NOW ( )

Description: Computes the serial date-time of the current date and time, taking into account the current date system.

Arguments: None.

Return Type and Value: number – The serial date-time of the current date and time.

[*Example*: On February 26, 2006, between 23:01 and 23:02, NOW() resulted in 38774.95958611110 for the 1900 date system. On February 26, 2006, between 23:02 and 23:03, NOW() resulted in 37312.95982569440 for the 1904 date system. *end example*]

18.17.7.235 NPER

Syntax:

NPER (*rate* , *pmt* , *pv* [, [*fv*] [, [*type*]]])

Description: Computes the number of periods for an investment based on periodic, constant payments and a constant interest rate.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>rate</td><td>number</td><td>The interest rate per period.</td></tr>
<tr><td>pmt</td><td>number</td><td>The payment made each period; it cannot change over the life of the annuity. Typically, pmt contains principal and interest but no other fees or taxes.</td></tr>
<tr><td>pv</td><td>number</td><td>The present value, or the lump-sum amount that a series of future payments is worth right now.</td></tr>
<tr><td>fv</td><td>number</td><td>The future value, or a cash balance to be attained after the last payment is made. If fv is omitted, it is assumed to be 0 (i.e., the future value of a loan, for example, is 0).</td></tr>
<tr><td>type</td><td>number</td><td>The timing of the payment, truncated to integer, as follows:
<table><tr><th>Value</th><th>Timing</th></tr><tr><td><code>0</code></td><td>Payment at the end of the period</td></tr><tr><td><code>1</code></td><td>Payment at the beginning of the period</td></tr></table></td></tr>
</table>

Return Type and Value: number – The number of periods for an investment based on periodic, constant payments and a constant interest rate.

However, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`NPER(0.12/12,-100,-1000,10000,1)` results in `59.67`
`NPER(0.12/12,-100,-1000)` results in `-9.58`

end example]

18.17.7.236 NPV

Syntax:

NPV (*rate* , *argument-list*)

Description: Calculates the net present value of an investment by using a discount rate and a series of future payments and income.

The NPV investment begins one period before the date of the first *argument* cash flow and ends with the last cash flow in the list. The calculation is based on future cash flows. If the first cash flow occurs at the beginning of the first period, the first value shall be added to the NPV result, not included in *argument-list*.

Mathematical Formula:

If n is the number of cash flows in the list of values:

$$NPV = \sum_{i=1}^{n} \frac{values_i}{(1 + rate)^i}$$

where:

n is the number of elements in *argument-list*.
$values_i$valuesj is the value of the i-th element in *argument-list*.
rate is the *rate* argument.

Arguments:

Name	Type	Description
rate	number	The rate of discount over the length of one period.
argument-list	number	The *argument*s in *argument-list* designate the series of future payments (negative values) and income (positive values). *argument*s shall be equally spaced in time and occur at the end of each period. The order of *argument*s is significant. *argument*s that are numbers, empty cells, logical values, or text representations of numbers are included; *argument*s that are error values or text that cannot be translated into numbers are ignored. If an *argument* is an array or reference, only numbers in that array or reference are included. Empty cells, logical values, text, or error values in the array or reference are ignored.

Return Type and Value: number – Net present value of an investment by using a discount rate and a series of future payments and income.

[*Example*:

`NPV(0.1,-10000,3000,4200,6800)` results in `1188.44`

end example]

18.17.7.237 OCT2BIN

Syntax:

`OCT2BIN ( number [ , num-bin-digits ] )`

Description: Makes the binary equivalent of *number*, with the result having *num-bin-digits* digits.

Arguments:

Name	Type	Description
number	text	A 10-digit octal number in a string that is to be converted to a binary string. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (30th bit from the right) representing the sign bit.
num-bin - digits	number	The number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-bin-digits* is ignored and the result has 10 digits. If *num-bin-digits* is omitted, the minimum number of digits is used in the result. *num-bin-digits* is truncated to an integer.

Return Type and Value: text – The binary equivalent of *number*.

However, if

- *number* is outside the range `"7777777000"` (111111111111111111111000000000 binary, -512 decimal) to `"777"` (000000000000000000000111111111 binary, 511 decimal), inclusive, `#NUM!` is returned.
- *number* contains one or more non-octal digits, `#NUM!` is returned.
- *number* contains more than 10 octal digits, `#NUM!` is returned.
- *number* needs more digits that *num-bin-digits*, `#NUM!` is returned.
- *num-bin-digits* ≤ 0 or > 10, `#NUM!` is returned.

[*Example*:

`OCT2BIN("67")` results in `110111`
`OCT2BIN("7777777776")` results in `1111111110`
`OCT2BIN("7",5)` results in `00111`

end example]

18.17.7.238 OCT2DEC

Syntax:

`OCT2DEC ( number )`

Description: Makes the decimal equivalent of *number*.

Arguments:

Name	Type	Description
number	number	A 10-digit octal number in a string that is to be converted to a decimal number. If *number* has less than 10 digits, leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (30th bit from the right) representing the sign bit.

Return Type and Value: number – The decimal equivalent of *number*.

However, if

- *number* contains one or more non-octal digits, `#NUM!` is returned.
- *number* contains more than 10 octal digits; that is, *number* is outside the range `"4000000000"` (-536,870,912 decimal) to `"3777777777"` (536,870,911 decimal), inclusive, `#NUM!` is returned.

[*Example*:

`OCT2DEC("67")` results in `55`
`OCT2DEC("7777777776")` results in `-2`
`OCT2DEC("7000000000")` results in `-134217728`

end example]

18.17.7.239 OCT2HEX

Syntax:

`OCT2HEX ( number [ , num-hex-digits ] )`

Description: Makes the hexadecimal equivalent of *number*, with the result having *num-hex-digits* digits.

Arguments:

Name	Type	Description
number	text	A 10-digit octal number in a string that is to be converted to a hexadecimal string. If *number* has less than 10 digits,

Name	Type	Description
		leading zero digits are implied until it has exactly 10 digits. The 10 digits use twos-complement representation with the left-most bit (30th bit from the right) representing the sign bit.
num-hex-digits	number	*num-hex-digits* is the number of digits in the result, with leading zeros added as necessary. However, if *number* is negative, *num-hex-digits* is ignored and the result has 10 digits. If *num-hex-digits* is omitted, the minimum number of digits is used in the result. *num-hex-digits* is truncated to an integer.

Return Type and Value: text – The hexadecimal equivalent of *number*.

However, if

- *number* contains one or more non-octal digits, `#NUM!` is returned.
- *number* contains more than 10 octal digits; that is, *number* is outside the range `"4000000000"` (20000000 hex, -536,870,912 decimal) to `"3777777777"` (1FFFFFFF hex, 536,870,911 decimal), inclusive, `#NUM!` is returned.
- *number* needs more digits that *num-hex-digits*, `#NUM!` is returned.
- *num-hex-digits* ≤ 0or > 10, `#NUM!` is returned.

[*Example*:

`OCT2HEX("777")` results in `1FF`
`OCT2HEX("7777777776")` results in `FFFFFFFFFE`
`OCT2HEX("7",5)` results in `00007`

end example]

18.17.7.240 ODD

Syntax:

`ODD ( x )`

Description: Computes *x* rounded to the nearest odd integer, away from zero. Regardless of the sign of *x*, a value is rounded up when adjusted away from zero.

Arguments:

Name	Type	Description
x	number	The value to be rounded.

Return Type and Value: number – The rounded value of x.

[*Example*:

`ODD(1.5)` rounds 1.5 up to the nearest odd integer; that is, to 3
`ODD(3)` rounds 3 up to the nearest odd integer; that is, to 3
`ODD(2)` rounds 2 up to the nearest odd integer; that is, to 3
`ODD(-1)` rounds -1 up to the nearest odd integer; that is, to `-1`
`ODD(-2)` rounds -2 up to the nearest odd integer; that is, to `-3`

end example]

18.17.7.241 ODDFPRICE

Syntax:

`ODDFPRICE (` *settlement* `,` *maturity* `,` *issue* `,` *first-coupon* `,` *rate* `,` *yld* `,` *redemption* `,` *frequency* `[ , [` *basis* `] ] )`

Description: Computes the price per $100 face value of a security having an odd (short or long) first period.

Mathematical Formula:

Odd short first coupon:

$$ODDFPRICE = \left[\frac{redemption}{\left(1+\frac{yld}{frequency}\right)^{\left(N-1+\frac{DSC}{E}\right)}}\right] + \left[\frac{100 \times \frac{rate}{frequency} \times \frac{DFC}{E}}{\left(1+\frac{yld}{frequency}\right)^{\frac{DSC}{E}}}\right] + \left[\sum_{k=2} \frac{100 \times \frac{rate}{frequency}}{\left(1+\frac{yld}{frequency}\right)^{\left(k-1+\frac{DSC}{E}\right)}}\right] - \left[100 \times \frac{rate}{frequency} \times \frac{A}{E}\right]$$

where:

- A = number of days from the beginning of the coupon period to the settlement date (accrued days).
- DFC = number of days from the beginning of the odd first coupon to the first coupon date.
- DSC = number of days from the settlement to the next coupon date.
- E = number of days in the coupon period.
- $frequency$ = argument $frequency$
- N = number of coupons payable between the settlement date and the redemption date. (If this number contains a fraction, it is raised to the next whole number.)
- $rate$ = argument $rate$
- $redemption$ = argument $redemption$

- *yld* = argument *yld*

Odd long first coupon:

$$\mathrm{ODDFPRICE} = \left[\frac{\mathrm{redemption}}{\left(1+\frac{\mathrm{yld}}{\mathrm{frequency}}\right)^{\left(\mathrm{N}+\mathrm{N_q}+\frac{\mathrm{DSC}}{\mathrm{E}}\right)}}\right] + \left[\frac{100 \times \frac{\mathrm{rate}}{\mathrm{frequency}} \times \left[\sum_{\mathrm{i}=1}^{\mathrm{NC}} \frac{\mathrm{DC_i}}{\mathrm{NL_i}}\right]}{\left(1+\frac{\mathrm{yld}}{\mathrm{frequency}}\right)^{\mathrm{N_q}+\frac{\mathrm{DSC}}{\mathrm{E}}}}\right] + \left[\sum_{\mathrm{k}=1}^{\mathrm{N}} \frac{100 \times \frac{\mathrm{rate}}{\mathrm{frequency}}}{\left(1+\frac{\mathrm{yld}}{\mathrm{frequency}}\right)^{\left(\mathrm{k}-\mathrm{N_q}+\frac{\mathrm{DSC}}{\mathrm{E}}\right)}}\right] - \left[100 \times \frac{\mathrm{rate}}{\mathrm{frequency}} \times \sum_{\mathrm{i}=1}^{\mathrm{NC}} \frac{\mathrm{A_i}}{\mathrm{NL_i}}\right]$$

where:

- $\mathrm{A_i}$ = number of days from the beginning of the ith, or last, quasi-coupon period within odd period.
- $\mathrm{DC_i}$ = number of days from dated date (or issue date) to first quasi-coupon (i = 1) or number of days in quasi-coupon (i = 2,…, i = NC).
- *DSC* = number of days from settlement to next coupon date.
- *E* = number of days in coupon period.
- *frequency* = argument *frequency*
- *N* = number of coupons payable between the first real coupon date and redemption date. (If this number contains a fraction, it is raised to the next whole number.)
- *NC* = number of quasi-coupon periods that fit in odd period. (If this number contains a fraction, it is raised to the next whole number.)
- $\mathrm{NL_i}$ = normal length in days of the full ith, or last, quasi-coupon period within odd period.
- $\mathrm{N_q}$ = number of whole quasi-coupon periods between settlement date and first coupon.
- *rate* = argument *rate*
- *redemption* = argument *redemption*
- *yld* = argument *yld*

…**Arguments**:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
issue	number	The security's issue date.
first-coupon	number	The security's first coupon date.
rate	number	The security's interest rate.
yld	number	The security's annual yield.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>redemption</td><td>number</td><td>The security's redemption value per $100 face value.</td></tr>
<tr><td>frequency</td><td>number</td><td>the number of coupon payments per year. (For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4.)</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days;</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description	
			otherwise it is 365 days.
		2	Actual/360. Similar to Basis 1, but only has 360 days per year.
		3	Actual/365. Similar to Basis 1, but always has 365 days per year.
		4	European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments: • If the date is 28 or 29 February, it is adjusted to 30 February. • For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February..

Time information in the date arguments is ignored.

Return Type and Value: number – The price per $100 face value of a security having an odd (short or long) first period.

However, if

- *settlement*, *maturity*, *issue*, or *first-coupon* is out of range for the current date system, `#NUM!` is returned.
- The following is not true: *maturity* is later than *first-coupon*, which is later than *settlement*, which is later than *issue*, so `#NUM!` is returned.
- *rate* or *yld* < 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`ODDFPRICE(DATE(2008,11,11),DATE(2021,3,1),DATE(2008,10,15),DATE(2009,3,1),`
`0.0785,0.0625,100,2,1)` results in `113.5977`

end example]

18.17.7.242 ODDFYIELD

Syntax:

`ODDFYIELD (` *settlement* `,` *maturity* `,` *issue* `,` *first-coupon* `,` *rate* `,` *pr* `,` *redemption* `,` *frequency* `[ , [` *basis* `] ] )`

Description: Computes the yield of a security that has an odd (short or long) first period.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
issue	number	The security's issue date.
first-coupon	number	The security's first coupon date.
rate	number	The security's interest rate.
pr	number	The security's price.
redemption	number	The security's redemption value per $100 face value.
frequency	number	the number of coupon payments per year. (For annual payments, *frequency* is 1; for semiannual payments, *frequency* is 2; for quarterly payments, *frequency* is 4.)
basis	number	The truncated integer type of day count basis to use, as follows: Value: Day Count Basis `0` or omitted: US (NASD) 30/360. Assumes that each

Name	Type	Description
		<table><tr><td></td><td>month has 30 days and the total number of days in the year is 360 by making the following adjustments: - If the date is 28 or 29 February, it is adjusted to 30 February. - For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change. </td></tr><tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr><tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr><tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr><tr><td>4</td><td>European 30/360. The European method for</td></tr></table>

Name	Type	Description
		<table><tr><td></td><td>adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments: - If the date is 28 or 29 February, it is adjusted to 30 February. - For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February. </td></tr></table>

Time information in the date arguments is ignored.

Return Type and Value: number – The yield of a security that has an odd (short or long) first period.

However, if

- *settlement*, *maturity*, *issue*, or *first-coupon* is out of range for the current date system, `#NUM!` is returned.
- The following is not true: *maturity* is later than *first-coupon*, which is later than *settlement*, which is later than *issue*, so `#NUM!` is returned.
- *rate* or *pr* < 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`ODDFYIELD(DATE(2008,11,11),DATE(2021,3,1),DATE(2008,10,15),DATE(2009,3,1), 0.0575,84.5,100,2,0)` results in `7.7246%`

end example]

18.17.7.243 ODDLPRICE

Syntax:

`ODDLPRICE (` *settlement* `,` *maturity* `,` *last-interest* `,` *rate* `,` *yld* `,` *redemption* `,` *frequency* `[ , [` *basis* `] ] )`

Description: Computes the price per $100 face value of a security having an odd (short or long) last coupon period.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>last-interest</td><td>number</td><td>The security's last coupon date.</td></tr>
<tr><td>rate</td><td>number</td><td>The security's interest rate.</td></tr>
<tr><td>yld</td><td>number</td><td>The security's annual yield.</td></tr>
<tr><td>redemption</td><td>number</td><td>The security's redemption value per $100 face value.</td></tr>
<tr><td>frequency</td><td>number</td><td>the number of coupon payments per year. (For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4.)</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table><tr><th>Value</th><th>Day Count Basis</th></tr><tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date</td></tr></table></td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
• If the date is 28 or 29 February, it is adjusted to 30</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
		February. • For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.

Time information in the date arguments is ignored.

Return Type and Value: number – The price per $100 face value of a security having an odd (short or long) last coupon period.

However, if

- *settlement*, *maturity*, or *last-interest* is out of range for the current date system, `#NUM!` is returned.
- The following is not true: *maturity* is later than *settlement*, which is later than *last-interest*, so `#NUM!` is returned.
- *rate* or *yld* < 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`ODDLPRICE(DATE(2008,11,11),DATE(2021,3,1),DATE(2008,10,15),`
`0.0785,0.0625,100,2,1)` results in `99.8783`

end example]

18.17.7.244 ODDLYIELD

Syntax:

`ODDLYIELD (` *settlement* `,` *maturity* `,` *last-interest* `,` *rate* `,` *pr* `,` *redemption* `,`
frequency `[ , [` *basis* `] ] )`

Description: Computes the yield of a security that has an odd (short or long) last period.

Mathematical Formula:

$$ODDLYIELD = \left[\frac{\left(redemption + \left(\left(\sum_{i=1}^{NC}\frac{DC_i}{NL_i}\right) \times \frac{100 \times rate}{frequency}\right)\right) - \left(par + \left(\left(\sum_{i=1}^{NC}\frac{A_i}{NL_i}\right) \times \frac{100 \times rate}{frequency}\right)\right)}{pr + \left(\left(\sum_{i=1}^{NC}\frac{A_i}{NL_i}\right) \times \frac{100 \times rate}{frequency}\right)}\right] \times \left[\frac{frequency}{\left(\sum_{i=1}^{NC}\frac{DSC_i}{NL_i}\right)}\right]$$

where:

- A_i = number of accrued days for the i[th], or last, quasi-coupon period within odd period counting forward from last interest date before redemption.
- DC_i = number of days counted in the i[th], or last, quasi-coupon period as delimited by the length of the actual coupon period.
- *frequency* = argument *frequency*
- *NC* = number of quasi-coupon periods that fit in odd period; if this number contains a fraction it is raised to the next whole number.
- NL_i = normal length in days of the i[th], or last, quasi-coupon period within odd coupon period.
- *par* = argument *par*
- *rate* = argument *rate*
- *redemption* = argument *redemption*
-

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
last-interest	number	The security's last coupon date.
rate	number	The security's interest rate.
pr	number	The security's price.
redemption	number	The security's redemption value per $100 face value.
frequency	number	the number of coupon payments per year. (For annual payments, *frequency* is 1; for semiannual payments, *frequency* is 2; for quarterly payments, *frequency* is 4.)
basis	number	The truncated integer type of day count basis to use, as follows:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr>
</table>
</td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The yield of a security that has an odd (short or long) last period.

However, if

- *settlement*, *maturity*, or *last-interest* is out of range for the current date system, `#NUM!` is returned.
- The following is not true: *maturity* is later than *settlement*, which is later than *last-interest*, so `#NUM!` is returned.
- *rate* or *pr* < 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.

- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`ODDLYIELD(DATE(2008,11,11),DATE(2021,3,1),DATE(2008,10,15),`
`0.0575,84.5,100,2,0)` results in `4.5192%`

end example]

18.17.7.245 OFFSET

Syntax:

`OFFSET ( `*reference*` , `*rows*` , `*cols*` [ , [ `*height*` ] [ , [ `*width*` ]]] )`

Description: Gets a reference to a range that is a specified number of rows and columns from a cell or range of cells. The reference that is returned can be a single cell or a range of cells. You can specify the number of rows and the number of columns to be returned.

Arguments:

Name	Type	Description
reference	reference	Designates the base. *reference* shall refer to a cell or range of adjacent cells.
rows	number	The number of rows, up or down, that indicates the upper-left cell of the result to refer to. A positive value means below the starting reference; a negative value means above the starting reference.
cols	number	The number of columns, to the left or right, that the upper-left cell of the result to refer to. A positive value means to the right of the starting reference; a negative value means to the left of the starting reference.
height	number	The height, in rows, of the set of cells referred to by the resulting reference. This height shall be positive. If omitted, it is the same as the height of *reference*.
width	number	The width, in columns, of the set of cells referred to by the resulting reference. The width shall be positive. If omitted, it is the same as the width of *reference*.

Return Type and Value: reference – A reference to a range that is a specified size and number of rows and columns from a cell or range of cells.

However, if

- *reference* does not refer to a cell or range of adjacent cells, `#VALUE!` is returned.

- The combination of *rows* and *cols* results outside the worksheet, #REF! is returned.

[*Example*:

OFFSET(C3,2,3,1,1) results in the value in cell F5
SUM(OFFSET(C3:E5,-1,0,3,3)) results in the sum of the range C2:E4

end example]

18.17.7.246 OR

Syntax:

OR (*argument-list*)

Description: Tests if any one or more *arguments* in *argument-list* are TRUE. The function evaluates all arguments prior to returning a value.

Arguments:

Name	Type	Description
argument-list	logical, array, reference.	The *arguments* in *argument-list* designate the values to be tested. For an array or cell reference, a cell that contains text or is empty shall be ignored.

Return Type and Value: logical – TRUE if any one or more *arguments* in *argument-list* are TRUE; otherwise, FALSE.

However, if no logical values are found, the return value is unspecified.

[*Example*:

OR(TRUE) results in TRUE
OR(FALSE,FALSE) results in FALSE
OR(10=5,3=1+2,0) results in TRUE
OR({10,5,6,7},TRUE,E6:F6) results in TRUE, when E6 contains FALSE and F6 contains 0

end example]

18.17.7.247 PEARSON

Syntax:

PEARSON (*array-1* , *array-2*)

Description: Computes the Pearson product moment correlation coefficient, a dimensionless index that ranges from -1.0 to 1.0, inclusive, and reflects the extent of a linear relationship between two data sets.

Mathematical Formula:

The formula for the Pearson product moment correlation coefficient, r, is:

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2\sum(y-\bar{y})^2}}$$

where:

- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*array-1*)
- y = a sample value
- $\bar{y}$ = the sample mean AVERAGE(*array-2*)

Arguments:

Name	Type	Description
array-1	number, name, array, reference to number	The set of independent numerical values. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
array-2	number, name, array, reference to number	The set of dependent numerical values. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The Pearson product moment correlation coefficient.

However, if

- *array-1* and *array-2* have a different number of data points, the return value is unspecified.
- *array-1* or *array-2* is empty, the return value is unspecified.

[*Example*:

`PEARSON({9,7,5,3,1},{10,6,1,5,3})` results in `0.699378606`

end example]

18.17.7.248 PERCENTILE

Syntax:

`PERCENTILE ( `*array*` , `*k*` )`

Description: Computes the k^{th} percentile of a set of values in a range.

Arguments:

Name	Type	Description
array	array, reference	The set of numerical data that defines relative standing.
k	number	The percentile value in the range 0–1, inclusive. If *k* is not a multiple of 1/(n - 1), `PERCENTILE` interpolates to determine the value at the k^{th} percentile.

Return Type and Value: number – The k^{th} percentile of a set of values in a range.

However, if

- *array* is empty, the return value is unspecified.
- *k* is < 0 or *k* > 1, `#NUM!` is returned.

[*Example*:

`PERCENTILE({1,3,2,4},0.3)` results in `1.9`
`PERCENTILE({1,3,2,4},0.75)` results in `3.25`

end example]

18.17.7.249 PERCENTRANK

Syntax:

PERCENTRANK (*array* , *x* [, *significance*])

Description: Computes the rank of a value in a data set as a percentage of the data set.

Arguments:

Name	Type	Description
array	array, reference	*array* is the set of numerical data that defines relative standing.
x	number	The value for which the rank is to be computed. If *x* does not match one of the values in *array*, PERCENTRANK interpolates to return the correct percentage rank.
significance	number	The number of significant digits for the returned percentage value. If omitted, a value of 3 is used.

Return Type and Value: number – The rank of a value in a data set as a percentage of the data set.

However, if

- *array* is empty, the return value is unspecified.
- *significance* < 1, `#NUM!` is returned.

[*Example*:

`PERCENTRANK({12,6,7,9,3,8},4)` results in `0.066`
`PERCENTRANK({12,6,7,9,3,8},5)` results in `0.133`

end example]

18.17.7.250 PERMUT

Syntax:

`PERMUT (` *number* `,` *number-chosen* `)`

Description: Computes the number of permutations for *number-chosen* objects that can be selected from *number* objects. [*Note*: A permutation is any set or subset of objects or events where internal order is significant. Permutations are different from combinations, for which the internal order is not significant. Use this function for lottery-style probability calculations. *end note*]

Mathematical Formula:

$$P_{k,n} = \frac{n!}{(n-k)!}$$

where:

- k = argument *number-chosen*
- n = argument *number*

Arguments:

Name	Type	Description
number	number	The total number of items available, truncated to integer.
number-chosen	number	The number of items in each permutation, truncated to integer.

Return Type and Value: number – The number of different permutations of *number-chosen* in *number*.

However, if

- *number* < 0, `#NUM!` is returned.
- *number-chosen* < 0, `#NUM!` is returned.
- *number* < *number-chosen*, `#NUM!` is returned.

[*Example*:

`PERMUT(8,2)` results in `56`
`PERMUT(10,4)` results in `5040`
`PERMUT(6,5)` results in `720`

end example]

18.17.7.251 PHONETIC

Syntax:

`PHONETIC ( ` *string* ` )`

Description: Extracts the phonetic (furigana) characters from *string*. [*Note*: Furigana are aids used to indicate correct pronunciation of Japanese text. *end note*]

Arguments:

Name	Type	Description
string	text, reference	Designates a furigana string. If *string* is a cell range, the furigana string in the upper-left corner cell of that range is returned.

Return Type and Value: text – The phonetic (furigana) characters from *string*.

However, if *string* is a range of non-contiguous cells, `#N/A` is returned.

18.17.7.252 PI

Syntax:

`PI ()`

Description: Computes the value π.

Arguments: None.

Return Type and Value: number – The value π.

[*Example*: The following results are displayed using 10 significant digits

`PI()` results in `3.141592654`
`PI()/2` results in `1.570796327`
`PI()*(2.5^2)` results in `19.63495408`

end example]

18.17.7.253 PMT

Syntax:

`PMT ( `*rate*` , `*nper*` , `*pv*` [ , [ `*fv*` ] [ , [ `*type*` ]]] )`

Description: Computes the payment for a loan based on constant payments and a constant interest rate.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>rate</td><td>number</td><td>The interest rate for the loan.</td></tr>
<tr><td>nper</td><td>number</td><td>The total number of payment for the loan.</td></tr>
<tr><td>pv</td><td>number</td><td>The present value, or the total amount that a series of future payments is worth now; also known as the principal.</td></tr>
<tr><td>fv</td><td>number</td><td>The future value, or a cash balance to be attained after the last payment is made. If omitted, it is assumed to be 0 (i.e., the future value of a loan, for example, is 0).</td></tr>
<tr><td>type</td><td>number</td><td>The timing of the payment, truncated to integer, as follows:
<table><tr><th>Value</th><th>Timing</th></tr><tr><td><code>0</code></td><td>Payment at the end of the period</td></tr><tr><td><code>1</code></td><td>Payment at the beginning of the period</td></tr></table></td></tr>
</table>

Return Type and Value: number – The payment for a loan based on constant payments and a constant interest rate. (The payment returned by PMT includes principal and interest but no taxes, reserve payments, or fees sometimes associated with loans.)

Howver, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`PMT(0.08/12,10,10000)` results in `-1,037.03`
`PMT(0.08/12,10,10000,0,1)` results in `-1,030.16`

end example]

18.17.7.254 POISSON

Syntax:

`POISSON ( x , mean , cumulative-flag )`

Description: Computes the Poisson distribution.

Mathematical Formula:

For *cumulative-flag* = FALSE:

$$POISSON = \frac{e^{-\lambda}\lambda^{x}}{x!}$$

For *cumulative-flag* = TRUE:

$$CUMPOISSON = \sum_{k=0}^{x} \frac{e^{-\lambda}\lambda^{k}}{k!}$$

where:

- x = argument x
- λ = argument *mean*

Arguments:

Name	Type	Description
x	number	The number of events, truncated to an integer.
mean	number	The expected numeric value.
cumulative-flag	logical	Determines the form of the function. If TRUE, `POISSON` returns the cumulative Poisson probability that the number of random events occurring are between zero and x, inclusive; if FALSE, it returns the Poisson probability mass function that the number of events occurring is exactly x.

Return Type and Value: number – The Poisson distribution.

However, if

- $x < 0$, `#NUM!` is returned.
- *mean* ≤ 0, `#NUM!` is returned.

[*Example*:

`POISSON(2,5,TRUE)` results in `0.124652019`
`POISSON(2,5,FALSE)` results in `0.084224337`

end example]

18.17.7.255 POWER

Syntax:

POWER (*x* , *y*)

Description: Computes x raised to the power y.

Arguments:

Name	Type	Description
x	number	The base and the number y is the exponent to which that base is raised.
y	number	The exponent to which the base is raised.

Return Type and Value: number – x^y.

However, if

- The value of x is negative and y is not a whole number, `#NUM!` is returned.
- x is zero and y is less than or equal to zero, `#DIV/0!` is returned.
- The result cannot be represented as a number, `#NUM!` is returned.

[*Example*:

`POWER(2,3)` results in `8`
`POWER(2,0.5)` results in `1.414213562`
`POWER(-1.234,5.0)` results in `-2.861381721`
`POWER(1.234,5.1)` results in `2.922182358`

end example]

18.17.7.256 PPMT

Syntax:

PPMT (*rate* , *per* , *nper* , *pv* [, [*fv*] [, [*type*]]])

Description: Computes the payment on the principal for a given period for an investment based on periodic, constant payments and a constant interest rate.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>rate</td><td>number</td><td>The interest rate per period.</td></tr>
<tr><td>per</td><td>number</td><td>The the period and shall be in the range 1–nper.</td></tr>
<tr><td>nper</td><td>number</td><td>The total number of payment in an annuity.</td></tr>
<tr><td>pv</td><td>number</td><td>The present value, or the total amount that a series of future payments is worth now.</td></tr>
<tr><td>fv</td><td>number</td><td>The future value, or a cash balance to be attained after the last payment is made. If omitted, it is assumed to be 0 (i.e., the future value of a loan, for example, is 0).</td></tr>
<tr><td>type</td><td>number</td><td>The timing of the payment, truncated to integer, as follows:
<table>
<tr><th>Value</th><th>Timing</th></tr>
<tr><td><code>0</code></td><td>Payment at the end of the period</td></tr>
<tr><td><code>1</code></td><td>Payment at the beginning of the period</td></tr>
</table>
</td></tr>
</table>

Return Type and Value: number – The payment on the principal for a given period for an investment based on periodic, constant payments and a constant interest rate.

However, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`PPMT(0.1/12,1,2*12,2000)` results in `-75.62`
`PPMT(0.08,10,10,200000)` results in `-27,598.05`

end example]

18.17.7.257 PRICE

Syntax:

`PRICE` (*settlement* , *maturity* , *rate* , *yld* , *redemption* , *frequency* [, [*basis*]])

Description: Computes the price per $100 face value of a security that pays periodic interest.

Mathematical Formula:

$$PRICE = \left[\frac{reemption}{\left(1+\frac{yld}{frequency}\right)^{\left(N-1+\frac{DSC}{E}\right)}}\right] + \left[\sum_{k=1}^{N}\frac{100\times\frac{rate}{frequency}}{\left(1+\frac{yld}{frequency}\right)^{\left(k-1+\frac{DSC}{E}\right)}}\right] - \left(100\times\frac{rate}{frequency}\times\frac{A}{E}\right)$$

where:

- A = number of days from beginning of coupon period to settlement date.
- DSC = number of days from settlement to next coupon date.
- E = number of days in coupon period in which the settlement date falls.
- *frequency* = argument *frequency*
- N = number of coupons payable between settlement date and redemption date.
- *rate* = argument *rate*
- *redemption* = argument *redemption*
- *yld* = argument *yld*

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>rate</td><td>number</td><td>The security's interest rate.</td></tr>
<tr><td>yield</td><td>number</td><td>The security's annual yield.</td></tr>
<tr><td>redemption</td><td>number</td><td>The security's redemption value per $100 face value.</td></tr>
<tr><td>frequency</td><td>number</td><td>the number of coupon payments per year. (For annual payments, frequency is 1; for semiannual payments, frequency is 2; for quarterly payments, frequency is 4.) frequency is truncated to an integer.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
		<table><tr><td></td><td>30 February. - For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change. </td></tr><tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr><tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr><tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr><tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments: - If the date </td></tr></table>

Name	Type	Description
		is 28 or 29 February, it is adjusted to 30 February. • For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.

Time information in the date arguments is ignored.

Return Type and Value: number – The price per $100 face value of a security that pays periodic interest.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *rate* or *yld* < 0, `#NUM!` is returned.
- *redemption* ≤ 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`PRICE(DATE(2008,2,15),DATE(2017,11,15),0.0575,0.065,100,2,0)` results in `94.6344`

end example]

18.17.7.258 PRICEDISC

Syntax:

PRICEDISC (*settlement* , *maturity* , *discount* , *redemption* [, [*basis*]])

Description: Computes the price per $100 face value of a discounted security.

Mathematical Formula:

$$PRICEDISC = redemption - discount \times redemption \times \frac{DSM}{B}$$

where:

- B = number of days in year, depending on year basis.
- DSM = number of days from settlement to maturity.
- $discount$ = argument $discount$
- $redemption$ = argument $redemption$

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
discount	number	The security's discount rate.
redemption	number	The security's redemption value per $100 face value.
basis	number	The truncated integer type of day count basis to use, as follows: <table><tr><th>Value</th><th>Day Count Basis</th></tr><tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments: - If the date is 28 or 29 February, it is adjusted to 30 February. - For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date </td></tr></table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
• If the date is 28 or 29 February, it is adjusted to 30 February.
• For months with 31 days, all dates with</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
		a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.

Time information in the date arguments is ignored.

Return Type and Value: number – The price per $100 face value of a discounted security.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *discount* or *redemption* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`PRICEDISC(DATE(2008,2,16),DATE(2008,3,1),0.0525,100,2)` results in `99.7958`

end example]

18.17.7.259 PRICEMAT

Syntax:

`PRICEMAT ( `*settlement*` , `*maturity*` , `*issue*` , `*rate*` , `*yld*` [ , [ `*basis*` ] ] )`

Description: Computes the price per $100 face value of a security that pays interest at maturity.

Mathematical Formula:

$$PRICEMAT = \frac{100 + \left(\frac{IM}{B} \times rate \times 100\right)}{1 + \left(\frac{DSM}{B} \times yld\right)} - \left(\frac{A}{B} \times rate \times 100\right)$$

where:

- A = number of days from issue to settlement.
- B = number of days in year, depending on year basis.
- DIM = number of days from issue to maturity.
- DSM = number of days from settlement to maturity.
- $rate$ = argument $rate$
- yld = argument yld

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>issue</td><td>number</td><td>The security's issue date.</td></tr>
<tr><td>rate</td><td>number</td><td>The security's interest rate.</td></tr>
<tr><td>yld</td><td>number</td><td>The security's annual yield.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:

If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or
</td></tr>
</table></td></tr>
</table>

Name	Type	Description

	29 February, in which case it does not change.
1	Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.
2	Actual/360. Similar to Basis 1, but only has 360 days per year.
3	Actual/365. Similar to Basis 1, but always has 365 days per year.
4	European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments: • If the date is 28 or 29 February, it is adjusted to 30 February. • For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations

Name	Type	Description
		where the first date is 28 or 29 February.

Time information in the date arguments is ignored.

Return Type and Value: number – The price per $100 face value of a security that pays interest at maturity.

However, if

- *settlement*, *maturity*, or *issue* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *rate* or *yld* < 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`PRICEMAT(DATE(2008,2,15),DATE(2008,4,13),DATE(2007,11,11),0.061,0.061,0)` results in `99.9845`

end example]

18.17.7.260 PROB

Syntax:

PROB (*x-range* , *probability-range* , *lower-limit* [, *upper-limit*])

Description: Computes the probability that values in a range are between two limits.

Arguments:

Name	Type	Description
x-range	array, reference	The set of numeric values of *x* with which there are associated probabilities.
probability-range	array, reference	A set of numeric probabilities associated with the values in *x-range*.
lower-limit	number	The lower bound on the value for which the probability is to be computed.
upper-limit	number	The upper bound on the value for which the probability is to be computed. If omitted, the probability that values in *x-range* are equal to *lower-limit* is returned.

Return Type and Value: number – The probability that values in a range are between two limits.

However, if

- Any value in *probability-range* ≤ 0 or any value in *probability-range* > 1, `#NUM!` is returned.
- The sum of the values in *probability-range* < 1, `#NUM!` is returned.
- *x-range* and *probability-range* contain a different number of data points, the return value is unspecified.

[*Example*:

`PROB({0,1,2,3},{0.2,0.3,0.1,0.4},2)` results in `0.1`
`PROB({0,1,2,3},{0.2,0.3,0.1,0.4},1,4)` results in `0.8`

end example]

18.17.7.261 PRODUCT

Syntax:

PRODUCT (*argument-list*)

Description: Multiplies the numeric values of *arguments* in *argument-list*.

Arguments:

Name	Type	Description
argument-list	logical, number, text, array, reference	The *arguments* in *argument-list* designate the numbers to be multiplied. Arguments that are numbers, logical values, or text representations of numbers shall be counted. If an argument is an array or reference, only numbers in that array or reference shall be counted. Empty cells, logical values, and text in the array or reference shall be ignored.

Return Type and Value: number – The product of the values of its arguments.

[*Example*:

`PRODUCT(1)` results in 1
`PRODUCT(1,2,3,4,5)` results in `120`
`PRODUCT({1,2;3,4})` results in 24
`PRODUCT({2,3},4,"5")` results in `120`

end example]

18.17.7.262 PROPER

Syntax:

PROPER (*string*)

Description: Makes a lowercase version of *string* except that the first letter in *string* and any other letters in *string* that immediately follow a character that is not a letter, are converted to uppercase.

Arguments:

Name	Type	Description
string	text, reference	Designates the string to be converted.

Return Type and Value: text – A version of *string* such that the first letter in *string* and any other letters in *string* that immediately follow a character that is not a letter, are converted to uppercase. All other letters are converted to lowercase, and all other non-letters are unchanged.

[*Example*:

`PROPER("12aBC d123aD#$%sd^")` results in `12Abc D123Ad#$%Sd^`
`PROPER(A10)` results in `12Abc D123Ad#$%Sd^`, when A10 contains `12aBC d123aD#$%sd^`

end example]

18.17.7.263 PV

Syntax:

PV (*rate* , *nper* , *pmt* [, [*fv*] [, [*type*]]])

Description: Computes the present value of an investment. (The present value is the total amount that a series of future payments is worth now.)

Mathematical Formula:

If *rate* is not 0, then:

$$pv * (1 + rate)^{nper} + pmt(1 + rate * type) * \left(\frac{(1 + rate)^{nper} - 1}{rate}\right) + fv = 0$$

If *rate* is 0, then:

$(\mathrm{pmt} \times \mathrm{nper}) + \mathrm{pv} + \mathrm{fv} = 0$(pmt * nper) + pv + fv = 0

where:

- *fv* = *fv* argument

- *nper* = *nper* argument
- *pmt* = *pmt* argument
- *rate* = *rate* argument
- *type* = *type* argument

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>rate</td><td>number</td><td>The interest rate per period.</td></tr>
<tr><td>nper</td><td>number</td><td>The total number of payment in an annuity.</td></tr>
<tr><td>pmt</td><td>number</td><td>The payment made each period and cannot change over the life of the annuity. If is omitted, fv shall be provided.
[Note: Typically, pmt includes principal and interest but no other fees or taxes. end note]</td></tr>
<tr><td>fv</td><td>number</td><td>The future value, or a cash balance to be attained after the last payment is made. If omitted, pmt shall be provided.</td></tr>
<tr><td>type</td><td>number</td><td>The timing of the payment, truncated to integer, as follows:
<table>
<tr><th>Value</th><th>Timing</th></tr>
<tr><td><code>0</code></td><td>Payment at the end of the period</td></tr>
<tr><td><code>1</code></td><td>Payment at the beginning of the period</td></tr>
</table>
</td></tr>
</table>

Return Type and Value: number – The present value of an investment.

However, if *type* is any number other than 0 or 1, `#NUM!` is returned.

[*Example*:

`PV(0.08/12,12*20,500,,0)` results in `-59,777.15`

end example]

18.17.7.264 QUARTILE

Syntax:

`QUARTILE (` *array* `,` *result-category* `)`

Description: Computes the quartile of a data set.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>array</td><td>array, reference</td><td>The set of numeric values for which the quartile value is to be computed.</td></tr>
<tr><td>result-category</td><td>number</td><td>When truncated to an integer, specifies which value is to be returned, as follows:
<table><tr><th>Value</th><th>Value Returned</th></tr><tr><td>0</td><td>Minimum value</td></tr><tr><td>1</td><td>First quartile (25th percentile)</td></tr><tr><td>2</td><td>Median value (50th percentile)</td></tr><tr><td>3</td><td>Third quartile (75th percentile)</td></tr><tr><td>4</td><td>Maximum value</td></tr></table></td></tr>
</table>

Return Type and Value: number – The quartile of a data set.

However, if

- *array* is empty, the return value is unspecified.
- *result-category* < 0 or *result-category* > 4, `#NUM!` is returned.

[*Example*:

`QUARTILE({1,2,4,7,8,9,10,12},1)` results in `3.5`

end example]

18.17.7.265 QUOTIENT

Syntax:

`QUOTIENT ( `*dividend*` , `*divisor*` )`

Description: Computes the integer portion of the division of *dividend* by *divisor*.

Arguments:

Name	Type	Description
dividend	number	The dividend
divisor	number	The divisor.

Return Type and Value: number – The integer portion of the division of *dividend* by *divisor*.

[*Example*:

`QUOTIENT(5,2)` results in `2`
`QUOTIENT(4.5,3.1)` results in `1`
`QUOTIENT(-10,3)` results in `-3`

end example]

18.17.7.266 RADIANS

Syntax:

`RADIANS ( `*angle*` )`

Description: Converts *angle* in degrees into radians.

Arguments:

Name	Type	Description
angle	number	The angle expressed in degrees that is to be converted into radians.

Return Type and Value: number – *angle* in radians.

[*Example*:

`RADIANS(360)` results in `6.283185307`
`RADIANS(270)` results in `4.71238898`
`RADIANS(45)` results in `0.785398163`
`RADIANS(8.5)` results in `0.148352986`

end example]

18.17.7.267 RAND

Syntax:

`RAND ( )`

Description: Computes an evenly distributed random real number greater than or equal to 0 and less than 1. A new random real number is returned every time the cell's value is calculated.

Arguments: None.

Return Type and Value: number – An evenly distributed random real number greater than or equal to 0 and less than 1.

[*Example*:

`RAND()` results in `0.437337454`
`INT(RAND()*(6-1)+1)` might result in 3

end example]

18.17.7.268 RANDBETWEEN

Syntax:

`RANDBETWEEN ( ` *lower-bound* ` , ` *upper-bound* ` )`

Description: Computes a random integer number in the range *lower-bound–upper-bound*. A new random integer number is returned every time the cell's value is calculated.

Arguments:

Name	Type	Description
lower-bound	number	The smallest integer that is returned.
upper-bound	number	The largest integer that is returned.

Return Type and Value: number – A random integer number in the range specified.

However, if *lower-bound* is greater than *upper-bound*, `#NUM!` is returned.

[*Example*:

`RANDBETWEEN(1,6)` results in an integer between `1` and `6`, inclusive
`RANDBETWEEN(-10,10)` results in an integer between `-10` and `10`, inclusive

end example]

18.17.7.269 RANK

Syntax:

`RANK ( ` *number* ` , ` *number-list* ` [ , ` *order* ` ] )`

Description: Computes the rank of a number in a list of numbers. RANK gives duplicate numbers the same rank. However, the presence of duplicate numbers affects the ranks of subsequent numbers.

Arguments:

Name	Type	Description
number	number	The number whose rank is to be found.

Name	Type	Description
number-list	reference	Designates the list of numbers. Non-numeric values in this list are ignored.
order	number	Specifies how *number* is to be ranked. If zero or omitted, *number* is ranked as if the list were sorted in descending order. If *order* is any non-zero value, *number* is ranked as if the list were sorted in ascending order.

Return Type and Value: number – The rank of a number in a list of numbers.

[*Example*:

When the cells `E1:I1` contain 7, 3.5, 3.5, 1, and 2

`RANK(E2,E1:I1,1)` results in 3
`RANK(E2,E1:I1,1,0)` results in 2
`RANK(E2,E1:I1,1)` results in 2
`RANK(E1,E1:I1,1)` results in 5, as the two 3.5 values both have a rank of 3; no value has rank 4.

end example]

18.17.7.270 RATE

Syntax:

RATE (*nper* , *pmt* , *pv* [, [[*fv*] [, [[*type*][, [*guess*]]]]]])

Description: Computes the interest rate per period of an annuity, using iteration, which can result in zero or more solutions.

Arguments:

Name	Type	Description
nper	number	The total number of payment periods.
pmt	number	The payment made each period and cannot change over the life of the annuity. (Typically, *pmt* includes principal and interest but no other fees or taxes.) If omitted, *fv* shall be present.
pv	number	The present value.
fv	number	The future value, or a cash balance to be attained after the last payment is made. If omitted, it is assumed to be 0 (i.e., the future value of a loan, for example, is 0).
type	number	The timing of the payment, truncated to integer, as follows:

Name	Type	Description
		<table><tr><th>Value</th><th>Timing</th></tr><tr><td>`0`</td><td>Payment at the end of the period</td></tr><tr><td>`1`</td><td>Payment at the beginning of the period</td></tr></table>
guess	number	A guess for what the rate are. If omitted, it is assumed to be 10 percent.

Return Type and Value: number – The interest rate per period of an annuity.

However, if

- *type* is any number other than 0 or 1, `#NUM!` is returned.
- The result has not converged after an implementation-defined number of iterations, `#NUM!` is returned.

[*Example*:

`RATE(4*12,-200,8000)` results in `0.7701%`
`RATE(4*12,-200,8000)*12` results in `9.2418%`

end example]

18.17.7.271 RECEIVED

Syntax:

`RECEIVED (` *settlement* `,` *maturity* `,` *investment* `,` *discount* `[ , [` *basis* `]] )`

Description: Computes the amount received at maturity for a fully invested security.

Mathematical Formula:

$$RECEIVED = \frac{investment}{1 - \left(discount \times \frac{DIM}{B}\right)}$$

where:

- B = number of days in a year, depending on the year basis.
- DIM = number of days from issue to maturity.
- $discount$ = argument $discount$
- $investment$ = argument $investment$

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>investment</td><td>number</td><td>The amount invested in the security.</td></tr>
<tr><td>discount</td><td>number</td><td>The security's discount rate.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
		<table><tr><td></td><td>year is 366 days; otherwise it is 365 days.</td></tr><tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr><tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr><tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments: - If the date is 28 or 29 February, it is adjusted to 30 February. - For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February. </td></tr></table>

Time information in the date arguments is ignored.

Return Type and Value: number – The amount received at maturity for a fully invested security.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *investment* or *discount* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`RECEIVED(DATE(2008,2,15),DATE(2008,5,15),1000000,0.0575,2)` results in `1014584.65`

end example]

18.17.7.272 REPLACE

Syntax:

REPLACE (*string-1* , *start-pos* , *number-chars* , *string-2*)

Description: Produces a new string that is *string-1* with *number-chars* characters starting at position *start-pos*, replaced by *string-2*. (REPLACE is intended for use with languages that use the single-byte character set (SBCS), whereas REPLACEB (§18.17.7.273) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string-1	text	Designates a string.
start-pos	number	The number of the start position within *string-1* from which characters in *string-1* are to be replaced. The start position of the first character is 1. *start-pos* shall be at least 0. If *start-pos* is beyond the end of *string-1*, the result is a new string that is *string-2* appended to *string-1*. If *start-pos* is within the bounds of *string-1*, but *number-chars* goes beyond the end of *string-1*, the characters starting at position *start-pos* through to the end of *string-1* shall be replaced by *string-2*.
number-chars	number	The number of characters within *string-1* that are to be replaced by the string designated by *string-2*.
string-2	text	Designates a string.

Return Type and Value: text – A copy of *string-1* with replacement characters from *string-2*.

However, if

- *start-pos* < 0, #VALUE! is returned.
- *number-chars* < 0, #VALUE! is returned.

[*Example*:

`REPLACE("abcdefghijk",3,4,"XY")` results in `abXYghijk`
`REPLACE("abcdefghijk",3,1,"12345")` results in `ab12345defghijk`
`REPLACE("abcdefghijk",15,4,"XY")` results in `abcdefghijkXY`

end example]

18.17.7.273 REPLACEB

Syntax:

REPLACEB (*string-1* , *start-pos* , *number-bytes* , *string-2*)

Description: Produces a new string that is *string-1* with *number-bytes* bytes starting at position *start-pos*, replaced by *string-2*. (REPLACEB is intended for use with languages that use the double-byte character set (DBCS), whereas REPLACE (§18.17.7.272) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string-1	text	Designates a string.
start-pos	number	The number of the start position within *string-1* from which characters in *string-1* are to be replaced. The start position of the first character is 1. *start-pos* shall be at least 0. If *start-pos* is beyond the end of *string-1*, the result is a new string that is *string-2* appended to *string-1*. If *start-pos* is within the bounds of *string-1*, but *number- bytes* goes beyond the end of *string-1*, the characters starting at position *start-pos* through to the end of *string-1* shall be replaced by *string-2*.
number-bytes	number	The number of characters within *string-1* that are to be replaced by the string designated by *string-2*.
string-2	text	Designates a string.

Return Type and Value: text – A copy of *string-1* with replacement characters from *string-2*.

However, if

- *start-pos* < 0, #VALUE! is returned.
- *number-bytes* < 0, #VALUE! is returned.

[*Example*: Assuming 1-byte characters:

`REPLACEB("abcdefghijk",3,4,"XY")` results in `abXYghijk`
`REPLACEB("abcdefghijk",3,1,"12345")` results in `ab12345defghijk`
`REPLACEB("abcdefghijk",15,4,"XY")` results in `abcdefghijkXY`

end example]

18.17.7.274 REPT

Syntax:

`REPT ( `*`string`*` , `*`replication-count`*` )`

Description: Creates a string that is *replication-count* number of occurrences of *string* concatenated together.

Arguments:

Name	Type	Description
string	text	Designate the string to be replicated.
replication-count	number	The number of times *string* is to be replicated, truncated to integer. If *replication-count* is 0, the resulting string is empty.

Return Type and Value: text – The final replicated string.

However, if *replication-count* < 0, `#VALUE!` is returned.

[*Example*:

`REPT("ABC",3)` results in ABCABCABC
`LEN(REPT("ABC",0))` results in `0`

end example]

18.17.7.275 RIGHT

Syntax:

`RIGHT ( `*`string`*` [ , `*`number-chars`*` ] )`

Description: Extracts the right-most *number-chars* characters from *string*. (`RIGHT` is intended for use with languages that use the single-byte character set (SBCS), whereas `RIGHTB` (§18.17.7.276) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-chars	number	The number of characters to be extracted. If omitted, a count of 1 shall be assumed. *number-chars* shall be at least 0. If *number-chars* exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing the right-most *number-chars* characters from *string*.

However, if *number-chars* < 0, #VALUE! is returned.

[*Example*:

`RIGHT("abcdef",2)` results in `ef`
`RIGHT(A10,4)` results in `z123`, when A10 contains `xyz123`

end example]

18.17.7.276 RIGHTB

Syntax:

RIGHTB (*string* , [*number-bytes*])

Description: Extracts the right-most *number-bytes*-worth of characters from *string*. (RIGHTB is intended for use with languages that use the double-byte character set (DBCS), whereas RIGHT (§18.17.7.275) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string	text	Designate the string from which a substring is to be extracted.
number-bytes	number	The number of bytes to be extracted. If omitted, a count of 1 shall be assumed. *number-bytes* shall be at least 0. If *number-bytes* exceeds the length of *string*, the whole of *string* shall be extracted.

Return Type and Value: text – A string containing the right-most *number-bytes*-worth of characters from *string*.

However, if *number-bytes* < 0, #VALUE! is returned.

[*Example*: Assuming 1-byte characters:

`RIGHTB("abcdef",2)` results in `ef`
`RIGHTB(A10,4)` results in `z123`, when A10 contains `xyz123`

end example]

18.17.7.277 ROMAN

Syntax:

ROMAN (*number* , *form*)

Description: Converts the Arabic number, *number*, to a Roman number according to *form*.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>number</td><td>number</td><td>The Arabic number to be converted.</td></tr>
<tr><td>form</td><td>number</td><td>Specifies the type of Roman numeral to be produced. The Roman numeral style ranges from Classic to Simplified, becoming more concise as the value of form increases, as follows:
<table>
<tr><th>Value</th><th>Type</th></tr>
<tr><td>0, omitted, or TRUE</td><td>Classic. Only subtract powers of ten (but not L or V). Do not subtract a number from one that is more than 10 times greater. If another letter follows the larger one, it shall be smaller than the number preceding the larger one.</td></tr>
<tr><td>1</td><td>Concise. Allow subtractraction of L and V as well as powers of ten. Do not subtract a number from one that is more than 10 times greater. If another letter follows the larger one, it shall be smaller than the number preceding the larger one.</td></tr>
<tr><td>2</td><td>More concise. Allow subtractraction of L (but not V) as well as powers of ten. Allow subtraction of a number from one that is more than 10 times greater. If another letter follows the larger one, it shall be smaller than the number preceding the larger one.</td></tr>
<tr><td>3</td><td>Most concise. Allow subtractraction of L and V as well as powers of ten. Allow subtraction of a number from one that</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description
		<table><tr><td></td><td>is more than 10 times greater. If another letter follows the larger one, it shall be smaller than the number preceding the larger one.</td></tr><tr><td>4 or FALSE</td><td>Simplified. Produce the fewest Roman digits.</td></tr></table>

Return Type and Value: text – The corresponding Roman number.

However, if

- *number* < 0 or > 3999, #VALUE! is returned.
- *form* is not one of the values listed above, #VALUE! is returned.

[*Example*:

ROMAN(499,0) results in CDXCIX, which is 100 less than 500, plus 10 less than 100, plus one less than 10.
ROMAN(499,1) results in LDVLIV, which is 50 less than 500, plus 5 less than 50, plus one less than 5.
ROMAN(499,2) results in XDIX, which is 10 less than 500, plus one less than 10.
ROMAN(499,3) results in VDIV, which is 5 less than 500, plus one less than 5.
ROMAN(499,4) results in ID, which is 1 less than 500.
ROMAN(2013,0) results in MMXIII, which is 2,000, plus 10, plus 3.

end example]

18.17.7.278 ROUND

Syntax:

ROUND (*x* , *number-digits*)

Description: Rounds *x* to the number of digits specified by *number-digits*.

Arguments:

Name	Type	Description
x	number	The value to be rounded.
number-digits	number	The number of digits to which *x* is to be rounded. If *number-digits* is greater than 0, *x* is rounded to the specified number of decimal places. If *number-digits* is 0, *x* is rounded to the nearest integer. If *number-digits* is less than 0, *x* is rounded to the left of the decimal point.

Return Type and Value: number – The rounded value of *x*. When rounding, digits 0–4 round down, while digits 5–9 round up.

[*Example*:

`ROUND(2.15,1)` results in `2.2`
`ROUND(2.149,1)` results in `2.1`
`ROUND(-1.475,2)` results in `-1.48`
`ROUND(21.5,-1)` results in `20`

end example]

18.17.7.279 ROUNDDOWN

Syntax:

`ROUNDDOWN ( x , number-digits )`

Description: Computes *x* rounded down, toward zero, to the number of digits specified by *number-digits*. [*Note*: ROUNDDOWN behaves like ROUND (§18.17.7.278), except that ROUNDDOWN always rounds a number down. *end note*]

Arguments:

Name	Type	Description
x	number	The value to be rounded down.
number-digits	number	The number of digits to which *x* is to be rounded. If *number-digits* is greater than 0, *x* is rounded to the specified number of decimal places. If *number-digits* is 0, *x* is rounded to the nearest integer. If *number-digits* is less than 0, *x* is rounded to the left of the decimal point.

Return Type and Value: number – The rounded-down value of *x*.

[*Example*:

`ROUNDDOWN(3.2,0)` rounds 3.2 down to zero decimal places; that is, to 3
`ROUNDDOWN(76.9,0)` rounds 76.9 down to zero decimal places; that is, to 76
`ROUNDDOWN(3.14159,3)` rounds 3.14159 down to three decimal places; that is, to `3.141`
`ROUNDDOWN(-3.14159,1)` rounds -3.14159 down to one decimal place; that is, to `-3.1`
`ROUNDDOWN(31415.92654,-2)` rounds 31415.92654 down to two decimal places to the left of the decimal; that is, to `31400`

end example]

18.17.7.280 ROUNDUP

Syntax:

`ROUNDUP ( x , number-digits )`

Description: Computes *x* rounded up, away from zero, to the number of digits specified by *number-digits*. [*Note*: ROUNDUP behaves like ROUND (§18.17.7.278), except that ROUNDUP always rounds a number up. *end note*]

Arguments:

Name	Type	Description
x	number	The value to be rounded up.
number-digits	number	The number of digits to which *x* is to be rounded. If *number-digits* is greater than 0, *x* is rounded up to the specified number of decimal places. If *number-digits* is 0, *x* is rounded up to the nearest integer. If *number-digits* is less than 0, *x* is rounded up to the left of the decimal point.

Return Type and Value: number – The rounded-up value of *x*.

[*Example*:

`ROUNDDOWN(3.2,0)` rounds 3.2 down to zero decimal places; that is, to 4
`ROUNDDOWN(76.9,0)` rounds 76.9 down to zero decimal places; that is, to 77
`ROUNDDOWN(3.14159,3)` rounds 3.14159 down to three decimal places; that is, to `3.142`
`ROUNDDOWN(-3.14159,1)` rounds -3.14159 down to one decimal place; that is, to `-3.2`
`ROUNDDOWN(31415.92654,-2)` rounds 31415.92654 down to two decimal places to the left of the decimal; that is, to `31500`

end example]

18.17.7.281 ROW

Syntax:

`ROW ( [ reference ] )`

Description: Finds the number of the row(s) corresponding to *reference*.

Arguments:

Name	Type	Description
reference	reference to a single cell or to a range of	If omitted, the behavior is as if *reference* referred to the cell containing the formula.

Name	Type	Description
	contiguous cells	

Return Type and Value: number – If *reference* refers to a single cell or to a single row of cells, the corresponding row is returned. If *reference* refers to a range of cells involving multiple rows, a vertical array of the corresponding rows as numbers is returned.

However, if the range of cells referred to by *reference* is not contiguous, `#REF!` is returned.

[*Example*:

`ROW()` results in 16, when the cell containing the formula is in row 16
`ROW(E17:G17)` results in 17
`ROW(E16:G17)` results in a vertical array containing 16 and 17, respectively

end example]

18.17.7.282 ROWS

Syntax:

`ROWS ( array )`

Description: Finds the number of rows corresponding to *array*.

Arguments:

Name	Type	Description
array	array, reference to a single cell, or a reference to a range of contiguous cells	A set of rows.

Return Type and Value: number – The number of rows corresponding to *array*.

However, if the range of cells referred to by *array* is not contiguous, `#NULL!` is returned.

[*Example*:

`ROWS(E16:H16)` results in 1
`ROWS(E16:G18)` results in 3
`ROWS({1,2;3,4})` results in 2

end example]

18.17.7.283 RSQ

Syntax:

`RSQ ( known-ys , known-xs )`

Description: Computes the square of the Pearson product moment correlation coefficient through data points in known ys and known xs.

Mathematical Formula:

The equation for the Pearson product moment correlation coefficient, r, is:

$$r = \frac{\sum(x-\bar{x})(y-\bar{y})}{\sqrt{\sum(x-\bar{x})^2\sum(y-\bar{y})^2}}$$

where:

- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*known-xs*)
- y = a sample value
- $\bar{y}$ = the sample mean AVERAGE(*known-ys*)

Arguments:

Name	Type	Description
known-xs	number, name, array, or reference to number, text, logical	Designate a set of numeric data points. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
known-ys	number, name, array, or reference to number, text, logical	Designate a set of numeric data points. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The square of the Pearson product moment correlation coefficient.

However, if

- *known-ys* and *known-xs* are empty or have a different number of data points, the return value is unspecified.
- *known-ys* and *known-xs* contain only one data point, the return value is unspecified.

[*Example*:

`RSQ({2,3,9,1,8,7,5},{6,5,11,7,5,4,4})` results in `0.057950192`

end example]

18.17.7.284 RTD

Syntax:

RTD (*progID* , [*rtd-server*] , *argument-list*)

Description: Retrieves data from a program in real-time. Periodically, this function returns new values and causes recalculation of the expression containing the call to it.

Arguments:

Name	Type	Description
progID	text	The name of the program from which the data is to be retrieved.
rtd-server	text	An optional string that is specific to the program with which RTD is communicating.
argument list	any	The presence and meaning of each *argument* in *argument-list* is specific to the program with which RTD is communicating.

Return Type and Value: array – The set of values returned by the program with which RTD is communicating.

[*Example*: Consider a stockprice program that is called as follows:

```
RTD("stockprice.rtd","NASD","MSFT")
```

The result it returns—the price of the stock MSFT according to NASD—changes over time, often every few seconds.

The *rtd-server* program could also be written to accept multiple arguments, allowing calls like the following::

```
RTD("stockprice.rtd","NASD","MSFT","GOOG","AMZN")
```


where three stock values are requested.

end example]

18.17.7.285 SEARCH

Syntax:

SEARCH (*string-1* , *string-2* [, *start-pos*])

Description: Performs a case-insensitive search, using a lexical comparison, for the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. (SEARCH is intended for use with languages that use the single-byte character set (SBCS), whereas SEARCHB (§18.17.7.286) is intended for use with languages that use the double-byte character set (DBCS).)

Arguments:

Name	Type	Description
string-1	text	Designate the string to be searched for within the string designated by *string-2*. *string-1* can contain the following wildcard characters: question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for an actual question mark or asterisk, that character shall be preceded by a tilde (~).
string-2	text	
start-pos	number	The number of the start position within *string-2* for which *string-1* is to be searched. The start position of the first character is 1. If omitted, a position of 1 shall be assumed. *start-pos* shall be at least 0.

Return Type and Value: number – The start position of the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. If *string-1* is an empty string, it shall always be found in any *string-2* at position *start-pos*, or at position 1 if *start-pos* is omitted.

However, if

- *string-1* is not found within *string-2*, #VALUE! is returned.
- *start-pos* designates a position outside *string-2*, #VALUE! is returned.

[*Example*:

`SEARCH("de","abcdEF")` results in 4
`SEARCH("?c*e","abcdEF")` results in 2

end example]

18.17.7.286 SEARCHB

Syntax:

`SEARCHB ( string-1 , string-2 [ , start-pos ] )`

Description: Performs a case-insensitive search, using a lexical comparison, for the first occurrence of *string-1* in *string-2*, starting at byte position *start-pos* within *string-2*. (SEARCHB is intended for use with languages that use the double-byte character set (DBCS), whereas SEARCH (§18.17.7.285) is intended for use with languages that use the single-byte character set (SBCS).)

Arguments:

Name	Type	Description
string-1	text	Designate the string to be searched for within the string designated by *string-2*. *string-1* can contain the following wildcard characters: question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for an actual question mark or asterisk, that character shall be preceded by a tilde (~).
string-2	text	
start-pos	number	The number of the start position within *string-2* for which *string-1* is to be searched. The start position of the first byte is 1. If omitted, a position of 1 shall be assumed. *start-pos* shall be at least 0.

Return Type and Value: number – The start position of the first occurrence of *string-1* in *string-2*, starting at character position *start-pos* within *string-2*. If *string-1* is an empty string, it shall always be found in any *string-2* at position *start-pos*, or at position 1 if *start-pos* is omitted.

However, if

- *string-1* is not found within *string-2*, `#VALUE!` is returned.
- *start-pos* designates a position outside *string-2*, `#VALUE!` is returned.

[*Example*: Assuming 1-byte characters

`SEARCHB("de","abcdEF")` results in 4
`SEARCHB("?c*e","abcdEF")` results in 2

end example]

18.17.7.287 SECOND

Syntax:

`SECOND ( time-value )`

Description: Computes the second for the date and/or time having the given *time-value*.

Arguments:

Name	Type	Description
time-value	number	The date and/or time whose second is to be computed. That date and/or time shall be expressed either as a serial date-time, in which case, its integer part is ignored, or as a *string-constant* having any date and/or time format, in which case, any date information shall be ignored.

Return Type and Value: number – The second for the date and/or time having the given *time-value*.

However, if *time-value* is out of range for the current date system, `#NUM!` is returned.

[*Example*:

`SECOND(DATE(2006,2,26)+TIME(2,10,20))` results in `20`
`SECOND(TIME(22,56,34))` results in 34
`SECOND(0)` results in `0`, since serial date-time 0 represents `00:00:00`
`SECOND(10.5)` results in `0`, since serial date-time .5 represents `12:00:00`
`SECOND("22-Oct-2001 10:53:12")` results in 12
`SECOND("10:53:12 pm")` results in 12
`SECOND("22:53:12")` results in 12

end example]

18.17.7.288 SERIESSUM

Syntax:

`SERIESSUM (` *input-value* `,` *initial-power* `,` *step* `,` *coefficients* `)`

Description: Computes the sum of a power series.

Mathematical Formula:

The sum of a power series is based on the formula:

$$SERIES(x,n,m,a) = a_1x^n + a_2x^{(n+m)} + a_3x^{(n+2m)} + \cdots + a_ix^{(n+(i-1)m)}$$

where:

- a = argument *coefficients*
- m = argument *step*

- *n* = argument *initial-power*
- *x* = argument *input-value*

Arguments:

Name	Type	Description
input-value	number	The input value to the power series;
initial-power	number	The initial power to which *input-value* is to be raised.
step	number	The step by which to increase *initial-power* for each term in the series;
coefficients	reference	A set of coefficients by which each successive power of *input-value* is multiplied. The number of values in *coefficients* determines the number of terms in the power series.

Return Type and Value: number – The sum of a power series.

[*Example*: Given the following data:

	A
1	1
2	=-1/FACT(2)
3	=1/FACT(4)
4	=-1/FACT(6)

`SERIESSUM(PI()/4,0,2,A1:A4)` results in `0.707103`, an approximation to the cosine of π/4 radians

end example]

18.17.7.289 SIGN

Syntax:

`SIGN ( x )`

Description: Determines the sign of *x*.

Arguments:

Name	Type	Description
x	number	The number whose sign is to be determined.

Return Type and Value: number – 1 if *x* is positive, 0 if *x* is 0, and -1 if *x* is negative.

[*Example*:

`SIGN(10.5)` results in `1`
`SIGN(0)` results in `0`
`SIGN(-5.4)` results in `-1`

end example]

18.17.7.290 SIN

Syntax:

```
SIN ( x )
```

Description: Computes the sine of x.

Arguments:

Name	Type	Description
x	number	The number, in radians, whose sine is to be computed.

Return Type and Value: number – The sine of x.

[*Example*:

`SIN(-1)` results in `-0.841470985`
`SIN(0)` results in `0`
`SIN(1)` results in `0.841470985`

end example]

18.17.7.291 SINH

Syntax:

```
SINH ( x )
```

Description: Computes the hyperbolic sine of x.

Arguments:

Name	Type	Description
x	number	The number whose hyperbolic sine is to be computed.

Return Type and Value: number – The hyperbolic sine of x.

However, if the magnitude of x is too large for the result to be represented, `#NUM!` is returned.

[*Example*:|

`SINH(1)` results in `1.175201194`
`SINH(10)` results in `11013.23287`
`SINH(100)` results in `1.34406E+43`

end example]

18.17.7.292 SKEW

Syntax:

`SKEW (` *argument-list* `)`

Description: Computes the skewness of a distribution. [*Note*: Skewness characterizes the degree of asymmetry of a distribution around its mean. Positive skewness indicates a distribution with an asymmetric tail extending toward more positive values. Negative skewness indicates a distribution with an asymmetric tail extending toward more negative values. *end note*]

Mathematical Formula:

$$\frac{n}{(n-1)(n-2)}\sum\left(\frac{x_i-\bar{x}}{s}\right)^3$$

where:

- n = the number of elements in *argument-list*
- s = standard deviation of the values in *argument-list*
- xi = value of each element in *argument-list*
- $\bar{x}$ = mean of the values in *argument-list*

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *arguments* in *argument-list* designate the numbers for which the skewness is to be computed. Logical values and text representations of numbers that are entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The skewness of a distribution.

However, if

- There are fewer than three data points, the return value is unspecified.
- The sample standard deviation is zero, the return value is unspecified.

[*Example*:

`SKEW(3,4,5,2,3,4,5,6,4,7)` results in `0.359543071`

end example]

18.17.7.293 SLN

`SLN (` *cost* `,` *salvage* `,` *life* `)`

Description: Computes the straight-line depreciation of an asset for one period.

Arguments:

Name	Type	Description
cost	number	The number *cost* is the initial cost of the asset.
salvage	number	The value at the end of the depreciation. (This is sometimes called the salvage value of the asset.)
life	number	The number of periods over which the asset is being depreciated. (This is sometimes called the useful life of the asset.)

Return Type and Value: number – The straight-line depreciation of an asset for one period.

[*Example*:

`SLN(30000,7500,10)` results in `2,250.00`

end example]

18.17.7.294 SLOPE

Syntax:

`SLOPE (` *known-ys* `,` *known-xs* `)`

Description: Computes the slope of the linear regression line through data points in known ys and known xs. The slope is the vertical distance divided by the horizontal distance between any two points on the line, which is the rate of change along the regression line.

Mathematical Formula:

$$b = \frac{\sum(x - \bar{x})(y - \bar{y})}{\sum(x - \bar{x})^2}$$

where:

- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE($known\text{-}xs$)
- y = a sample value
- $\bar{y}$ = the sample mean AVERAGE($known\text{-}ys$)

Arguments:

Name	Type	Description
known-xs	number, name, array, or reference to number, text, logical	Designate a set of numeric dependent data points. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
known-ys	number, name, array, or reference to number, text, logical	Designate a set of numeric independent data points. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The slope of the linear regression line through data points in known ys and known xs.

However, if

- *known-ys* and *known-xs* are empty, the return value is unspecified.
- *known-ys* and *known-xs* have a different number of data points, the return value is unspecified.

[*Example*:

`SLOPE({2,3,9,1,8,7,5},{6,5,11,7,5,4,4})` results in `0.305555556`

end example]

18.17.7.295 SMALL

Syntax:

```
SMALL ( array , k )
```


Description: Computes the k^{th} smallest value in a data set.

Arguments:

Name	Type	Description
array	array, reference	The set of numbers from which the k^{th}-smallest value is to be determined.
k	number	The position (from the smallest) in the array or cell range of data to return.

Return Type and Value: number – The k^{th} smallest value in a data set.

However, if

- *array* is empty, the return value is unspecified.
- $k \le 0$ or k is greater than the number of data points, `#NUM!` is returned.

[*Example*:

`SMALL({3,5,3,5,4;4,2,4,6,7},3)` results in `3`
`SMALL({3,5,3,5,4;4,2,4,6,7},7)` results in `5`

end example]

18.17.7.296 SQRT

Syntax:

`SQRT ( x )`

Description: Computes the positive square root of x.

Arguments:

Name	Type	Description
x	number	The number whose positive root is to be found.

Return Type and Value: number – The positive square root of x.

However, if x is negative, `#NUM!` is returned.

[*Example*:

`SQRT(2)` results in `1.414213562`
`SQRT(5)` results in `2.236067977`

end example]

18.17.7.297 SQRTPI

Syntax:

`SQRTPI ( x )`

Description: Computes the positive square root of $x \times \pi$.

Arguments:

Name	Type	Description
x	number	The number, which when multiplied by π, whose positive root is to be found.

Return Type and Value: number – The positive square root of $x \times \pi$.

However, if x is negative, `#NUM!` is returned.

[*Example*:

`SQRTPI(1)` results in `1.772453851`
`SQRTPI(2)` results in `2.506628275`

end example]

18.17.7.298 STANDARDIZE

Syntax:

`STANDARDIZE ( x , mean , standard-dev )`

Description: Computes a normalized value from a distribution characterized by *mean* and *standard-dev*.

Mathematical Formula:

$$Z = \frac{X - \mu}{\sigma}$$

where:

- X = argument x
- μ = argument *mean*
- σ = argument *standard-dev*

Arguments:

Name	Type	Description
x	number	The number whose value is to be normalized. Represented by X in the mathematical formula presented.
mean	number	The arithmetic mean of the distribution. Represented by mu (µ) in the mathematical formula presented.
standard-dev	number	The standard deviation of the distribution. Represented by sigma (σ) in the mathematical formula presented.

Return Type and Value: number – A normalized value from a distribution.

However, if *standard-dev* ≤ 0, `#NUM!` is returned.

[*Example*:

`STANDARDIZE(42,40,1.5)` results in `1.333333333`

end example]

18.17.7.299 STDEV

Syntax:

STDEV (*argument-list*)

Description: Makes an estimate of the standard deviation based on a sample, using the "unbiased" or "n-1" method. [*Note*: STDEV assumes that its arguments are a sample of the population. If the data represents the entire population, STDEVP should be used instead. If logical values and text representations of numbers in a reference are to be included as part of the calculation, use STDEVA instead. *end note*]

Mathematical Formula:

$$\sqrt{\frac{\sum(x-\bar{x})^2}{(n-1)}}$$

where:

- n = the sample size
- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*argument-1*, *argument-2*, …, *argument-n*)

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *argument*s in *argument-list* designate the numbers that are samples of the population. *argument-list* can also be an array of numbers. Logical values and text representations of numbers that are entered directly into the list of arguments are included. If an argument is an array or reference, only numbers in that array or reference are included. Empty cells, logical values, text, or error values in the array or reference are ignored.

Return Type and Value: number – An estimate of the standard deviation based on a sample.

[*Example*:

`STDEV(123,134,143,173,112,109)` results in `23.72902583`

end example]

18.17.7.300 STDEVA

Syntax:

STDEVA (*argument-list*)

Description: Makes an estimate of the standard deviation based on a sample, using the "unbiased" or "n-1" method. [*Note*: STDEVA assumes that its arguments are a sample of the population. If the data represents the entire population, STDEVPA should be used instead. If logical values and text representations of numbers in a reference are to be excluded as part of the calculation, use STDEV instead. *end note*]

Mathematical Formula:

$$\sqrt{\frac{\sum(x-\bar{x})^2}{(n-1)}}$$

where:

- n = the sample size
- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*argument-1*, *argument-2*, …, *argument-n*)

Arguments:

Name	Type	Description
argument-list	logical, number,	The *argument*s in *argument-list* designate the numbers

Name	Type	Description
	name, text, array, reference. The argument list can also be an array of numbers.	that are samples of the population. Arguments that contain TRUE evaluate as 1; arguments that contain text or FALSE evaluate as zero. If an argument is an array or reference, only values in that array or reference are used. Empty cells and text values in the array or reference are ignored.

Return Type and Value: number – An estimate of the standard deviation based on a sample.

[*Example*:

`STDEVA(123,134,143,173,112,109)` results in `23.72902583`

end example]

18.17.7.301 STDEVP

Syntax:

`STDEVP ( `*argument-list*` )`

Description: Computes the standard deviation of an entire population, using the "biased" or "n" method. [*Note*: STDEVP assumes that its arguments are the total population. If the data represents a population sample only, STDEVA should be used instead. If logical values and text representations of numbers in a reference are to be included as part of the calculation, use STDEVPA instead. *end note*]

Mathematical Formula:

$$\sqrt{\frac{\sum(x-\bar{x})^2}{n}}$$

where:

- n = the sample size
- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*argument-1*, *argument-2*, …, *argument-n*)

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference.	The *arguments* in *argument-list* designate the numbers that are the members of the population. Logical values, and text representations of numbers that are entered

Name	Type	Description
	The argument list can also be an array of numbers.	directly into the list of arguments are included. If an argument is an array or reference, only numbers in that array or reference are included. Empty cells, logical values, text, or error values in the array or reference are ignored.

Return Type and Value: number – The standard deviation of an entire population.

[*Example*:

`STDEVP(123,134,143,173,112,109)` results in `21.66153785`

end example]

18.17.7.302 STDEVPA

Syntax:

STDEVPA (*argument-list*)

Description: Computes the standard deviation of an entire population, using the "biased" or "n" method. [*Note*: STDEVPA assumes that its arguments are the total population. If the data represents a population sample only, STDEVA should be used instead. If logical values and text representations of numbers in a reference are to be excluded as part of the calculation, use STDEVP instead. *end note*]

Mathematical Formula:

$$\sqrt{\frac{\sum(x-\bar{x})^2}{n}}$$

where:

- n = the sample size
- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*argument-1*, *argument-2*, …, *argument-n*)

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference. The argument list can also be an	The *arguments* in *argument-list* designate the numbers that are the members of the population.

Name	Type	Description
	array of numbers.	

Return Type and Value: number – The standard deviation of an entire population.

Arguments can be numbers; names, arrays, or references that contain numbers; text representations of numbers; or logical values, in a reference. Text representations of numbers that are entered directly into the list of arguments are included. Arguments that contain TRUE evaluate as 1; arguments that contain text or FALSE evaluate as zero. If an argument is an array or reference, only values in that array or reference are used. Empty cells and text values in the array or reference are ignored.

[*Example*:

`STDEVPA(123,134,143,173,112,109)` results in `21.66153785`

end example]

18.17.7.303 STEYX

Syntax:

STEYX (*known-ys* , *known-xs*)

Description: Computes the standard error of the predicted y-value for each x in the regression. The standard error is a measure of the amount of error in the prediction of y for an individual x.

Mathematical Formula:

$$\sqrt{\frac{1}{(n-2)}\left[\sum(y-\bar{y})^2 - \frac{[\sum(x-\bar{x})(y-\bar{y})]^2}{\sum(x-\bar{x})^2}\right]}$$

where:

- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*known-xs*)
- y = a sample value
- $\bar{y}$ = the sample mean AVERAGE(*known-ys*)

Arguments:

Name	Type	Description
known-xs	number, name, array, or reference to	Designate a set of numeric dependent data points. Logical values and text representations of numbers entered directly into the list of arguments are included. If

Name	Type	Description
	number, text, logical	an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.
known-ys	number, name, array, or reference to number, text, logical	Designate a set of numeric independent data points. Logical values and text representations of numbers entered directly into the list of arguments are included. If an array or reference argument contains text, logical values, or empty cells, those values are ignored; however, cells with the value 0 are included.

Return Type and Value: number – The standard error of the predicted y-value for each *x* in the regression.

However, if

- *known-ys* and *known-xs* have a different number of data points, the return value is unspecified.
- *known-ys* and *known-xs* are empty or have less than three data points, the return value is unspecified.

[*Example*:

`STEYX({2,3,9,1,8,7,5},{6,5,11,7,5,4,4})` results in `3.30571895`

end example]

18.17.7.304 SUBSTITUTE

Syntax:

`SUBSTITUTE (` *string* `,` *old-string* `,` *new-string* `[ ,` *occurence* `] )`

Description: Produces a new string that is *string* with one or all occurrences of *old-string* replaced by *new-string*.

Arguments:

Name	Type	Description
string	text	Designates a string.
old-string	text	Designates a string.
new string	text	Designates a string.
occurence	number	The occurrence number of the *old-string* characters within *string-1* that is to be replaced by the string designated by *new-string*. If omitted, all occurrences of *old-string* characters shall be replaced.

Return Type and Value: text – A string that is *string* with one or all occurrences of *old-string* replaced by *new-string*.

However, if *occurance* < 0, #VALUE! is returned.

[*Example*:

`SUBSTITUTE("abcaaabca","a","xx")` results in xxbcxxxxxxbcxx
`SUBSTITUTE("abcaaabca","a","",10)` results in bcbc
`SUBSTITUTE("abcaaabca","a","xx",3)` results in abcaxxabca

end example]

18.17.7.305 SUBTOTAL

Syntax:

`SUBTOTAL ( function-number , argument-list )`

Description: Computes a value using the function designated by *function-number*, using the *arguments* in *argument-list*.

Arguments:

Name	Type	Description
function-number	number	Indicates the function to be called, as shown in the table below.
argument-list	number	Each *argument* in *argument-list* is passed to the called function, in the order specified. That shall be no more than 254 *arguments*.

function-number (includes hidden values)	*function-number* (excludes hidden values)	Function
1	101	AVERAGE
2	102	COUNT
3	103	COUNTA
4	104	MAX
5	105	MIN
6	106	PRODUCT
7	107	STDEV
8	108	STDEVP
9	109	SUM

function-number (includes hidden values)	*function-number* (excludes hidden values)	Function
10	110	VAR
11	111	VARP

If any argument contains a SUBTOTAL function call, that call shall be ignored to avoid double counting.

For the *function-number* values 1–11, the values of hidden rows are included. For the *function-number* values 101–111, the values of hidden rows are excluded.

The SUBTOTAL function shall ignore any rows that are not included in the result of a filter, regardless of which *function-number* value is used.

The SUBTOTAL function is designed for columns of data, or vertical ranges. It is not designed for rows of data, or horizontal ranges. [*Example*: When a horizontal range is subtotaled using a *function-number* of 101 or greater, hiding a column does not affect the subtotal. However, hiding a row in a subtotal of a vertical range does affect the subtotal. *end example*]

Return Type and Value: number – The result from calling the function designated by *function-number*, using the *argument*s in *argument-list*.

However, if *function-number* does not have one of the values specified above, `#NUM!` is returned.

[*Example*:

`SUBTOTAL(2,E5:E15)` counts the number of values in the cell range E5:E15, including hidden values
`SUBTOTAL(4,E5:E15)` finds the maximum value of the values in the cell range E5: E15, including hidden values
`SUBTOTAL(106,E5:E15)` finds the product of the values in the cell range E5: E15, excluding hidden values

end example]

18.17.7.306 SUM

Syntax:

`SUM (` *argument-list* `)`

Description: Adds the numeric values of *argument*s in *argument-list*.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference.	The *argument*s in *argument-list* designate the numeric values to be added. Arguments that are numbers, logical values, or text representations of numbers shall be

Name	Type	Description
		counted. If an argument is an array or reference, only numbers in that array or reference shall be counted. Empty cells, logical values, and text in the array or reference shall be ignored.

Return Type and Value: number – The sum of the values of its arguments.

[*Example*:

`SUM(1,2,3,4,5)` results in `15`
`SUM({1,2;3,4})` results in `10`
`SUM({1,2,3,4,5},6,"7")` results in 28
`SUM({1,"2",TRUE,4})` results in 5, as the logical value and numeric text are ignored

end example]

18.17.7.307 SUMIF

Syntax:

`SUMIF (` *cell-range* `,` *selection-criteria* [`,` *sum-range*] `)`

Description: Applies selection criteria on the values in one range of cells and sums the values of the cells in a corresponding range.

Arguments:

Name	Type	Description
cell-range	reference	Designates the range of cells to be inspected.
selection-criteria	number, expression, reference, text	Defines which cells are counted. In the case of text, *selection-criteria* can consist of any comparison operator followed by the operand against which each cell's value is to be compared. *selection-criteria* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~).
sum-range	reference	If present, *sum-range* designates the cells whose values are summed. In this case, *sum-range* does not have to have the same size and shape as *cell-range*. The actual cells that are added are determined by using the top, left cell in *sum-range* as the beginning cell, and then including cells that correspond in size and shape to *cell-range*. If

Name	Type	Description
		omitted, *cell-range* also designates the cells whose values are summed.

Return Type and Value: number – The sum of the cells corresponding to those selected.

[*Example*: Given that A1, B1, C1, and D1, respectively, contain the values 3, 10, 7, and 10

`SUMIF(A1:D1,"=10")` results in `20`
`SUMIF(A1:D1,">5")` results in 27
`SUMIF(A1:D1,"<>10")` results in `10`

Given that A2, B2, C2, and D2, respectively, contain the values `apples`, `melons`, `10`, and 15
`SUMIF(A2:B2,"*es",C2:D2)` results in `10`

end example]

18.17.7.308 SUMIFS

Syntax:

> `SUMIFS (` *sum-range* `,` *cell-range-1* `,` *selection-criteria-1*
> `[ ,` *cell-range-2* `,` *selection-criteria-2* `[ , … ] ] )`

Description: Adds the cells in a range that meet multiple criteria.

Arguments:

Name	Type	Description
sum-range	reference	Designates the cells whose values are summed. In this case, *sum-range* does not have to have the same size and shape as *cell-range-1* through *cell-range-n*. The actual cells that are added are determined by using the top, left cell in *sum-range* as the beginning cell, and then including cells that correspond in size and shape to *cell-range-1* through *cell-range-n*. Each cell in *sum-range* is summed only if all of the corresponding criteria specified are true for that cell. Cells in *sum-range* that contain TRUE evaluate to 1; cells in sum-range that contain FALSE evaluate to 0.
cell-range-1	reference	Designates the first range of cells to be inspected.
selection-criteria-1	number, expression, reference, text	Specifies the criteria for the first range of cells that is counted. In the case of text, *selection-criteria-1* can consist of any comparison operator followed by the operand against which each cell's value is to be

Name	Type	Description
		compared. *selection-criteria-1* can include one or more wildcard characters, question mark (?) and asterisk (*). A question mark matches any single character; an asterisk matches any sequence of characters. To search for a question mark, asterisk, or tilde character, prefix that character with a tilde (~).
cell-range-n	reference	The optional arguments *selection-criteria-2* through *selection-criteria-n* have corresponding arguments *cell-range-2* through *cell-range-n*, and have the same semantics as *selection-criteria-1* and *cell-range-1*, respectively.
selection-criteria-n	number, expression, reference, text	

Return Type and Value: number – The sum of the cells corresponding to those selected.

[*Example*: Given the following data:

	A	B	C	D
1	Sales Person	Tables	Chairs	Desks
2	Emilio	34	85	97
3	Julie	353	23	18
4	Hans	13	67	14
5	Frederique	0	98	0

`SUMIFS(B2:C5,A2:A5,"=Julie")` results in 353 (the sum of the number of tables and chairs sold by Julie)

`SUMIFS(B2:B5,A2:A5,"=Julie",A2:A5,"=Hans")` results in `0` (the sum of the number of tables sold by Julie and Hans)

`SUMIFS(B2:B5,A3,"=Julie",A4,"=Hans")` results in 34 (the sum of the the number of tables sold by Julie and Hans)

`SUMIFS(B2:D5,A2:A5,"<>Emilio")` results in 768 (the sum of the number of tables, chairs, and desks sold by all sales persons except Emilio)

end example]

18.17.7.309 SUMPRODUCT

Syntax:

`SUMPRODUCT ( `*argument-list*` )`

Description: Multiplies the corresponding elements in the array *arguments* in *argument-list*, and returns the sum of those products. An array element that is not numeric is treated as if it contained 0.

Arguments:

Name	Type	Description
argument-list	array of numbers	The *arguments* in *argument-list* designate the numeric values to be multiplied.

Return Type and Value: number – The sum of the products of the corresponding elements in the *arguments* in *argument-list*.

However, if the array arguments do not have the same dimensions, `#NUM!` is returned.

[*Example*:

`SUMPRODUCT({2,3})` results in 5
`SUMPRODUCT({2,3},{4,5})` results in 23
`SUMPRODUCT({2,3},{4,5},{2,2})` results in 46
`SUMPRODUCT({2,3;4,5},{2,2;3,4})` results in 42

end example]

18.17.7.310 SUMSQ

Syntax:

`SUMSQ ( ` *argument-list* ` )`

Description: Adds the squares of *arguments* in *argument-list*.

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *arguments* in *argument-list* designate the values whose squares are to be summed. Arguments that are numbers, logical values, or text representations of numbers shall be counted. If an argument is an array or reference, only numbers in that array or reference shall be counted. Empty cells, logical values, and text in the array or reference shall be ignored.

Return Type and Value: number – The sum of the squares of its arguments.

[*Example*:

`SUMSQ(2)` results in 4
`SUMSQ(2.5,-3.6))` results in `19.21`
`SUMSQ({2.5,-3.6)},2.4)` results in `24.97`

end example]

18.17.7.311 SUMX2MY2

Syntax:

`SUMX2MY2 (` *array-1* `,` *array-2* `)`

Description: Computes the sum of the difference of squares of the corresponding numerical elements in two arrays designated by *array-1* and *array-2*.

Mathematical Formula:

$$SUMX2MY2 = \sum (x^2 - y^2)$$

where:

- *array-1* contains the x values
- *array-2* contains the y values

Arguments:

Name	Type	Description
array-1	array, reference	Designated the arrays to be operated on. If an argument contains text, logical values, or empty cells, those elements shall be ignored; however, cells with the value 0 shall be included.
array-2		

Return Type and Value: number – The sum of the difference of squares of the corresponding elements in two arrays designated by *array-1* and *array-2*.

However, if *array-1* and *array-2* have a different number of values, the return value is unspecified.

[*Example*:

`SUMX2MY2({2,3,9,1,8,7,5},{6,5,11,7,5,4,4})` results in 55
`SUMX2MY2({2,3,9;1,8,7},{6,5,11;7,5,4})` results in -64
end example]

18.17.7.312 SUMX2PY2

Syntax:

`SUMX2PY2 ( array-1 , array-2 )`

Description: Computes the sum of the sum of the squares of the corresponding numerical elements in two arrays designated by *array-1* and *array-2*.

Mathematical Formula:

$$SUMX2PY2 = \sum (x^2 + y^2)$$

where:

- *array-1* contains the x values
- *array-2* contains the y values

Arguments:

Name	Type	Description
array-1	array, reference	Designated the arrays to be operated on. If an argument contains text, logical values, or empty cells, those elements shall be ignored; however, cells with the value 0 shall be included.
array-2		

Return Type and Value: number – The sum of the sum of the squares of the corresponding elements in two arrays designated by *array-1* and *array-2*.

However, if *array-1* and *array-2* have a different number of values, the return value is unspecified.

[*Example*:

`SUMX2PY2({2,3,9,1,8,7,5},{6,5,11,7,5,4,4})` results in 521
`SUMX2PY2({2,3,9;1,8,7},{6,5,11;7,5,4})` results in 480

end example]

18.17.7.313 SUMXMY2

Syntax:

`SUMXMY2 ( array-1 , array-2 )`

Description: Computes the sum of the squares of the difference between corresponding numerical elements in two arrays designated by *array-1* and *array-2*.

Mathematical Formula:

$$SUMXMY2 = \sum (x - y)^2$$

where:

- *array-1* contains the x values
- *array-2* contains the y values

Arguments:

Name	Type	Description
array-1	array, reference	Designated the arrays to be operated on. If an argument contains text, logical values, or empty cells, those elements shall be ignored; however, cells with the value 0 shall be included.
array-2		

Return Type and Value: number – The sum of the squares of the difference between the corresponding elements in two arrays designated by *array-1* and *array-2*.

However, if *array-1* and *array-2* have a different number of values, the return value is unspecified.

[*Example*:

`SUMXMY2({2,3,9,1,8,7,5},{6,5,11,7,5,4,4})` results in 79
`SUMXMY2({2,3,9;1,8,7},{6,5,11;7,5,4})` results in 78

end example]

18.17.7.314 SYD

SYD (*cost* , *salvage* , *life* , *per*)

Description: Computes the sum-of-years' digits depreciation of an asset for a specified period.

Mathematical Formula:

$$SYD = \frac{(cost - salvage) * (life - per + 1) * 2}{(life)(life + 1)}$$

where:

- *cost* = argument *cost*
- *life* = argument *life*
- *per* = argument *per*
- *salvage* = argument *salvage*

Arguments:

Name	Type	Description
cost	number	The initial cost of the asset.
salvage	number	The value at the end of the depreciation. (This is sometimes called the salvage value of the asset.)
life	number	The number of periods over which the asset is being depreciated. (This is sometimes called the useful life of the asset.)
per	number	The period and shall have the same units as *life*.

Return Type and Value: number – The sum-of-years' digits depreciation of an asset for a specified period.

[*Example*:

`SYD(30000,7500,10,1)` results in `4,090.91`
`SYD(30000,7500,10,10)` results in `409.09`

end example]

18.17.7.315 T

Syntax:

`T ( value )`

Description: Retrieves the text referencedby *value*.

Arguments:

Name	Type	Description
value	any	The value to be tested for text. No conversion to text shall take place on an argument passed to this function.

Return Type and Value: text – *value* if *value* designates text; otherwise, `""`. [*Note*: T cannot differentiate between text that is an empty string, and any *value* of non-text type. *end note*]

[*Example*:

`T("Hello")` results in `Hello`
`T(123)` results in an empty string
`LEN(T(123))` results in `0`

end example]

18.17.7.316 TAN

Syntax:

```
TAN ( x )
```

Description: Computes the tangent of x.

Arguments:

Name	Type	Description
x	number	The number, in radians, whose tangent is to be computed.

Return Type and Value: number – The tangent of x.

[*Example*:

`TAN(-1)` results in `-1.557407725`
`TAN(0)` results in `0`
`TAN(1)` results in `1.557407725`

end example]

18.17.7.317 TANH

Syntax:

```
TANH ( x )
```

Description: Computes the hyperbolic tangent of x.

Arguments:

Name	Type	Description
x	number	The number whose hyperbolic tangent is to be computed.

Return Type and Value: number – The hyperbolic tangent of x

[*Example*:

`TANH(-1)` results in `-0.761594156`

`TANH(0)` results in `0`
`TANH(1)` results in `0.761594156`

end example]

18.17.7.318 TBILLEQ

`TBILLEQ (` *settlement* `,` *maturity* `,` *discount* `)`

Description: Computes the bond-equivalent yield for a U.S. Treasury bill.

Arguments:

Name	Type	Description
settlement	number	The Treasury bill's settlement date. Any time information in the date is ignored.
maturity	number	The Treasury bill's maturity date. Any time information in the date is ignored.
discount	number	The Treasury bill's discount rate.

Return Type and Value: number – The bond-equivalent yield for a U.S. Treasury bill.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* > *maturity*, `#NUM!` is returned.
- *maturity* is more than one year after *settlement*, `#NUM!` is returned.
- *discount* ≤ 0, `#NUM!` is returned.

[*Example*:

`TBILLEQ(DATE(2008,3,31),DATE(2008,6,1),0.0914)` results in `9.4151%`

end example]

18.17.7.319 TBILLPRICE

`TBILLPRICE (` *settlement* `,` *maturity* `,` *discount* `)`

Description: Computes the price per $100 face value for a U.S. Treasury bill.

Mathematical Formula:

$$TBILLPRCE = 100 \times \left(1 - \frac{discunt \times DSM}{360}\right)$$

where:

- $discount$ = argument $discount$
- DSM = number of days from settlement to maturity, excluding any maturity date that is more than one calendar year after the settlement date.

Arguments:

Name	Type	Description
settlement	number	The Treasury bill's settlement date. Any time information in the date is ignored.
maturity	number	The Treasury bill's maturity date. Any time information in the date is ignored.
discount	number	The Treasury bill's discount rate.

Return Type and Value: number – The price per $100 face value for a U.S. Treasury bill.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* > *maturity*, `#NUM!` is returned.
- *maturity* is more than one year after *settlement*, `#NUM!` is returned.
- *discount* ≤ 0, `#NUM!` is returned.

[*Example*:

`TBILLPRICE(DATE(2008,3,31),DATE(2008,6,1),0.09)` results in `98.4500`

end example]

18.17.7.320 TBILLYIELD

`TBILLYIELD (` *settlement* `,` *maturity* `,` *pr* `)`

Description: Computes the yield for a U.S. Treasury bill.

Mathematical Formula:

$$TBILLYIELD = \frac{100 - pr.}{pr.} \times \frac{360}{DSM}$$

where:

- DSM = number of days from settlement to maturity, excluding any maturity date that is more than one calendar year after the settlement date.
- $pr.$ = argument pr

Arguments:

Name	Type	Description
settlement	number	The Treasury bill's settlement date. Any time information in the date is ignored.
maturity	number	The Treasury bill's maturity date. Any time information in the date is ignored.
pr	number	The Treasury bill's price per $100 face value.

Return Type and Value: number – The yield for a U.S. Treasury bill.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* > *maturity*, `#NUM!` is returned.
- *maturity* is more than one year after *settlement*, `#NUM!` is returned.
- *pr* ≤ 0, `#NUM!` is returned.

[*Example*:

`TBILLYIELD(DATE(2008,3,31),DATE(2008,6,1),98.45)` results in `9.1417%`

end example]

18.17.7.321 TDIST

Syntax:

TDIST (*x* , *degrees-freedom* , *distribution-tails*)

Description: Computes the Percentage Points (probability) for the Student t-distribution where a numeric value, *x*, is a calculated value of *t* for which the Percentage Points are to be computed.

Mathematical Formula:

If *distribution-tails* = 1, $TDIST = P(X > x)$, where X is a random variable that follows the t-distribution.

If *distribution-tails* = 2, $TDIST = P(|X| > x) = P(X > x \text{ or } X < -x)$

Arguments:

Name	Type	Description
x	number	The value at which to evaluate the distribution.
degrees-freedom	number	The number of degrees of freedom, truncated to an integer.
distribution-tails	number	The number of distribution tails to return, truncated to an integer. If 1, TDIST returns the one-tailed distribution.

Name	Type	Description
		If 2, TDIST returns the two-tailed distribution.

Return Type and Value: number – The Percentage Points (probability) for the Student t-distribution.

However, if

- *degrees-freedom* < 1, #NUM! is returned.
- *tails* has any value other than 1 or 2, #NUM! is returned.
- *x* < 0, #NUM! is returned.

[*Example*:

TDIST(1.959999998,60,1) results in 0.027322464
TDIST(1.959999998,60,2) results in 0.054644927

end example]

18.17.7.322 TEXT

Syntax:

TEXT (*value* , *format*)

Description: Produces a string containing *value* formatted according to *format*.

Arguments:

Name	Type	Description
value	number	The number that is to be formatted.
format	text	Designates the number, currency, date, or time format to be used. (See §18.8.31 for the set of formats.)

Return Type and Value: text – The string containing *number* formatted according to *format*.

[*Example*:

TEXT(1234.567,"$0.00") results in $1234.57
TEXT(.125,"$0.0%") results in 12.5%
TEXT(1234.567,"YYYY-MM-DD HH:MM:SS") results in 1903-05-18 13:36:29 in the 1900 date system.

end example]

18.17.7.323 TIME

Syntax:

`TIME ( `*hour*` , `*minute*` , `*second*` )`

Description: Computes the serial date-time for the given time.

Arguments:

Name	Type	Description
hour	number	A number in the range 0–32767, inclusive, truncated to integer, that represents the hour. Any value greater than 23 shall be divided by 24 and the remainder shall be treated as the hour value.
minute	number	A number in the range 0–32767, inclusive, truncated to integer, that represents the minute. Any value greater than 59 shall be converted to the corresponding number of hours and minutes.
second	number	A number in the range 0–32767, inclusive, truncated to integer, that represents the second. Any value greater than 59 shall be converted to the corresponding number of hours, minutes, and seconds.

Return Type and Value: number – The serial date-time for the given time, as a value greater than or equal to 0 and less than or equal to 1.

However, if *hour*, *minute*, or *second* are out of range, `#NUM!` is returned.

[*Example*: The following serial date-times are displayed with 16 decimal places.

`TIME(0,0,0)` results in a serial date-time of `0.0000000000000000`
`TIME(0,0,1)` results in a serial date-time of `0.0000115740740741`
`TIME(0,0,2)` results in a serial date-time of `0.0000231481481481`
`TIME(0,0,20)` results in a serial date-time of `0.0002314814814815`
`TIME(2,3,20)` results in a serial date-time of `0.0856481481481481`
`TIME(12,0,0)` results in a serial date-time of `0.5000000000000000`
`TIME(23,59,59)` results in a serial date-time of `0.9999884259259260`
`TIME(26,120,240)` results in a serial date-time of `0.1694444444444450`

end example]

18.17.7.324 TIMEVALUE

Syntax:

`TIMEVALUE ( date-time-string )`

Description: Computes the serial date-time of the time represented by the string *date-time-string*.

Arguments:

Name	Type	Description
date-time-string	text	The date and/or time whose time component serial date-time is to be computed. *date-time-string* can have any date and/or time format. Any date information in *date-time-string* shall be ignored.

Return Type and Value: number – The serial date-time of the time represented by the string *date-time-string*, as a value greater than or equal to 0 and less than or equal to 1.

However, if *date-time-string* is ill-formed, #VALUE! is returned.

[*Example*: The following serial date-times are displayed with 16 decimal places.

`TIMEVALUE("10:02:34 ")` results in `0.4184490740740740`
`TIMEVALUE("01-Feb-2006 10:15:29 AM")` results in `0.4274189814823330`
`TIMEVALUE("22:02")` results in `0.9180555555555560`

end example]

18.17.7.325 TINV

Syntax:

`TINV ( probability , degrees-freedom )`

Description: Computes the t-value of the Student's t-distribution as a function of the probability and the degrees of freedom.

Arguments:

Name	Type	Description
probability	number	A probability associated with the two-tailed Student's t-distribution.
degrees-freedom	number	The number of degrees of freedom with which to characterize the distribution, truncated to an integer.

Return Type and Value: number – The t-value of the Student's t-distribution.

However, if

- *probability* < 0 or *probability* > 1, `#NUM!` is returned.
- *degrees-freedom* < 1, `#NUM!` is returned.

[*Example*:

`TINV(0.054644927,60)` results in `1.95999999`

end example]

18.17.7.326 TODAY

Syntax:

`TODAY ( )`

Description: Computes the serial date-time of the current date, taking into account the current date base system.

Arguments: None.

Return Type and Value: number – The serial date-time of the current date.

[*Example*:

On February 25, 2006, `TODAY()` results in 38773 for the 1900 date system, or 37311 for the 1904 date system

end example]

18.17.7.327 TRANSPOSE

Syntax:

`TRANSPOSE (` *array* `)`

Description: Creates a new array that is the transpose of an existing array, by copying the first row of the existing array to the first column of the new array, the second row of the existing array as the second column of the new array, and so on. The formula containing the call to `TRANSPOSE` shall be an array formula in a range that has the same number of rows and columns, respectively, as *array* has columns and rows.

Arguments:

Name	Type	Description
array	array, reference	The set of values to be transposed.

Return Type and Value: array – The new array.

[*Example*:

`TRANSPOSE({10,20,30})` results in the array `{10;20;30}`

end example]

18.17.7.328 TREND

Syntax:

`TREND ( ` *known-ys* `[ , [` *known-xs* `][ , [` *new-xs* `][ ,` *const-flag* `]] )`

Description: Computes values along a linear trend. Fits a straight line (using the method of least squares) to the arrays *known-ys* and *known-xs*. The y-values along that line for the array of *new-xs* specified.

Arguments:

Name	Type	Description
known-ys	array	The set of y-values already known in the relationship *y=mx+b*. If that array is in a single column, each column of *known-xs* is interpreted as a separate variable. If that array is in a single row, each row of *known-xs* is interpreted as a separate variable.
known-xs	array	An optional set of x-values that might already be known in the relationship *y=mx+b*. The array *known-xs* can include one or more sets of variables. If only one variable is used, *known-ys* and *known-xs* can be ranges of any shape, as long as they have equal dimensions. If more than one variable is used, *known-ys* shall be a vector. If *known-xs* is omitted, it is assumed to be the array {1,2,3,...} that is the same size as *known-ys*.
new-xs	array	New x-values for which TREND is to return corresponding y-values. *new-xs* shall include a column (or row) for each independent variable, just as *known-xs* does. So, if *known-ys* is in a single column, *known-xs* and *new-xs* shall have the same number of columns. If *known-ys* is in a single row, *known-xs* and *new-xs* shall have the same number of rows. If *new-xs* is omitted, it is assumed to be the same as *known-xs*. If both *known-xs* and *new-xs* are omitted, they are assumed to be the array {1,2,3,...} that is the same size as *known-ys*.
const-flag	logical	Specifies whether to force the constant b to equal 0. If TRUE or omitted, *b* is calculated normally. If FALSE, *b* is set equal to 0 and the m-values are adjusted so that *y=mx*.

Return Type and Value: array – The values along a linear trend, as an array of numbers.

18.17.7.329 TRIM

Syntax:

`TRIM ( string )`

Description: Makes a string that is a copy of *string* with the leading and trailing space characters removed, and each sequence of embedded spaces reduced to a single space. The space character referred to here is character `U+0020`.

Arguments:

Name	Type	Description
string	text	Designates the string to be trimmed.

Return Type and Value: text – The trimmed copy of *string*.

[*Example*:

`TRIM("     abc     def   ")` results in `abc def`

end example]

18.17.7.330 TRIMMEAN

Syntax:

`TRIMMEAN ( array , percent )`

Description: Computes the mean of the interior of a data set by excluding a percentage of data points from the top and bottom tails of a data set. `TRIMMEAN` rounds the number of excluded data points down to the nearest multiple of 2. For symmetry, `TRIMMEAN` excludes a single value from the top and bottom of the data set.

Arguments:

Name	Type	Description
array	array, reference	The numeric values to trim and average.
percent	number	The fractional number of data points to exclude from the calculation. [*Example*: If *percent* = 0.2, 4 points are trimmed from a data set of 20 points (20x0.2): 2 from the top and 2 rom the bottom of the set. *end example*]

Return Type and Value: number – The mean of the interior of a data set.

However, if *percen* < 0 or *percen* > 1, `#NUM!` is returned.

[*Example*:

`TRIMMEAN({4,6,2,5,7,8,9},0.2)` results in `5.857142857`

end example]

18.17.7.331 TRUE

Syntax:

`TRUE ( )`

Description: Computes the value TRUE. (A call to function TRUE is equivalent to using the *logical-constant* TRUE.)

Arguments: None.

Return Type and Value: logical – The value TRUE.

[*Example*:

`TRUE()` results in TRUE

end example]

18.17.7.332 TRUNC

Syntax:

`TRUNC ( ` *x* ` [ , ` *number-digits* ` ] )`

Description: Truncates *x* to the number of fractional digits by *number-digits*.

Arguments:

Name	Type	Description
x	array, reference	The value to be rounded down.
number-digits	number	The number of fractional digits to which *x* is to be truncated. The default value for *number-digits* is 0.

Return Type and Value: number – The truncated value of *x*.

[*Example*:

`TRUNC(PI())` results in 3

`TRUNC(PI(),1)` results in `3.1`
`TRUNC(PI(),3)` results in `3.141`
`TRUNC(PI(),5)` results in `3.14159`

end example]

18.17.7.333 TTEST

Syntax:

TTEST (*array-1* , *array-2* , *distribution-tails* , *test-type*)

Description: Computes the probability associated with a Student's t-Test.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>array-1</td><td>array, reference</td><td>The first numerical data set.</td></tr>
<tr><td>array-1</td><td>array, reference</td><td>The first numerical data set.</td></tr>
<tr><td>distribution-tails</td><td>number</td><td>Specifies the number of distribution tails, truncated to an integer. If 1, TTEST uses the one-tailed distribution. If 2, TTEST uses the two-tailed distribution.</td></tr>
<tr><td>test-type</td><td>number</td><td>The truncated-to-integer kind of t-Test to perform, as follows:
<table>
<tr><th>Value</th><th>Test Performed</th></tr>
<tr><td>1</td><td>Paired</td></tr>
<tr><td>2</td><td>Two-sample equal variance (homoscedastic)</td></tr>
<tr><td>3</td><td>Two-sample unequal variance (heteroscedastic)</td></tr>
</table></td></tr>
</table>

Return Type and Value: number – The probability associated with a Student's t-Test.

However, if

- *array-1* and *array-2* have a different number of data points, and *test-type* is 1, the return value is unspecified.
- *distribution-tails* is any value other than 1 or 2, `#NUM!` is returned.

[*Example*: Given the following data:

	A	B
1	Data 1	Data 2

	A	B
2	3	6
3	4	19
4	5	3
5	8	2
6	9	14
7	1	4
8	2	5
9	4	17
10	5	1

`TTEST(A2:A10,B2:B10,2,1)` results in `0.196016`

end example]

18.17.7.334 TYPE

Syntax:

`TYPE (` *value* `)`

Description: Computes the type of *value* or, if *value* is a reference to a single cell, the type of the value in that cell.

Arguments:

Name	Type	Description
value	any	The value whose type is to be determined. No conversion shall take place on an argument passed to this function.

Return Type and Value: number – An integer that indicates the type of *value* or, if *value* is a reference to a single cell, the type of the value in that cell, as follows:

Type of *value*	Value Returned
number	1
text	2
logical	4
error value	16
array of any kind	64

[*Example*:

`TYPE(10.5)` results in 1
`TYPE(A10)` results in 1, when A10 contains a number
`TYPE("ABC")` results in 2
`TYPE(A10)` results in 2, when A10 contains a string
`TYPE(TRUE)` results in 4
`TYPE(A10)` results in 4, when A10 contains a logical value

`TYPE(5/0)` results in 16
`TYPE(A10)` results in 16, when A10 contains any error value
`TYPE({1,2,3})` results in 64
`TYPE({TRUE,2.5,#N/A})` results in 64
`TYPE(IF(10>5,"Yes",20))` results in 2
`TYPE(IF(10<5,"Yes",20))` results in 1

end example]

18.17.7.335 UPPER

Syntax:

`UPPER (` *string* `)`

Description: Makes an uppercase version of *string*.

Arguments:

Name	Type	Description
string	text	Designates the string to be converted.

Return Type and Value: text – The uppercase version of *string*.

[*Example*:

`UPPER("AbCd123#$%^")` results in ABCD123#$%^
`UPPER(A10)` results in 234FRTQWC$#%, when A10 contains 234FRTqwc$#%

end example]

18.17.7.336 USDOLLAR

Syntax:

`USDOLLAR (` *number* `[ ,` *num-decimal* `] )`

Description: Produces a string containing *number* rounded to *num-decimal* decimal places. The thousands separator is the comma, the radix point is the period, and the currency symbol is "$". The format used is $#,##0.00;($#,##0.00).

Arguments:

Name	Type	Description
number	number	The number that is to be formatted.
num-decimal	number	Designate the number of decimal places to be used in the resulting string; it is truncated to an integer. If *num-decimal* is negative, *number* is rounded to the left of the decimal point. If omitted, a value of 2 shall be assumed.

Return Type and Value: text – The string containing *number* rounded to *num-decimal* decimal places, and have a currency symbol and thousands separators.

[*Example*:

`USDOLLAR(1234.567)` results in `$1,234.57`
`USDOLLAR(1234.567,-2)` results in `$1,200`
`USDOLLAR(-1234.567,4)` results in `($1,234.5670)`

end example]

18.17.7.337 VALUE

Syntax:

VALUE (*string*)

Description: Converts *string* to a number.

Arguments:

Name	Type	Description
string	text	Designates a string that contains a number formatted using any number, currency, date, or time format. (See §18.8.31 for the set of formats.) Date and time strings are converted to their equivalent serial date-time.

Return Type and Value: number – The number represented by *string*.

[*Example*:

VALUE("123.456") results in 123.456
VALUE("$1,000") results in 1000
VALUE("23-Mar-2002") results in the corresponding serial date-time
VALUE("16:48:00")-VALUE("12:17:12") results in 0.188056

end example]

18.17.7.338 VAR

Syntax:

VAR (*argument-list*)

Description: Makes an estimate of the variance based on a sample. [*Note*: VAR assumes that its arguments are a sample of the population. If the data represents the entire population, VARP should be used instead. If logical values and text representations of numbers in a reference are to be included as part of the calculation, use VARA instead. *end note*]

Mathematical Formula:

$$\frac{\sum(x-\bar{x})^2}{(n-1)}$$

where:

- n = the sample size
- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*argument-1*, *argument-1*,…, *argument-n*)

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *arguments* in *argument-list* designate the numbers that are samples of the population. Logical values, and text representations of numbers that are entered directly into the list of arguments are included. If an argument is an array or reference, only numbers in that array or reference are included. Empty cells, logical values, text, or error values in the array or reference are ignored.

Return Type and Value: number – An estimate of the variance based on a sample.

[*Example*:

VAR(1202,1220,1323,1254,1302) results in 2683.2

end example]

18.17.7.339 VARA

Syntax:

VARA (*argument-list*)

Description: Makes an estimate of the variance based on a sample. [*Note*: VARA assumes that its arguments are a sample of the population. If the data represents the entire population, VARPA should be used instead. If logical values and text representations of numbers in a reference are to be excluded as part of the calculation, use VAR instead. *end note*]

Mathematical Formula:

$$\frac{\sum(x-\bar{x})^2}{(n-1)}$$

where:

- n = the sample size
- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE($argument\text{-}1$, $argument\text{-}1$,…, $argument\text{-}n$)

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *arguments* in *argument-list* designate the numbers that are samples of the population. Logical values and text representations of numbers that are entered directly into the list of arguments are included. Arguments that contain TRUE evaluate as 1; arguments that contain text or FALSE evaluate as zero. If an argument is an array or reference, only values in that array or reference are used. Empty cells and text values in the array or reference are ignored.

Return Type and Value: number – An estimate of the variance based on a sample.

[*Example*:

`VARA(1202,1220,1323,1254,1302)` results in `2683.2`

end example]

18.17.7.340 VARP

Syntax:

VARP (*argument-list*)

Description: Computes the variance of an entire population. [*Note*: VARP assumes that its arguments are the total population. If the data represents a population sample only, VAR should be used instead. If logical values and text representations of numbers in a reference are to be included as part of the calculation, use VARPA instead. *end note*]

Mathematical Formula:

$$\frac{\sum(-\bar{x})^2}{n}$$

where:

- n = the sample size
- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*argument-1*, *argument-1*,…, *argument-n*)

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *arguments* in *argument-list* designate the numbers that are the members of the population. Logical values, and text representations of numbers that are entered directly into the list of arguments are included. If an argument is an array or reference, only numbers in that array or reference are included. Empty cells, logical values, text, or error values in the array or reference are ignored.

Return Type and Value: number – The variance of an entire population.

[*Example*:

`VARP(1202,1220,1323,1254,1302)` results in `2146.56`

end example]

18.17.7.341 VARPA

Syntax:

VARPA (*argument-list*)

Description: Makes the variance of an entire population. [*Note*: VARPA assumes that its arguments are the total population. If the data represents a population sample only, VARA should be used instead. If logical values and text representations of numbers in a reference are to be excluded as part of the calculation, use VARP instead. *end note*]

Mathematical Formula:

$$\frac{\sum(x-\bar{x})^2}{n}$$

where:

- n = the sample size
- x = a sample value
- $\bar{x}$ = the sample mean AVERAGE(*argument-1*, *argument-1*,…, *argument-n*)

Arguments:

Name	Type	Description
argument-list	logical, number, name, text, array, reference	The *arguments* in *argument-list* designate the numbers that are the members of the population.

Return Type and Value: number – The variance of an entire population.

Arguments can be numbers; names, arrays, or references that contain numbers; text representations of numbers; or logical values, in a reference. Text representations of numbers that are entered directly into the list of arguments are included. Arguments that contain TRUE evaluate as 1; arguments that contain text or FALSE evaluate as zero. If an argument is an array or reference, only values in that array or reference are used. Empty cells and text values in the array or reference are ignored.

[*Example*:

`VARPA(1202,1220,1323,1254,1302)` results in `2146.56`

end example]

18.17.7.342 VDB

VDB (*cost* , *salvage* , *life* , *start-period* , *end-period* [, [[*factor*]
[, [*no-switch-flag*]]]]])

Description: Computes the depreciation of an asset for the period specified, including partial periods, using the double-declining balance or some other specified method.

Arguments:

Name	Type	Description
cost	number	The number *cost* is the initial cost of the asset.
salvage	number	The value at the end of the depreciation. (This is sometimes called the salvage value of the asset.) This value can be 0.
life	number	The number of periods over which the asset is being depreciated. (This is sometimes called the useful life of the asset.)
start-period	number	The starting period for which the depreciation is to be calculated. (*start-period* shall use the same units as *life*.)
end-period	number	The ending period for which the depreciation is to be calculated. (*end-period* shall use the same units as *life*.)
factor	number	The rate at which the balance declines. If omitted, it is assumed to be 2 (the double-declining balance method).
no-switch-flag	logical	Specifies whether to switch to straight-line depreciation when depreciation is greater than the declining balance calculation. If TRUE, straight-line depreciation is not used even when the depreciation is greater than the declining balance calculation. If FALSE or omitted, the straight-line depreciation is used when depreciation is greater than the declining balance calculation.

Return Type and Value: number – The depreciation of an asset for the period specified.

However, if any numerical argument value is non-positive, `#NUM!` is returned.

[*Example*:

`VDB(2400,300,10*365,0,1)` results in `1.32`
`VDB(2400,300,10*12,0,1)` results in `40.00`
`VDB(2400,300,10*12,6,18)` results in `396.31`

end example]

18.17.7.343 VLOOKUP

Syntax:

VLOOKUP (*lookup-value* , *table-array* , *col-index-num* [, [*range-lookup-flag*]])

Description: Performs a vertical search for a value in the left-most column of a table or an array, noting the row in which the matching value is found. From that row, the value from a given column is returned.

Arguments:

Name	Type	Description
lookup-value	value of any type or a reference to a value of any type.	The value to be located in the left-most column of the table. If *range-lookup* is FALSE and *lookup-value* is a string, the wildcard characters, question mark (?) and asterisk (*), can be included in *lookup-value*. A question mark matches any single character; an asterisk matches any sequence of characters. To find a question mark or asterisk, use a tilde (~) before the character.
table-array	array, reference, name	Designates the table of information to be searched. The values in the left-most column of *table-array* can be text, numbers, or logical values. The values in the left-most column of *table-array* shall be placed in "ascending order", as follows: …, -2, -1, 0, 1, 2, …, A–Z, FALSE, TRUE. Uppercase and lowercase text is treated as equivalent.
col-index-num	number	The column number in *table-array* from which the matching value is to be returned. (A *col-index-num* of 1 returns the left-most column value in *table-array*, a *col-index-num* of 2 returns the next column in *table-array*, and so on.)
range-lookup-flag	logical	Specifies whether HLOOKUP is to find an exact or approximate match. If TRUE or omitted, an approximate match is returned. That is, if an exact match is not found, the next largest value that is less than *lookup-value* is returned. If FALSE, an exact match is performed, in which case, the values in the left-most column of *table-array* need not be sorted. If there are two or more values in the left-most column of *table-array* that match *lookup-value*, the top-most value found is used.

Return Type and Value: any – The value from a given row number, where the column is determined by a search of the top row looking for a match with a given value.

However, if

- An exact match is performed, but no match is found, #N/A is returned.
- *col-index-num* is less than 1, #VALUE! is returned.
- *col-index-num* is greater than the number of columns in *table-array*, #REF! is returned.
- *lookup-value* is smaller than the smallest value in the left-most column of *table-array*, #N/A is returned.

[*Example*: Given the following data:

	A	B	C
1	Density	Bearings	Bolts
2	0.457	3.55	500

	A	B	C
3	0.525	3.25	400
4	0.616	2.93	300
5	0.675	2.75	250
6	0.746	2.57	200
7	0.835	2.38	150
8	0.946	2.17	100
9	1.09	1.95	50
10	1.29	1.71	0

`VLOOKUP(1,A2:C10,2)` results in `2.17`
`VLOOKUP(1,A2:C10,3,TRUE)` results in `100.00`
`VLOOKUP(2,A2:C10,2,TRUE)` results in `1.71`

end example]

18.17.7.344 WEEKDAY

Syntax:

`WEEKDAY (` *serial-value* `[ ,` *weekday-start-flag* `] )`

Description: Computes the weekday number for the date having the given *serial-value*, taking into account the current date system and *weekday-start-flag*, if present. See §18.17.4.1 for special handling of certain days in 1900.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>serial-value</td><td>number</td><td>The date whose weekday number is to be computed. The value of serial-value is truncated to an integer.</td></tr>
<tr><td>weekday-start-flag</td><td>number</td><td>When truncated to integer, indicates the weekday numbering convention to be used, as follows:
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>1 or omitted</td><td>1 (Sunday) through 7 (Saturday)</td></tr>
<tr><td>2</td><td>1 (Monday) through 7 (Sunday)</td></tr>
<tr><td>3</td><td>0 (Monday) through 6 (Sunday)</td></tr>
<tr><td>11</td><td>1 (Monday) through 7 (Sunday)</td></tr>
<tr><td>12</td><td>1 (Tuesday) through 7 (Monday)</td></tr>
</table>
</td></tr>
</table>

Name	Type	Description	
		13	1 (Wednesday) through 7 (Tuesday)
		14	1 (Thursday) through 7 (Wednesday)
		15	1 (Friday) through 7 (Thursday)
		16	1 (Saturday) through 7 (Friday)
		17	1 (Sunday) through 7 (Saturday)

Return Type and Value: number – The weekday number for the date having the given serial date-time.

However, if

- *serial-value* is out of range for the current date system, `#NUM!` is returned.
- *weekday-start-flag* is out of the range specified in the table above, `#NUM!` is returned.

[*Example*:

`WEEKDAY(DATE(2006,2,1))` results in 4 (Wednesday)
`WEEKDAY(DATE(2006,2,1),11)` results in 3 (Wednesday)
`WEEKDAY(DATE(2006,2,1),12)` results in 2 (Wednesday)
`WEEKDAY(DATE(2006,2,1),3)` results in 2 (Wednesday)

end example]

18.17.7.345 WEEKNUM

Syntax:

`WEEKNUM (` *serial-value* `[ ,` *weekday-start-flag* `] )`

Description: Computes the week number of the date corresponding to *serial-value*. The function allows two number systems:

- System 1: The week containing January 1 is the first week of the year, and is numbered week 1.
- System 2: The week containing the first Thursday of the year is the first week of the year, and is numbered as week 1 [ISO 8601].

Arguments:

Name	Type	Description
serial-value	number	The date whose week number is to be computed. The value of *serial-value* is truncated to an integer.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>weekday-start-flag</td><td>number</td><td>When truncated to integer, indicates the weekday on which the week begins, as follows:
<table>
<tr><th>weekday-start-flag</th><th>Meaning</th><th>Number System</th></tr>
<tr><td>1 or omitted</td><td>Week begins on Sunday.</td><td>System 1</td></tr>
<tr><td>2</td><td>Week begins on Monday.</td><td>System 1</td></tr>
<tr><td>11</td><td>Week begins on Monday.</td><td>System 1</td></tr>
<tr><td>12</td><td>Week begins on Tuesday.</td><td>System 1</td></tr>
<tr><td>13</td><td>Week begins on Wednesday.</td><td>System 1</td></tr>
<tr><td>14</td><td>Week begins on Thursday.</td><td>System 1</td></tr>
<tr><td>15</td><td>Week begins on Friday.</td><td>System 1</td></tr>
<tr><td>16</td><td>Week begins on Saturday.</td><td>System 1</td></tr>
<tr><td>17</td><td>Week begins on Sunday.</td><td>System 1</td></tr>
<tr><td>21</td><td>Week begins on Monday.</td><td>System 2</td></tr>
</table>
</td></tr>
</table>

Return Type and Value: number – The week number of the date corresponding to *serial-value*.

However, if

- *serial-value* is out of range for the current date system, `#NUM!` is returned.
- *weekday-start-flag* is out of the range specified in the table above, `#NUM!` is returned.

[*Example*:

`WEEKNUM(DATE(2006,1,1)` results in 1
`WEEKNUM(DATE(2006,1,1),1)` results in 1
`WEEKNUM(DATE(2006,1,1),17)` results in 1
`WEEKNUM(DATE(2006,1,1),21)` results in 1
`WEEKNUM(DATE(2006,2,1),1)` results in 5

`WEEKNUM(DATE(2006,2,1),2)` results in 6
`WEEKNUM(DATE(2006,2,1),11)` results in 6

end example]

18.17.7.346 WEIBULL

Syntax:

`WEIBULL ( `*x*` , `*alpha*` , `*beta*` , `*cumulative-flag*` )`

Description: Computes the Weibull distribution.

Mathematical Formula:

The equation for the Weibull cumulative distribution function is:

$$F(x, \alpha, \beta) = 1 - e^{-(x/\beta)^{\alpha}}$$

The equation for the Weibull probability density function is:

$$f(x, \alpha, \beta) = \frac{}{\beta^{\alpha}} x^{\alpha-1} e^{-(x/\beta)^{\alpha}}$$

When *alpha* = 1, `WEIBULL` returns the exponential distribution with:

$$\lambda = \frac{1}{\beta}$$

where:

- x = argument x
- α = argument *alpha*
- β = argument *beta*

Arguments:

Name	Type	Description
x	number	The value at which the distribution is to be evaluated.
alpha	number	A parameter of the distribution.
beta	number	A parameter of the distribution.
cumulative-flag	logical	Determines the form of the function. If TRUE, `GAMMADIST` returns the cumulative distribution function; if `FALSE`, it returns the probability density function.

Return Type and Value: number – The Weibull distribution.

However, if

- *x* < 0, `#NUM!` is returned.
- *alpha* ≤ 0, `#NUM!` is returned.
- *beta* ≤ 0, `#NUM!` is returned.

[*Example*:

`WEIBULL(105,20,100,TRUE)` results in `0.92958139`
`WEIBULL(105,20,100,FALSE)` results in `0.035588864`

end example]

18.17.7.347 WORKDAY

Syntax:

`WORKDAY ( ` *start-date* ` , ` *day-offset* ` [ , ` *holidays* ` ] )`

Description: Computes the serial date-time of the date that is *day-offset* working days offset from *start-date*. Weekend days (Saturday and Sunday) and any holidays specified by *holidays* are not considered as working days.

Arguments:

Name	Type	Description
start-date	number	The start date, truncated to integer.
day-offset	number	The number of working days before or after *start-date*. A positive value yields a future date; a negative value yields a past date; a zero value yields the date *start-date*. *day-offset* is truncated to an integer.
holidays	reference, array	An optional set of one or more dates that are to be excluded from the working day calendar. *holidays* shall be a range of cells that contain the dates, or an array constant of the serial date-times that represent those dates. The ordering of dates or serial date-times in *holidays* can be arbitrary.

Return Type and Value: number – The serial date-time of the date that is *day-offset* working days offset from *start-date*, excluding the specified holidays.

However, if

- *start-date* is out of range for the current date system, `#NUM!` is returned.
- Any date in *holidays* is out of range for the current date system, `#NUM!` is returned.

- *start-date* plus *day-offset* does not yield a date, `#NUM!` is returned.

[*Example*:

`WORKDAY(DATE(2006,1,1),0)` results in a serial date-time corresponding to 1-Jan-2006
`WORKDAY(DATE(2006,1,1),10)` results in a serial date-time corresponding to 13-Jan-2006
`WORKDAY(DATE(2006,1,1),-10)` results in a serial date-time corresponding to 19-Dec-2005
`WORKDAY(DATE(2006,1,1),20,{"2006/1/2","2006/1/16"})` results in a serial date-time corresponding to 31-Jan-2006

end example]

18.17.7.348 WORKDAY.INTL

Syntax:

Number form: `WORKDAY.INTL ( `*start-date*` , `*day-offset* [, [*weekend-number*] [, *holidays*]])

String form: `WORKDAY.INTL ( `*start-date*` , `*day-offset* [, [*weekend-string*] [, *holidays*]])

Description: Computes the serial date-time of the date that is *day-offset* working days offset from *start-date*. Weekend days and any holidays specified by *holidays* are not considered as working days.

Arguments:

Name	Type	Description
start-date	number	The start date, truncated to integer.
day-offset	number	The number of working days before or after *start-date*. A positive value yields a future date; a negative value yields a past date; a zero value yields the date *start-date*. *day-offset* is truncated to an integer.
weekend-number	number	Indicates the days of the week that are weekend days and are not considered working days. Values are shown in the table below
weekend-string	string	Indicates the days of the week that are weekend days and are not considered working days. Values of *weekend-string* are seven characters long and each character in the string represents a day of the week, beginning with Monday. [*Example*: "0000011" would result in a weekend that is Saturday and Sunday. *end example*]
holidays	reference, array	An optional set of one or more dates that are to be excluded from the working day calendar. *holidays* shall be a range of cells that contain the dates, or an array constant of the serial date-times that represent those

Name	Type	Description
		dates. The ordering of dates or serial date-times in *holidays* can be arbitrary.

weekend-number	Weekend days
1 or omitted	Saturday, Sunday
2	Sunday, Monday
3	Monday, Tuesday
4	Tuesday, Wednesday
5	Wednesday, Thursday
6	Thursday, Friday
7	Friday, Saturday
11	Sunday only
12	Monday only
13	Tuesday only
14	Wednesday only
15	Thursday only
16	Friday only
17	Saturday only

Return Type and Value: number – The serial date-time of the date that is *day-offset* working days offset from *start-date*, excluding the specified weekend days and holidays.

However, if

- *start-date* is out of range for the current date system, `#NUM!` is returned.
- Any date in *holidays* is out of range for the current date system, `#NUM!` is returned.
- *start-date* plus *day-offset* does not yield a date, `#NUM!` is returned.

[*Example*:

`WORKDAY.INTL(DATE(2006,1,1),0)` results in a serial date-time corresponding to 1-Jan-2006
`WORKDAY.INTL(DATE(2006,1,1),10)` results in a serial date-time corresponding to 13-Jan-2006
`WORKDAY.INTL(DATE(2006,1,1),10,7)` results in a serial date-time corresponding to 13-Jan-2006
`WORKDAY.INTL(DATE(2006,1,1),-10)` results in a serial date-time corresponding to 19-Dec-2005
`WORKDAY.INTL(DATE(2006,1,1),20,1,{"2006/1/2","2006/1/16"})` results in a serial date-time corresponding to 31-Jan-2006
`WORKDAY.INTL(DATE(2006,1,1),20,"0000011",{"2006/1/2","2006/1/16"})` results in a serial date-

time corresponding to 31-Jan-2006

end example]

18.17.7.349 XIRR

`XIRR ( values , dates [ , [ guess ]] )`

Description: Computes the internal rate of return for a schedule of cash flows that is not necessarily periodic. XIRR uses an iterative calculation technique that cycles through the calculation until the result is accurate within 0.000001 percent.

Mathematical Formula:

Using a changing rate (starting with *guess*), XIRR cycles through the calculation until the result is accurate within 0.000001 percent. The rate is changed until:

$$0 = \sum_{i=1}^{N} \frac{P_i}{(1 + rate)^{\frac{(d_i - d_1)}{365}}}$$

where:

- d_i = the i^{th}, or last, payment date.
- d_1 = the 0^{th} payment date.
- P_i = the i^{th}, or last, payment.

Arguments:

Name	Type	Description
values	array, reference	A series of cash flows that corresponds to a schedule of payment dates specified in *dates*. The first payment is optional and corresponds to a cost or payment that occurs at the beginning of the investment. If the first value is a cost or payment, it shall have a negative value. All succeeding payments are discounted based on a 365-day year. The series of values shall contain at least one positive and one negative value.
dates	reference	A schedule of payment dates that corresponds to the cash flow payments in *values*. The first payment date indicates the beginning of the schedule of payments. All other dates shall be later than this date, but they can occur in any order. Time information in the date arguments is ignored.
guess	number	An estimate of the result of XIRR. If omitted, it is assumed to be 0.1 (i.e., 10 percent).

Return Type and Value: number – The internal rate of return for a schedule of cash flows that is not necessarily periodic.

However, if

- Any date in *dates* is out of range for the current date system, `#NUM!` is returned.
- Any date in *dates* precedes the starting date, `#NUM!` is returned.
- *values* and *dates* contain different numbers of values, `#NUM!` is returned.
- The calculation has not converged after an implementation-defined number of tries, `#NUM!` is returned.

[*Example*: When the cells `F2397:J2397`contain the dates January 1, 2008; March 1,2008; October 30, 2008; February 15, 2009, and April 1, 2009:

`XIRR({-10000,2750,4250,3250,2750},F2397:J2397,0.1)` results in `37.34%`

end example]

18.17.7.350 XNPV

XNPV (*rate* , *values* , *dates*)

Description: Computes the net present value for a schedule of cash flows that is not necessarily periodic.

Mathematical Formula:

$$XNPV = \sum_{i=1}^{N} \frac{P_i}{(1+rte)^{\frac{(d_i-d_1)}{365}}}$$

where:

- d_i = the i^{th}, or last, payment date.
- d_1 = the 0^{th} payment date.
- P_i = the i^{th}, or last, payment.

Arguments:

Name	Type	Description
rate	number	The discount rate to apply to the cash flows.
values	array, reference	A series of cash flows that corresponds to a schedule of payment dates specified in *dates*. The first payment is optional and corresponds to a cost or payment that occurs at the beginning of the investment. If the first value is a cost or payment, it shall have a negative value. All succeeding payments are discounted based on a 365-day year. The series of values shall contain at least one positive and one negative value.

Name	Type	Description
dates	reference	A schedule of payment dates that corresponds to the cash flow payments in *values*. The first payment date indicates the beginning of the schedule of payments. All other dates shall be later than this date, but they can occur in any order. Time information in the date arguments is ignored.

Return Type and Value: number – The net present value for a schedule of cash flows that is not necessarily periodic.

However, if

- Any date in *dates* is out of range for the current date system, `#NUM!` is returned.
- Any date in *dates* precedes the starting date, `#NUM!` is returned.
- *values* and *dates* contain different numbers of values, `#NUM!` is returned.

[*Example*: When the cells `F2397:J2397` contain the dates January 1, 2008; March 1,2008; October 30, 2008; February 15, 2009, and April 1, 2009:

`XNPV(0.09,{-10000,2750,4250,3250,2750},F2397:J2397)` results in `2086.65`

end example]

18.17.7.351 YEAR

Syntax:

YEAR (*date-value*)

Description: Computes the numeric year in the Gregorian calendar [ISO 8601 §3.2.1] for the date and/or time having the given *date-value*, taking into account the current date system. That date and/or time shall be expressed either as a serial date-time, in which case, its fractional part is ignored, or as a *string-constant* having any date and/or time format, in which case, any time information shall be ignored.

Arguments:

Name	Type	Description
date-value	number, text	The date and/or time whose year is to be computed. That date and/or time shall be expressed either as a serial date-time, in which case, its fractional part is ignored, or as a *string-constant* having any date and/or time format, in which case, any time information shall be ignored.

Return Type and Value: number – The year in the Gregorian calendar [ISO 8601 §3.2.1] for the date and/or time having the given *date-value*. The range of return values is determined by the date system currently in use (§18.17.4).

However, if *date-value* is out of range for the current date system, `#NUM!` is returned.

[*Example*:

`YEAR(DATE(2006,1,2))` results in `2006`
`YEAR(DATE(2006,0,2))` results in `2005`
`YEAR("2006/1/2 10:45 AM")` results in `2006`
`YEAR(30000)` results in 1982 for the 1900 date system, or 1986 for the 1904 date system

end example]

18.17.7.352 YEARFRAC

Syntax:

YEARFRAC (*start-date* , *end-date* [, *basis*])

Description: Computes the fractional number of years represented by the number of whole days between two dates, *start-date* and *end-date*., according to *basis*.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>start-date</td><td>number</td><td>The period's starting date. start-date can be earlier than, the same as, or later than end-date.</td></tr>
<tr><td>end-date</td><td>number</td><td>The period's ending date.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td><code>0</code> or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months</td></tr>
</table></td></tr>
</table>

Name	Type	Description
		<table><tr><td></td><td>with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr><tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr><tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr><tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr><tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments: If the date is 28 or 29 February, it </td></tr></table>

Name	Type	Description		
				is adjusted to 30 February. • For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.

All arguments are truncated to integers.

Return Type and Value: number – The fractional number of years represented by the number of whole days between two dates, *start-date* and *end-date*., according to *basis*. If the Actual/actual basis is used, the year length used is the average length of the years that the range crosses, regardless of where *start-date* and *end-date* fall in their respective years.

However, if the value of *basis* is out of range, `#NUM!` is returned.

[*Example*:

`YEARFRAC(DATE(2006,1,1),DATE(2006,3,26))` results in `0.236111111`
`YEARFRAC(DATE(2006,3,26),DATE(2006,1,1))` results in `0.236111111`
`YEARFRAC(DATE(2006,1,1),DATE(2006,7,1))` results in `0.5`
`YEARFRAC(DATE(2006,1,1),DATE(2007,9,1))` results in `1.666666667`
`YEARFRAC(DATE(2006,1,1),DATE(2006,7,1),0)` results in `0.5`
`YEARFRAC(DATE(2006,1,1),DATE(2006,7,1),1)` results in `0.495890411`
`YEARFRAC(DATE(2006,1,1),DATE(2006,7,1),2)` results in `0.502777778`
`YEARFRAC(DATE(2006,1,1),DATE(2006,7,1),3)` results in `0.495890411`
`YEARFRAC(DATE(2006,1,1),DATE(2006,7,1),4)` results in `0.5`
`YEARFRAC(DATE(2004,3,1),DATE(2006,3,1),1)` results in 1.998175 (because 2004 is a leap year and Actual/actual basis is used, the average year length is 365.3333)

end example]

18.17.7.353 YIELD

Syntax:

`YIELD ( settlement , maturity , rate , pr , redemption , frequency [ , [ basis ]] )`

Description: Computes the yield on a security that pays periodic interest.

Mathematical Formula:

If there is one coupon period or less until redemption, `YIELD` is calculated as follows:

$$YIELD = \frac{\left(\frac{redempion}{100} + \frac{rate}{frequency}\right) - \left(\frac{par}{100} + \left(\frac{A}{E} \times \frac{rate}{frequency}\right)\right)}{\frac{par}{100} + \left(\frac{A}{E} \times \frac{rate}{frequency}\right)} \times \frac{frequency \times E}{DSR}$$

where:

- A = number of days from the beginning of the coupon period to the settlement date (accrued days).
- DSR = number of days from the settlement date to the redemption date.
- E = number of days in the coupon period.
- $frequency$ = argument $frequency$
- par = argument pr
- $rate$ = argument $rate$
- $redemption$ = argument $redemption$

If there is more than one coupon period until redemption, `YIELD` is calculated through some number of iterations. The resolution uses the Newton method, based on the formula used for the function `PRICE`. The yield is changed until the estimated price given the yield is close to price.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.
maturity	number	The security's maturity date.
rate	number	The security's interest rate.
pr	number	The security's price.
redemption	number	The security's redemption value per $100 face value.
frequency	number	the number of coupon payments per year. (For annual payments, *frequency* is 1; for semiannual payments, *frequency* is 2; for quarterly payments, *frequency* is 4.) *frequency* is truncated to an integer.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>360 days per year.</td></tr>
<tr><td></td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr>
</table>
</td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The yield on a security that pays periodic interest.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *rate* < 0, `#NUM!` is returned.

- *pr* or *redemption* ≤ 0, `#NUM!` is returned.
- *frequency* is any number other than 1, 2, or 4, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`YIELD(DATE(2008,2,15),DATE(2016,11,15),0.0575,95.04287,100,2,0)` results in `6.5000%`

end example]

18.17.7.354 YIELDDISC

Syntax:

`YIELDDISC (` *settlement* `,` *maturity* `,` *pr* `,` *redemption* `[ , [` *basis* `]] )`

Description: Computes the annual yield for a discounted security.

Arguments:

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>settlement</td><td>number</td><td>The security's settlement date.</td></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>pr</td><td>number</td><td>The security's price.</td></tr>
<tr><td>redemption</td><td>number</td><td>The security's redemption value per $100 face value.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td><code>0</code> or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:

If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date
</td></tr>
</table></td></tr>
</table>

Name	Type	Description
		<table><tr><td></td><td>is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr><tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the year is 366 days; otherwise it is 365 days.</td></tr><tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr><tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr><tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments: - If the date is 28 or 29 February, it is adjusted to 30 February. - For months </td></tr></table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td><table><tr><td></td><td>with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr></table></td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The annual yield for a discounted security.

However, if

- *settlement* or *maturity* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *pr* or *redemption* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`YIELDDISC(DATE(2008,2,16),DATE(2008,3,1),99.795,100,2)` results in `5.2823%`

end example]

18.17.7.355 YIELDMAT

Syntax:

`YIELDMAT ( `*settlement*` , `*maturity*` , `*issue*` , `*rate*` , `*pr*` [ , [ `*basis*` ]] )`

Description: Computes the annual yield of a security that pays interest at maturity.

Arguments:

Name	Type	Description
settlement	number	The security's settlement date.

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td>maturity</td><td>number</td><td>The security's maturity date.</td></tr>
<tr><td>issue</td><td>number</td><td>The security's issue date.</td></tr>
<tr><td>rate</td><td>number</td><td>The security's interest rate.</td></tr>
<tr><td>pr</td><td>number</td><td>The security's price.</td></tr>
<tr><td>basis</td><td>number</td><td>The truncated integer type of day count basis to use, as follows:
<table>
<tr><th>Value</th><th>Day Count Basis</th></tr>
<tr><td>0 or omitted</td><td>US (NASD) 30/360. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, if the first date has a day value of 31, the date is converted to day 30. If the second date has a day value of 31, it is changed to 30 days as long as the first date was not 28 or 29 February, in which case it does not change.</td></tr>
<tr><td>1</td><td>Actual/actual. The actual number of days between the two dates are counted. If the date range includes the date 29 February, the</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Name</th><th>Type</th><th>Description</th></tr>
<tr><td></td><td></td><td>
<table>
<tr><td></td><td>year is 366 days; otherwise it is 365 days.</td></tr>
<tr><td>2</td><td>Actual/360. Similar to Basis 1, but only has 360 days per year.</td></tr>
<tr><td>3</td><td>Actual/365. Similar to Basis 1, but always has 365 days per year.</td></tr>
<tr><td>4</td><td>European 30/360. The European method for adjusting day counts. Assumes that each month has 30 days and the total number of days in the year is 360 by making the following adjustments:
If the date is 28 or 29 February, it is adjusted to 30 February.
For months with 31 days, all dates with a day value of 31 are changed to day 30, including situations where the first date is 28 or 29 February.</td></tr>
</table>
</td></tr>
</table>

Time information in the date arguments is ignored.

Return Type and Value: number – The annual yield of a security that pays interest at maturity.

However, if

- *settlement*, *maturity*, or *issue* is out of range for the current date system, `#NUM!` is returned.
- *settlement* ≥ *maturity*, `#NUM!` is returned.
- *rate* or *pr* ≤ 0, `#NUM!` is returned.
- *basis* < 0 or *basis* > 4, `#NUM!` is returned.

[*Example*:

`YIELDMAT(DATE(2008,3,15),DATE(2008,11,3),DATE(2007,11,8),0.0625,`
`100.0123,0)` results in `6.0954%`

end example]

18.17.7.356 ZTEST

Syntax:

`ZTEST ( ` *array* ` , ` *test-value* ` [ , ` *sigma* ` ] )`

Description: Computes the one-tailed probability-value of a z-test. For a given hypothesized population mean, *test-value*, ZTEST returns the probability that the sample mean would be greater than the average of observations in the data set *array*; that is, the observed sample mean.

Mathematical Formula:

When *sigma* is present:

$$ZTEST(array, \mu_0) = 1 - NORMSDIST\left((\bar{x} - \mu_0)/(sigma/\sqrt{n})\right)$$

When *sigma* is omitted:

$$ZTEST(array, \mu_0) = 1 - NORMSDIST\left((\bar{x} - \mu_0)/(s/\sqrt{n})\right)$$

where:

- n = the number of observations in the sample `COUNT(`*array*`)`
- s = the sample standard deviation `STDEV(`*array*`)`
- x = a sample value
- $\bar{x}$ = the sample mean `AVERAGE(`*array*`)`
- μ_0 = the argument *test-value*

Arguments:

Name	Type	Description
array	array	The set of numerical data against which to test *test-value*.

Name	Type	Description
test-value	number	The number to test.
sigma	number	The number is the population (known) standard deviation. If omitted, the sample standard deviation is used.

Return Type and Value: number – The one-tailed probability-value of a z-test.

However, if *array* is empty, the return value is unspecified.

[*Example*:

`ZTEST({3,6,7,8,6,5,4,2,1,9},4)` results in `0.090574197`
`ZTEST({3,6,7,8,6,5,4,2,1,9},6)` results in `0.863043389`

end example]

18.18 Simple Types

This is the complete list of simple types dedicated to SpreadsheetML.

18.18.1 ST_Axis (PivotTable Axis)

This simple type defines the axes for a PivotTable selection.

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
axisCol (Column Axis)	Column axis
axisPage (Include Count Filter)	Page axis
axisRow (Row Axis)	Row axis
axisValues (Values Axis)	Values axis

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Axis) is located in §A.2. *end note*]

18.18.2 ST_BorderId (Border Id)

Zero-based index of the border record used by this cell format.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

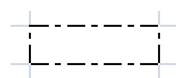
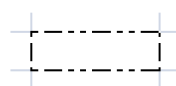
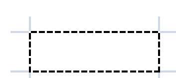
[*Note*: The W3C XML Schema definition of this simple type's content model (ST_BorderId) is located in §A.2. *end note*]

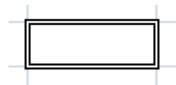
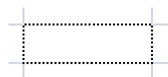
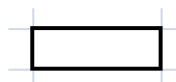
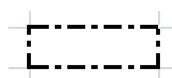
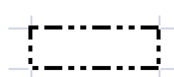
18.18.3 ST_BorderStyle (Border Line Styles)

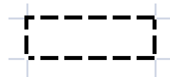
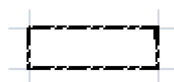
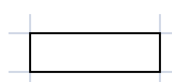
The line style of a border in a cell.

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
dashDot (Dash Dot)	The line style of a border is dash-dot. [*Example*:  *end example*]
dashDotDot (Dash Dot Dot)	The line style of a border is dash-dot-dot. [*Example*:  *end example*]
dashed (Dashed)	The line style of a border is dashed. [*Example*:  *end example*]
dotted (Dotted)	The line style of a border is dotted. [*Example*:  *end example*]
double (Double Line)	The line style of a border is double line.

Enumeration Value	Description
	[*Example*:  *end example*]
hair (Hairline Border)	The line style of a border is hairline. [*Example*:  *end example*]
medium (Medium Border)	The line style of a border is medium. [*Example*:  *end example*]
mediumDashDot (Medium Dash Dot)	The line style of a border is medium dash-dot. [*Example*:  *end example*]
mediumDashDotDot (Medium Dash Dot Dot)	The line style of a border is medium dash-dot-dot. [*Example*:  *end example*]
mediumDashed (Medium Dashed)	The line style of a border is medium dashed. [*Example*:

Enumeration Value	Description
	 *end example*]
none (None)	The line style of a border is none (no border visible). [*Example*: *end example*]
slantDashDot (Slant Dash Dot)	The line style of a border is slant-dash-dot. [*Example*:  *end example*]
thick (Thick Line Border)	The line style of a border is 'thick'. [*Example*:  *end example*]
thin (Thin Border)	The line style of a border is thin. [*Example*:  *end example*]

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_BorderStyle) is located in §A.2. *end note*]

18.18.4 ST_CalcMode (Calculation Mode)

This simple type defines the supported modes for performing calculations on workbook data.

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 (Automatic)	Indicates that calculations in the workbook are performed automatically when cell values change. The application recalculates those cells that are dependent on other cells that contain changed values. This mode of calculation helps to avoid unnecessary calculations.
autoNoTable (Automatic Calculation (No Tables))	Indicates tables be excluded during automatic calculation.
manual (Manual Calculation Mode)	Indicates that calculations in the workbook be triggered manually by the user. For example, the application might expose a command in the user interface.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CalcMode) is located in §A.2. *end note*]

18.18.5 ST_CellComments (Cell Comments)

These enumerations specify how cell comments shall be displayed for paper printing purposes.

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
asDisplayed (Print Comments As Displayed)	Print cell comments as displayed.
atEnd (Print At End)	Print cell comments at end of document.
none (None)	Do not print cell comments.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CellComments) is located in §A.2. *end note*]

18.18.6 ST_CellFormulaType (Formula Type)

Indicates the type of formula in the cell.

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
array (Array Formula)	Formula is an array formula.

Enumeration Value	Description
dataTable (Table Formula)	Formula is a data table formula.
normal (Normal)	Formula is a regular cell formula.
shared (Shared Formula)	Formula is part of a shared formula.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CellFormulaType) is located in §A.2. *end note*]

18.18.7 ST_CellRef (Cell Reference)

Represents a single cell reference in a SpreadsheetML document.

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_CellRef) is located in §A.2. *end note*]

18.18.8 ST_CellSpan (Cell Span Type)

A single cell span item.

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_CellSpan) is located in §A.2. *end note*]

18.18.9 ST_CellSpans (Cell Spans)

List of the cell spans of the item.

This simple type allows a list of items of the ST_CellSpan simple type (§18.18.8).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CellSpans) is located in §A.2. *end note*]

18.18.10 ST_CellStyleXfId (Cell Style Format Id)

Used by xf records and cellStyle records to reference xf records defined in the cellStyleXfs collection.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CellStyleXfId) is located in §A.2. *end note*]

18.18.11 ST_CellType (Cell Type)

Indicates the cell's data type.

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
b (Boolean)	Cell containing a boolean.
d (Date)	Cell contains a date in the ISO 8601 format.
e (Error)	Cell containing an error.
inlineStr (Inline String)	Cell containing an (inline) rich string, i.e., one not in the shared string table. If this cell type is used, then the cell value is in the is element rather than the v element in the cell (c element).
n (Number)	Cell containing a number.
s (Shared String)	Cell containing a shared string.
str (String)	Cell containing a formula string.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CellType) is located in §A.2. *end note*]

18.18.12 ST_CfType (Conditional Format Type)

Conditional format rule type.

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
aboveAverage (Above or Below Average)	This conditional formatting rule highlights cells that are above or below the average for all values in the range.
beginsWith (Begins With)	This conditional formatting rule highlights cells in the range that begin with the given text. Equivalent to using the LEFT() sheet function and comparing values.
cellIs (Cell Is)	This conditional formatting rule compares a cell value to a formula calculated result, using an operator.
colorScale (Color Scale)	This conditional formatting rule creates a gradated color scale on the cells.
containsBlanks (Contains Blanks)	This conditional formatting rule highlights cells that are completely blank. Equivalent of using LEN(TRIM()). This means that if the cell contains only characters that TRIM() would remove, then it is considered blank. An empty cell is also considered blank.

Enumeration Value	Description
containsErrors (Contains Errors)	This conditional formatting rule highlights cells with formula errors. Equivalent to using ISERROR() sheet function to determine if there is a formula error.
containsText (Contains Text)	This conditional formatting rule highlights cells containing given text. Equivalent to using the SEARCH() sheet function to determine whether the cell contains the text.
dataBar (Data Bar)	This conditional formatting rule displays a gradated data bar in the range of cells.
duplicateValues (Duplicate Values)	This conditional formatting rule highlights duplicated values.
endsWith (Ends With)	This conditional formatting rule highlights cells ending with given text. Equivalent to using the RIGHT() sheet function and comparing values.
expression (Expression)	This conditional formatting rule contains a formula to evaluate. When the formula result is true, the cell is highlighted.
iconSet (Icon Set)	This conditional formatting rule applies icons to cells according to their values.
notContainsBlanks (Contains No Blanks)	This conditional formatting rule highlights cells that are not blank. Equivalent of using LEN(TRIM()). This means that if the cell contains only characters that TRIM() would remove, then it is considered blank. An empty cell is also considered blank.
notContainsErrors (Contains No Errors)	This conditional formatting rule highlights cells without formula errors. Equivalent to using ISERROR() sheet function to determine if there is a formula error.
notContainsText (Does Not Contain Text)	This conditional formatting rule highlights cells that do not contain given text. Equivalent to using the SEARCH() sheet function.
timePeriod (Time Period)	This conditional formatting rule highlights cells containing dates in the specified time period. The underlying value of the cell is evaluated, therefore the cell does not need to be formatted as a date to be evaluated. For example, with a cell containing the value 38913 the conditional format shall be applied if the rule requires a value of 7/14/2006.
top10 (Top 10)	This conditional formatting rule highlights cells whose values fall in the top N or bottom N bracket, as specified.
uniqueValues (Unique Values)	This conditional formatting rule highlights unique values in the range.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CfType) is located in §A.2. *end note*]

18.18.13 ST_CfvoType (Conditional Format Value Object Type)

This simple type expresses the type of the conditional formatting value object (cfvo). In general the cfvo specifies one value used in the gradated scale (max, min, midpoint, etc).

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
formula (Formula)	The minimum/ midpoint / maximum value for the gradient is determined by a formula.
max (Maximum)	Indicates that the maximum value in the range shall be used as the maximum value for the gradient.
min (Minimum)	Indicates that the minimum value in the range shall be used as the minimum value for the gradient.
num (Number)	Indicates that the minimum / midpoint / maximum value for the gradient is specified by a constant numeric value.
percent (Percent)	Value indicates a percentage between the minimum and maximum values in the range shall be used as the minimum / midpoint / maximum value for the gradient.
percentile (Percentile)	Value indicates a percentile ranking in the range shall be used as the minimum / midpoint / maximum value for the gradient.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CfvoType) is located in §A.2. *end note*]

18.18.14 ST_Comments (Comment Display Types)

This simple type defines options for displaying comments in the user interface.

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
commIndAndComment (Show Comment & Indicator)	Indicates that both the comment indicator and comment text be show in the user interface.

Enumeration Value	Description
commIndicator (Show Comment Indicator)	Indicates that only the comment indicator be shown in the user interface.
commNone (No Comments)	Indicates that comments not be shown in the user interface.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Comments) is located in §A.2. *end note*]

18.18.15 ST_ConditionalFormattingOperator (Conditional Format Operators)

These conditional format operators are used for "Highlight Cells That Contain..." rules. For example, "highlight cells that begin with "M2" and contain "Mountain Gear"".

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
beginsWith (Begins With)	'Begins with' operator
between (Between)	'Between' operator
containsText (Contains)	'Contains' operator
endsWith (Ends With)	'Ends with' operator
equal (Equal)	'Equal to' operator
greaterThan (Greater Than)	'Greater than' operator
greaterThanOrEqual (Greater Than Or Equal)	'Greater than or equal to' operator
lessThan (Less Than)	'Less than' operator
lessThanOrEqual (Less Than Or Equal)	'Less than or equal to' operator
notBetween (Not Between)	'Not between' operator
notContains (Does Not Contain)	'Does not contain' operator
notEqual (Not Equal)	'Not equal to' operator

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_ConditionalFormattingOperator) is located in §A.2. *end note*]

18.18.16 ST_CredMethod (Credentials Method)

Credentials method used for server access.

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
integrated (Integrated Authentication)	Integrated authentication.
none (No Credentials)	Use no credentials at all.
prompt (Prompt Credentials)	Prompt for credentials.
stored (Stored Credentials)	Use stored credentials.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_CredMethod) is located in §A.2. *end note*]

18.18.17 ST_DataConsolidateFunction (Data Consolidation Functions)

Data consolidation functions specified by the user and used to consolidate ranges of data.

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
average (Average)	The average of the values.
count (Count)	The number of data values. The Count consolidation function works the same as the COUNTA worksheet function.
countNums (CountNums)	The number of data values that are numbers. The Count Nums consolidation function works the same as the COUNT worksheet function.
max (Maximum)	The largest value.
min (Minimum)	The smallest value.
product (Product)	The product of the values.
stdDev (StdDev)	An estimate of the standard deviation of a population, where the sample is a subset of the entire population.
stdDevp (StdDevP)	The standard deviation of a population, where the population is all of the data to be summarized.
sum (Sum)	The sum of the values.
var (Variance)	An estimate of the variance of a population, where the sample is a subset of the entire population.
varp (VarP)	The variance of a population, where the population is all of the data to be summarized.

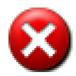
[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DataConsolidateFunction) is located in §A.2. *end note*]

18.18.18 ST_DataValidationErrorStyle (Data Validation Error Styles)

The style of data validation error alert.

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
information (Information Icon)	This data validation error style uses an information icon in the error alert. 
stop (Stop Icon)	This data validation error style uses a stop icon in the error alert. 
warning (Warning Icon)	This data validation error style uses a warning icon in the error alert. 

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DataValidationErrorStyle) is located in §A.2. *end note*]

18.18.19 ST_DataValidationImeMode (Data Validation IME Mode)

These values specify that the IME (input method editor) mode is controlled by data validation.

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
disabled (Disabled IME Mode)	IME mode is disabled. Forces the IME control to be disabled when this cell is selected.
fullAlpha (Full-Width Alpha-Numeric IME Mode)	Forces the IME control to be on and in full-width alph-numeric input mode when the cell is first selected.
fullHangul (Full Width Hangul)	Forces the IME control to be on and in full-width Hangul input mode when first selecting the cell. Applies when the application's language is Korean and

Enumeration Value	Description
	a Korean IME control is selected.
fullKatakana (Full Katakana IME Mode)	Forces the IME control to be on and in full-width Katakana input mode when first selecting the cell. Applies when the application's language is Japanese and a Japanese IME control is selected.
halfAlpha (Half Alpha IME)	Forces the IME control to be on and in half-width alph-numeric input mode when the cell is first selected.
halfHangul (Half-Width Hangul IME Mode)	Forces the IME control to be on and in half-width Hangul input mode when first selecting the cell. Applies when the application's language is Korean and a Korean IME control is selected.
halfKatakana (Half-Width Katakana)	Forces the IME control to be on and in half-width Katakana input mode when first selecting the cell. Applies when the application's language is Japanese and a Japanese IME control is selected.
hiragana (Hiragana IME Mode)	Forces the IME control to be on and in Hiragana input mode when first selecting the cell. Applies when the application's language is Japanese and a Japanese IME control is selected
noControl (IME Mode Not Controlled)	Data validation does not control the IME control's mode.
off (IME Off)	Forces the IME control to be off when first selecting the cell (goes to direct cell input mode).
on (IME On)	Forces the IME control to be on when first selecting the cell.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DataValidationImeMode) is located in §A.2. *end note*]

18.18.20 ST_DataValidationOperator (Data Validation Operator)

The relational operator used in data validation.

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
between (Between)	Data validation which checks if a value is between two other values.
equal (Equal)	Data validation which checks if a value is equal to a specified value.

Enumeration Value	Description
greaterThan (Greater Than)	Data validation which checks if a value is greater than a specified value.
greaterThanOrEqual (Greater Than Or Equal)	Data validation which checks if a value is greater than or equal to a specified value.
lessThan (Less Than)	Data validation which checks if a value is less than a specified value.
lessThanOrEqual (Less Than Or Equal)	Data validation which checks if a value is less than or equal to a specified value.
notBetween (Not Between)	Data validation which checks if a value is not between two other values.
notEqual (Not Equal)	Data validation which checks if a value is not equal to a specified value.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DataValidationOperator) is located in §A.2. *end note*]

18.18.21 ST_DataValidationType (Data Validation Type)

Specifies the type of data validation used to validate user input.

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
custom (Custom)	Data validation which uses a custom formula to check the cell value.
date (Date)	Data validation which checks for date values satisfying the given condition.
decimal (Decimal)	Data validation which checks for decimal values satisfying the given condition.
list (List)	Data validation which checks for a value matching one of list of values.
none (None)	No data validation.
textLength (Text Length)	Data validation which checks for text values, whose length satisfies the given condition.
time (Time)	Data validation which checks for time values satisfying the given condition.
whole (Whole Number)	Data validation which checks for whole number values satisfying the given condition.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DataValidationType) is located in §A.2. *end note*]

18.18.22 ST_DateTimeGrouping (Date Time Grouping)

Specifies how to group dateTime values.

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
day (Day)	Group by day
hour (Group by Hour)	Group by hour
minute (Group by Minute)	Group by minute
month (Month)	Group by month
second (Second)	Group by second
year (Group by Year)	Group by year

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DateTimeGrouping) is located in §A.2. *end note*]

18.18.23 ST_DdeValueType (DDE Value Types)

This simple type indicates the type of the DDE value.

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
b (Boolean)	Indicates that the value is a boolean.
e (Error)	Indicates that the value is an error.
n (Real Number)	Indicates that the value is a real number.
nil (Nil)	Indicates that the value is nil.
str (String)	Indicates that the value is a string.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DdeValueType) is located in §A.2. *end note*]

18.18.24 ST_DvAspect (Data View Aspect Type)

Specifies the desired data or view aspect of the object when drawing or getting data.

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
DVASPECT_CONTENT (Object Display Content)	Provides a representation of an object so it can be displayed as an embedded object inside of a container.
DVASPECT_ICON (Object Display Icon)	Provides an iconic representation of an object.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DvAspect) is located in §A.2. *end note*]

18.18.25 ST_DxfId (Format Id)

This simple type defines the identifier to CT_Dxfs in the styles part. This a zero-based index. See §18.8.30 in Style for more information on formats.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DxfId) is located in §A.2. *end note*]

18.18.26 ST_DynamicFilterType (Dynamic Filter)

These are the dynamic filter types. A dynamic filter returns a result set which might vary due to a change in the data itself or a change in the date on which the filter is being applied. For example, for a set of data {1,1,2,3}, the aboveAverage filter would return or highlight the last two values in the set. If the data is refreshed or changed to {1,1,1,2}, then only the last value would be highlighted. Similarly, the meaning of "lastQuarter" shall be the same for the dates in January, February, and March, but shall change meaning once the date advances from March to April.

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
aboveAverage (Above Average)	Shows values that are above average.
belowAverage (Below Average)	Shows values that are below average.
lastMonth (Last Month)	Shows last month's dates.
lastQuarter (Last Quarter)	Shows last calendar quarter's dates.
lastWeek (Last Week)	Shows last week's dates, using Sunday as the first weekday.
lastYear (Last Year)	Shows last year's dates.
M1 (1st Month)	Shows the dates that are in January, regardless of year.

Enumeration Value	Description
M10 (10th Month)	Shows the dates that are in October, regardless of year.
M11 (11th Month)	Shows the dates that are in November, regardless of year.
M12 (12th Month)	Shows the dates that are in December, regardless of year.
M2 (2nd Month)	Shows the dates that are in February, regardless of year.
M3 (3rd Month)	Shows the dates that are in March, regardless of year.
M4 (4th Month)	Shows the dates that are in April, regardless of year.
M5 (5th Month)	Shows the dates that are in May, regardless of year.
M6 (6th Month)	Shows the dates that are in June, regardless of year.
M7 (7th Month)	Shows the dates that are in July, regardless of year.
M8 (8th Month)	Shows the dates that are in August, regardless of year.
M9 (9th Month)	Shows the dates that are in September, regardless of year.
nextMonth (Next Month)	Shows next month's dates.
nextQuarter (Next Quarter)	Shows next calendar quarter's dates.
nextWeek (Next Week)	Shows next week's dates, using Sunday as the first weekday.
nextYear (Next Year)	Shows next year's dates.
null (Null)	Common filter type not available.
Q1 (1st Quarter)	Shows the dates that are in the 1st calendar quarter, regardless of year.
Q2 (2nd Quarter)	Shows the dates that are in the 2nd calendar quarter, regardless of year.
Q3 (3rd Quarter)	Shows the dates that are in the 3rd calendar quarter, regardless of year.
Q4 (4th Quarter)	Shows the dates that are in the 4th calendar quarter, regardless of year.
thisMonth (This Month)	Shows this month's dates.
thisQuarter (This Quarter)	Shows this calendar quarter's dates.
thisWeek (This Week)	Shows this week's dates, using Sunday as the first weekday.
thisYear (This Year)	Shows this year's dates.
today (Today)	Shows today's dates.
tomorrow (Tomorrow)	Shows tomorrow's dates.

Enumeration Value	Description
yearToDate (Year To Date)	Shows the dates between the beginning of the year and today, inclusive.
yesterday (Yesterday)	Shows yesterday's dates.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_DynamicFilterType) is located in §A.2. *end note*]

18.18.27 ST_ExternalConnectionType (Text Field Datatype)

These are the possible data types to use when importing text into the SpreadsheetML document. Strings are converted to these data types in the worksheet.

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
DMY (Day Month Year)	Field contains a date in the order: day, month, year.
DYM (Day Year Month)	Field contains a date in the order: day, year, month.
EMD (East Asian Year Month Day)	Field contains an East Asian date in the order: EA era year, month, day.
general (General)	The SpreadsheetML application decides the best fit data type based on the content.
MDY (Month Day Year)	Field contains a date in the order: month, day, year.
MYD (Month Day Year)	Field contains a date in the order: month, year, day.
skip (Skip Field)	Don't import this field at all.
text (Text)	Field contains text.
YDM (Year Day Month)	Field contains a date in the order: year, day, month.
YMD (Year Month Day)	Field contains a date in the order: year, month, day.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_ExternalConnectionType) is located in §A.2. *end note*]

18.18.28 ST_FieldSortType (Field Sort Type)

This simple type defines the sort orders that can be applied to fields in a PivotTable.

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
ascending (Ascending)	Indicates the field is sorted in ascending order.
descending (Descending)	Indicates the field is sorted in descending order.
manual (Manual Sort)	Indicates the field is sorted manually.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FieldSortType) is located in §A.2. *end note*]

18.18.29 ST_FileType (File Type)

The file type being used for text import.

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
dos (DOS)	DOS (PC-8).
lin (Linux)	Linux
mac (Macintosh)	Macintosh.
other (Other Non-Specified Values)	Other non-specified values at the time of writing.
win (Windows (ANSI))	Windows (ANSI).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FileType) is located in §A.2. *end note*]

18.18.30 ST_FillId (Fill Id)

Zero-based index used to reference a fill record.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FillId) is located in §A.2. *end note*]

18.18.31 ST_FilterOperator (Filter Operator)

Operator enumerations for filtering.

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
equal (Equal)	Show results which are equal to criteria.
greaterThan (Greater Than)	Show results which are greater than criteria.
greaterThanOrEqual (Greater Than Or Equal)	Show results which are greater than or equal to criteria.
lessThan (Less Than)	Show results which are less than criteria.
lessThanOrEqual (Less Than Or Equal)	Show results which are less than or equal to criteria.
notEqual (Not Equal)	Show results which are not equal to criteria.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FilterOperator) is located in §A.2. *end note*]

18.18.32 ST_FontId (Font Id)

An integer that represents a zero based index into the <fonts> collection in the style sheet.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FontId) is located in §A.2. *end note*]

18.18.33 ST_FontScheme (Font scheme Styles)

Defines the font scheme, if any, to which this font belongs.

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
major (Major Font)	This font is the major font for this theme.
minor (Minor Font)	This font is the minor font for this theme.
none (None)	This font is not a theme font.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FontScheme) is located in §A.2. *end note*]

18.18.34 ST_FormatAction (PivotTable Format Types)

This simple type defines the type of formats that can be applied to PivotTables.

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
blank (Blank)	Indicates no format is applied to the PivotTable. This value is used when formatting is cleared from already formatted cells in the PivotTable.
drill (Drill Type)	Indicates the PivotTable has drill-through format.
formatting (Formatting)	Indicates the PivotTable has formatting.
formula (Formula Type)	Indicates the PivotTable has formulas.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FormatAction) is located in §A.2. *end note*]

18.18.35 ST_Formula (Formula)

A formula

This simple type's contents are a restriction of the ST_Xstring datatype (§22.9.2.19).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Formula) is located in §A.2. *end note*]

18.18.36 ST_FormulaExpression (Formula Expression Type)

This simple type specifies an expression type that can comprise a formula.

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
area (Area)	Reference to a range of cells.
areaError (Area Error)	Reference to a range of cells that now evaluates to an error.
computedArea (Computed Area)	Computed area reference.
ref (Reference)	Single cell reference.
refError (Reference Is Error)	Single cell reference that now evaluates to an error.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FormulaExpression) is located in §A.2. *end note*]

18.18.37 ST_GradientType (Gradient Type)

Type of gradient fill being used, either linear or path.

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
linear (Linear Gradient)	This gradient fill is of linear gradient type. Linear gradient type means that the transition from one color to the next is along a line (e.g., horizontal, vertical, diagonal, etc.).
path (Path)	This gradient fill is of path gradient type. Path gradient type means the that the boundary of transition from one color to the next is a rectangle, defined by top, bottom, left, and right attributes on the gradientFill element.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_GradientType) is located in §A.2. *end note*]

18.18.38 ST_GroupBy (Values Group By)

This simple type defines types of data grouping that can be performed on a PivotTable.

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
days (Days)	Indicates a grouping on "days" for date values.
hours (Hours)	Indicates a grouping on "hours" for date values.
minutes (Minutes)	Indicates a grouping on "minutes" for date values.
months (Months)	Indicates a grouping on "months" for date values.
quarters (Quarters)	Indicates a grouping on "quarters" for date values.
range (Group By Numeric Ranges)	Indicates a grouping by numeric ranges for numeric values.
seconds (Seconds)	Indicates a grouping on "seconds" for date values.
years (Years)	Indicates a grouping on "years" for date values.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_GroupBy) is located in §A.2. *end note*]

18.18.39 ST_GrowShrinkType (Grow Shrink Type)

This type enumerates behavior patterns for refreshing external data in a query table.

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
insertClear (Insert & Clear On Refresh)	Insert entire rows for new data, clear unused cells.
insertDelete (Insert & Delete On Refresh)	Insert cells for new data, delete unused cells.
overwriteClear (Overwrite & Clear On Refresh)	Overwrite existing cells with new data, clear unused cells.

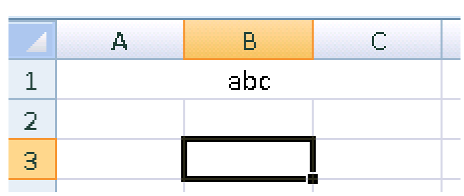
[*Note*: The W3C XML Schema definition of this simple type's content model (ST_GrowShrinkType) is located in §A.2. *end note*]

18.18.40 ST_HorizontalAlignment (Horizontal Alignment Type)

The enumeration value indicating the portion of Cell Alignment in a cell format (XF) that is horizontal alignment, i.e., whether it is aligned general, left, right, horizontally centered, filled (replicated), justified, centered across multiple cells, or distributed.

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 (Centered Horizontal Alignment)	The horizontal alignment is centered, meaning the text is centered across the cell.
centerContinuous (Center Continuous Horizontal Alignment)	The horizontal alignment is centered across multiple cells. The information about how many cells to span is expressed in the Sheet Part, in the row of the cell in question. For each cell that is spanned in the alignment, a cell element needs to be written out, with the same style Id which references the `centerContinuous` alignment. [*Example*: This shows the value of A1 centered across A1:C1:  The XML from the Sheet Part: `<row r="1" spans="1:3">`

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td><c r="A1" s="1" t="s">
  <v>0</v>
</c>
<c r="B1" s="1"/>
<c r="C1" s="1"/>
</row>
The XML from the Styles Part:

<cellXfs count="2">
  <xf numFmtId="0" fontId="0"
    fillId="0" borderId="0" xfId="0"/>
  <xf numFmtId="0" fontId="0"
    fillId="0" borderId="0" xfId="0"
    applyAlignment="1">
    <alignment
      horizontal="centerContinuous"/>
  </xf>
</cellXfs>

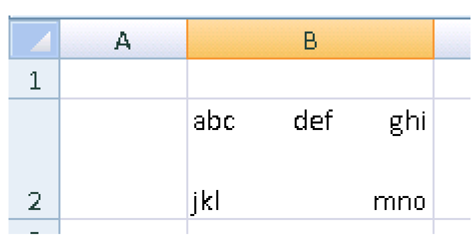
end example]</td></tr>
<tr><td>distributed (Distributed Horizontal Alignment)</td><td>I/ndicates that each 'word' in each line of text inside the cell is evenly distributed across the width of the cell, with flush right and left margins.

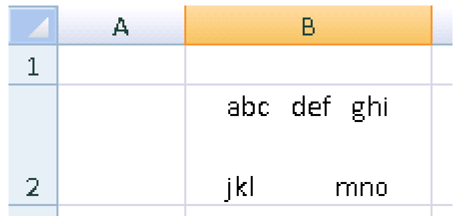
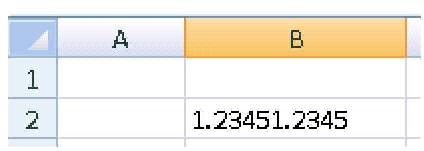
When there is also an indent value to apply, both the left and right side of the cell are padded by the indent value.

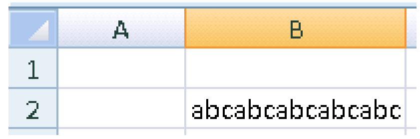
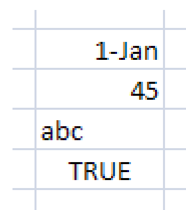
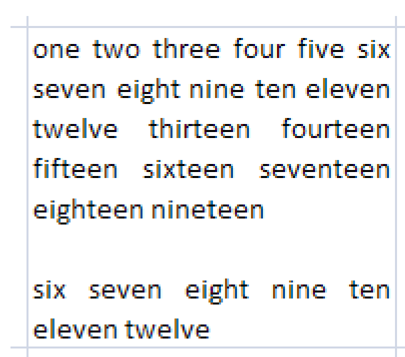
A 'word' is a set of characters with no space character in them.

Two lines inside a cell are separated by a carriage return.

[Example: This shows three lines of text evenly distributed horizontally across the cell. The first line is "abc def ghi", the second line is blank, and the third line is "jkl mno".


</td></tr>
</table>

Enumeration Value	Description
	This shows the same example, with an indent value of 2:  There is no vertical component to the alignment being shown here. The row has been manually adjusted to display the text. *end example*]
fill (Fill)	Indicates that the value of the cell should be filled across the entire width of the cell. If blank cells to the right also have the fill alignment, they are also filled with the value, using a convention similar to `centerContinuous`. Additional rules: • Only whole values can be appended, not partial values. • The column will not be widened to 'best fit' the filled value • If appending an additional occurrence of the value exceeds the boundary of the cell left/right edge, don't append the additional occurrence of the value. • The display value of the cell is filled, not the underlying raw number. [*Example*: This cell is filled with the value 1.2345 and has a width of 15 characters:  This cell is filled with the value abc and has width of 15 characters:

Enumeration Value	Description
	 • *end example*] •
general (General Horizontal Alignment)	The horizontal alignment is general-aligned. Text data is left-aligned. Numbers, dates, and times are right-aligned. Boolean types are centered. Changing the alignment does not change the type of data. [*Example*: These cells are `general` aligned:  *end example*]
justify (Justify)	The horizontal alignment is justified (flush left and right). For each line of text, aligns each line of the wrapped text in a cell to the right and left (except the last line). If no single line of text wraps in the cell, then the text is not justified. [*Example*: There are two lines of text in this cell, and the cell's horizontal alignment is `justify`:  *end example*]
left (Left Horizontal Alignment)	The horizontal alignment is left-aligned, even in Right-

Enumeration Value	Description
	to-Left mode. Aligns contents at the left edge of the cell. If an indent amount is specified, the contents of the cell is indented from the left by the specified number of character spaces. The character spaces are based on the default font and font size for the workbook.
right (Right Horizontal Alignment)	The horizontal alignment is right-aligned, meaning that cell contents are aligned at the right edge of the cell, even in Right-to-Left mode.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_HorizontalAlignment) is located in §A.2. *end note*]

18.18.41 ST_HtmlFmt (HTML Formatting Handling)

How to handle formatting from the HTML source.

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 (All)	Transfer all HTML formatting into the worksheet along with data.
none (No Formatting)	Bring data in as unformatted text (setting data types still occurs).
rtf (Honor Rich Text)	Translate HTML formatting to rich text formatting on the data brought into the worksheet.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_HtmlFmt) is located in §A.2. *end note*]

18.18.42 ST_IconSetType (Icon Set Type)

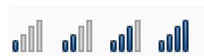
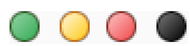
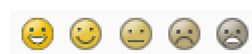
Icon set type for conditional formatting. The threshold values for triggering the different icons within a set are configurable, and the icon order is reversible. See element iconSet for more information.

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
3Arrows (3 Arrows)	3 arrows icon set.

Enumeration Value	Description
	
3ArrowsGray (3 Arrows (Gray))	3 gray arrows icon set. 
3Flags (3 Flags)	3 flags icon set. 
3Signs (3 Signs)	3 signs icon set. 
3Symbols (3 Symbols Circled)	3 symbols icon set. 
3Symbols2 (3 Symbols)	3 Symbols icon set. 
3TrafficLights1 (3 Traffic Lights)	3 traffic lights icon set (#1). 
3TrafficLights2 (3 Traffic Lights Black)	3 traffic lights icon set with thick black border. 
4Arrows (4 Arrows)	4 arrows icon set. 

Enumeration Value	Description
4ArrowsGray (4 Arrows (Gray))	4 gray arrows icon set. 
4Rating (4 Ratings)	4 ratings icon set. 
4RedToBlack (4 Red To Black)	4 'red to black' icon set. 
4TrafficLights (4 Traffic Lights)	4 traffic lights icon set. 
5Arrows (5 Arrows)	5 arrows icon set. 
5ArrowsGray (5 Arrows (Gray))	5 gray arrows icon set. 
5Quarters (5 Quarters)	5 quarters icon set. 
5Rating (5 Ratings Icon Set)	5 rating icon set. 

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_IconSetType) is located in §A.2. *end note*]

18.18.43 ST_ItemType (PivotItem Type)

This simple type defines the pivot type for a pivotItem.

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
avg (Average)	Indicates the pivot item represents an "average" aggregate function.
blank (Blank Pivot Item)	Indicates the pivot item represents a blank line.
count (Count)	Indicates the pivot item represents custom the "count" aggregate."
countA (CountA)	Indicates the pivot item represents the "count numbers" aggregate function.
data (Data)	Indicate the pivot item represents data.
default (Default)	Indicates the pivot item represents the default type for this PivotTable. The default pivot item type is the "total" aggregate function.
grand (Grand Total Item)	Indicates the pivot items represents the grand total line.
max (Max)	Indicates the pivot item represents the "maximum" aggregate function.
min (Min)	Indicates the pivot item represents the "minimum" aggregate function.
product (Product)	Indicates the pivot item represents the "product" function.
stdDev (stdDev)	Indicates the pivot item represents the "standard deviation" aggregate function.
stdDevP (StdDevP)	Indicates the pivot item represents the "standard deviation population" aggregate function.
sum (Sum)	Indicates the pivot item represents the "sum" aggregate value.
var (Var)	Indicates the pivot item represents the "variance" aggregate value.
varP (VarP)	Indicates the pivot item represents the "variance population" aggregate value.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_ItemType) is located in §A.2. *end note*]

18.18.44 ST_MdxFunctionType (MDX Function Type)

This simple type is an enumeration representing different MDX function types.

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
c (Cube Set Count)	CUBESETCOUNT
k (Cube KPI Member)	CUBEKPIMEMBER
m (Cube Member)	CUBEMEMBER
p (Cube Member Property)	CUBEMEMBERPROPERTY
r (Cube Ranked Member)	CUBERANKEDMEMBER
s (Cube Set)	CUBESET
v (Cube Value)	CUBEVALUE

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_MdxFunctionType) is located in §A.2. *end note*]

18.18.45 ST_MdxKPIProperty (MDX KPI Property)

An enumeration representing the different types of KPI properties.

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
g (Goal)	Goal.
m (Current Time Member)	Current time member.
s (Status)	Status.
t (Trend)	Trend.
v (Value)	Value.
w (Weight)	Weight.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_MdxKPIProperty) is located in §A.2. *end note*]

18.18.46 ST_MdxSetOrder (MDX Set Order)

This simple type represents an enumeration specifying an MDX set order.

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
a (Ascending)	Sort ascending.
aa (Alpha Ascending Sort Order)	Sorted alphabetically in ascending order by the caption.
ad (Alpha Descending Sort Order)	Sort in descending order alphabetically by the caption.
d (Descending)	Sort descending.
na (Natural Ascending)	Sorted in ascending order by the natural order of the data - usually by the key. For instance if there is a list of accounts in a general ledger, this might be in order of account number.
nd (Natural Descending)	Sorted in descending order by the natural order of the data - usually by the key. For instance if there is a list of accounts in a general ledger, this might be in order of account number.
u (Unsorted)	Unsorted.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_MdxSetOrder) is located in §A.2. *end note*]

18.18.47 ST_NumFmtId (Number Format Id)

This simple type defines the identifier to a style sheet number format entry in CT_NumFmts. Number formats are written to the styles part. See §18.8.31 for more information on number formats.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_NumFmtId) is located in §A.2. *end note*]

18.18.48 ST_Objects (Object Display Types)

This simple type defines how the application displays objects in this workbook. Objects might include charts, images, and other object data that the application supports.

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 (All)	Indicates that all objects be shown in the workbook.
none (None)	Indicates that all objects be hidden in the workbook.
placeholders (Show Placeholders)	Indicates that the application show placeholders for objects in the workbook.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Objects) is located in §A.2. *end note*]

18.18.49 ST_OleUpdate (OLE Update Types)

Indicates whether the linked object updates the cached data for the linked object automatically or only when the container calls IOleObject::Update or IOleLink::Update methods.

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
OLEUPDATE_ALWAYS (Always Update OLE)	Update the link object whenever possible, this option corresponds to the 'automatic update' option in the Links dialog box.
OLEUPDATE_ONCALL (Update OLE On Call)	Update the link object only when IOleObject::Update or IOleLink::Update is called, this option coresponds to the Manual update option in the Links dialog box.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_OleUpdate) is located in §A.2. *end note*]

18.18.50 ST_Orientation (Orientation)

Print orientation for this sheet.

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)	Orientation not specified, use the default.
landscape (Landscape)	Landscape orientation.
portrait (Portrait)	Portrait orientation.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Orientation) is located in §A.2. *end note*]

18.18.51 ST_PageOrder (Page Order)

Specifies printed page order.

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
downThenOver (Down Then Over)	Order pages vertically first, then move horizontally.
overThenDown (Over Then Down)	Order pages horizontally first, then move vertically

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PageOrder) is located in §A.2. *end note*]

18.18.52 ST_Pane (Pane Types)

Defines the names of the four possible panes into which the view of a workbook in the application can be split.

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
bottomLeft (Bottom Left Pane)	Bottom left pane, when both vertical and horizontal splits are applied. This value is also used when only a horizontal split has been applied, dividing the pane into upper and lower regions. In that case, this value specifies the bottom pane.
bottomRight (Bottom Right Pane)	Bottom right pane, when both vertical and horizontal splits are applied.
topLeft (Top Left Pane)	Top left pane, when both vertical and horizontal splits are applied. This value is also used when only a horizontal split has been applied, dividing the pane into upper and lower regions. In that case, this value specifies the top pane. This value is also used when only a vertical split has been applied, dividing the pane into right and left regions. In that case, this value specifies the left pane

Enumeration Value	Description
topRight (Top Right Pane)	Top right pane, when both vertical and horizontal splits are applied. This value is also used when only a vertical split has been applied, dividing the pane into right and left regions. In that case, this value specifies the right pane.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Pane) is located in §A.2. *end note*]

18.18.53 ST_PaneState (Pane State)

State of the sheet's pane.

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
frozen (Frozen)	Panes are frozen, but were not split being frozen. In this state, when the panes are unfrozen again, a single pane results, with no split. In this state, the split bars are not adjustable.
frozenSplit (Frozen Split)	Panes are frozen and were split before being frozen. In this state, when the panes are unfrozen again, the split remains, but is adjustable.
split (Split)	Panes are split, but not frozen. In this state, the split bars are adjustable by the user.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PaneState) is located in §A.2. *end note*]

18.18.54 ST_ParameterType (Parameter Type)

Parameter Type.

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
cell (Parameter From Cell)	Get the parameter value from a cell on each refresh.

Enumeration Value	Description
prompt (Prompt on Refresh)	Prompt the user on each refresh for a parameter value.
value (Value)	Use a constant value on each refresh for the parameter value.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_ParameterType) is located in §A.2. *end note*]

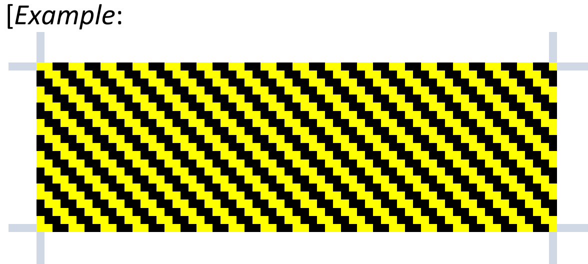
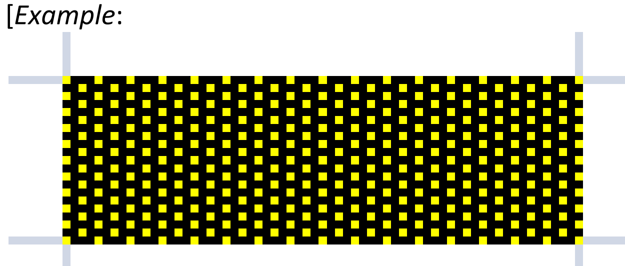
18.18.55 ST_PatternType (Pattern Type)

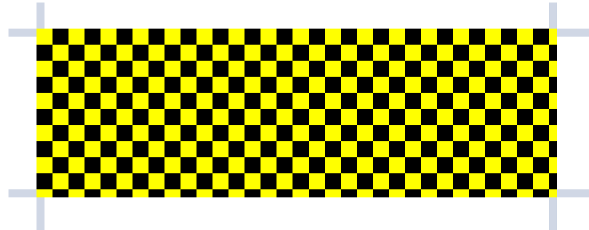
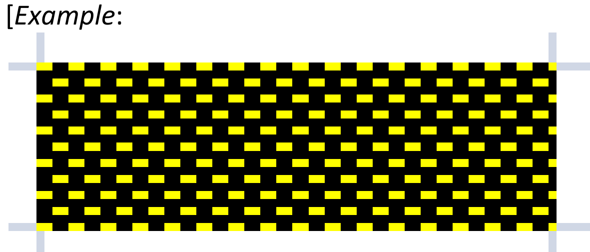
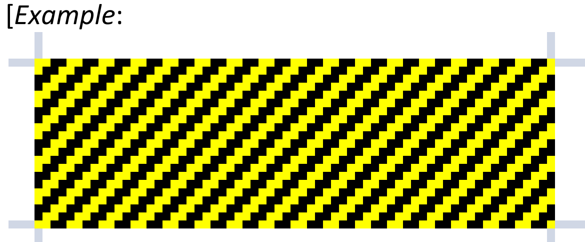
Indicates the style of fill pattern being used for a cell format.

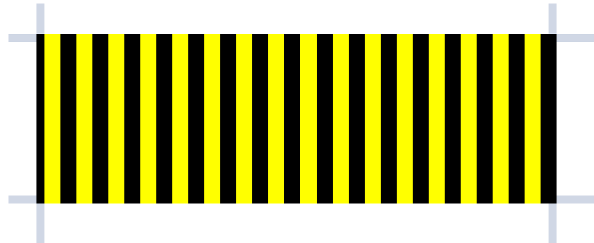
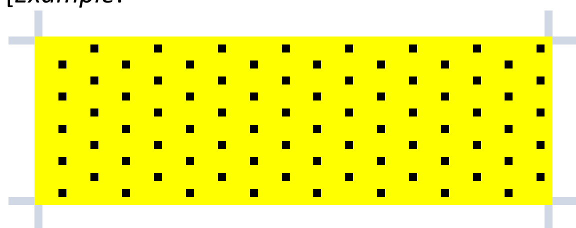
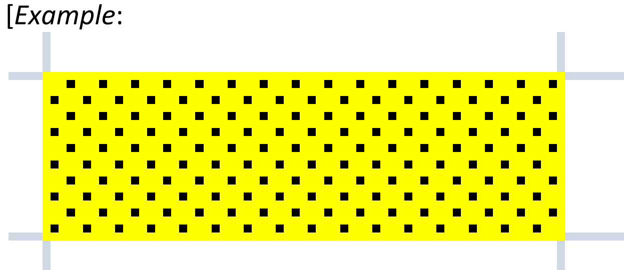
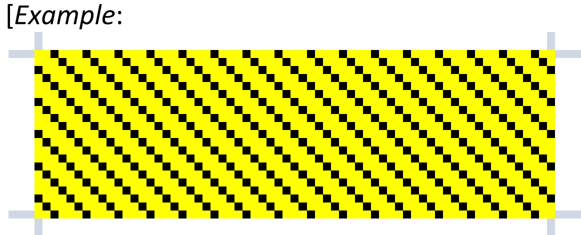
The examples below use yellow background and black foreground colors.

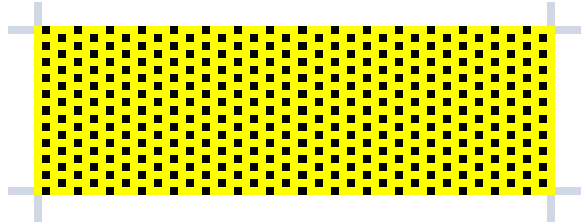
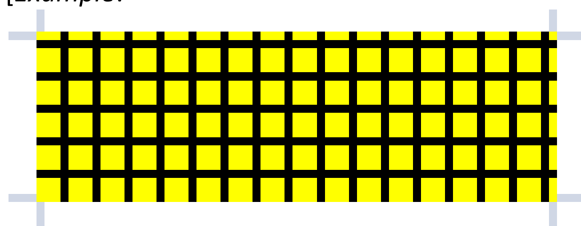
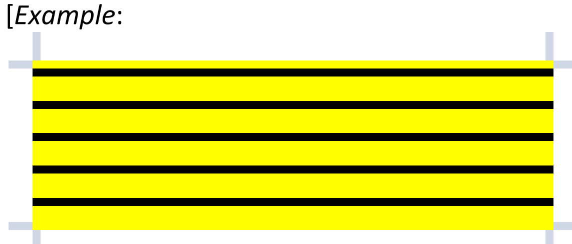
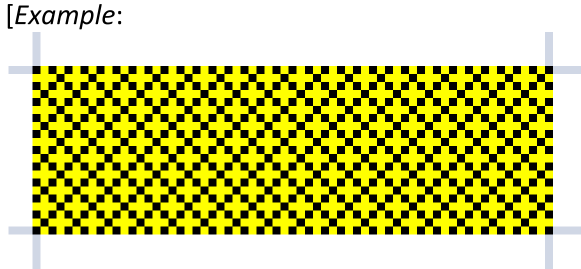
This simple type's contents are a restriction of the W3C XML Schema string datatype.

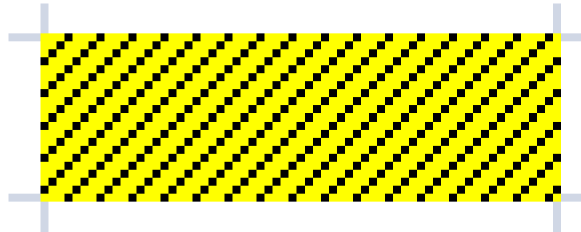
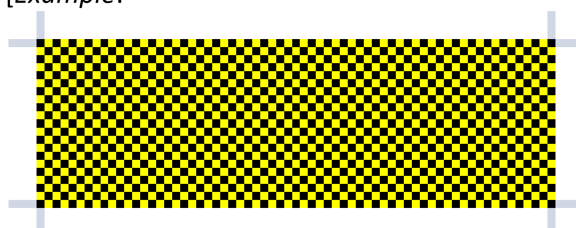
This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
darkDown (Dark Down)	The fill style is 'dark down'. [*Example*:  *end example*]
darkGray (Dary Gray)	The fill style is 'dark gray'. [*Example*:  *end example*]
darkGrid (Dark Grid)	The fill style is 'dark grid'. [*Example*:

Enumeration Value	Description
	 *end example*]
darkHorizontal (Dark Horizontal)	The fill style is dark horizontal. [*Example*:  *end example*]
darkTrellis (Dark Trellis)	The fill style is 'dark trellis'. [*Example*:  *end example*]
darkUp (Dark Up)	The fill style is 'dark up'. [*Example*:  *end example*]
darkVertical (Dark Vertical)	The fill style is 'dark vertical'. [*Example*:

Enumeration Value	Description
	 *end example*]
gray0625 (Gray 0.0625)	The fill style is grayscale of 0.0625 (1/16) value. [*Example*:  *end example*]
gray125 (Gray 0.125)	The fill style is grayscale of 0.125 (1/8) value. [*Example*:  *end example*]
lightDown (Light Down)	The fill style is 'light down'. [*Example*:  *end example*]
lightGray (Light Gray)	The fill style is light gray. [*Example*:

Enumeration Value	Description
	 *end example*]
lightGrid (Light Grid)	The fill style is 'light grid'. [*Example*:  *end example*]
lightHorizontal (Light Horizontal)	The fill style is light horizontal. [*Example*:  *end example*]
lightTrellis (Light Trellis)	The fill style is 'light trellis'. [*Example*:  *end example*]
lightUp (Light Up)	The fill style is light up. [*Example*:

Enumeration Value	Description
	 *end example*]
lightVertical (Light Vertical)	The fill style is light vertical.
mediumGray (Medium Gray)	The fill style is medium gray. [*Example*:  *end example*]
none (None)	The fill style is none (no fill). When foreground and/or background colors are specified, a pattern of 'none' overrides and means the cell has no fill. [*Example*:  *end example*]
solid (Solid)	The fill style is solid. When solid is specified, the foreground color (fgColor) is the only color rendered, even when a background color (bgColor) is also specified. [*Example*:

Enumeration Value	Description
	end example]

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PatternType) is located in §A.2. *end note*]

18.18.56 ST_PhoneticAlignment (Phonetic Alignment Types)

Phonetic alignment settings. These specify how to align the phonetic text, which represent the sounds, above the base text or base word.

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 Alignment)	Center the phonetic characters over the base word, per word.
distributed (Distributed)	Each phonetic character is distributed above each base word character, per word.
left (Left Alignment)	Each phonetic character is left justified with respect to the base text., per word.
noControl (No Control)	Each phonetic character is left justified without respect to the base text (so it is not per word).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PhoneticAlignment) is located in §A.2. *end note*]

18.18.57 ST_PhoneticType (Phonetic Type)

Represents the different East Asian character sets that shall be used for displaying phonetic hints.

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
fullwidthKatakana (Full-Width Katakana)	Full-width Katakana is used

Enumeration Value	Description
halfwidthKatakana (Half-Width Katakana)	Half-width Katakana is used, this is the same Katakana character set, just half as wide so it takes up less space.
Hiragana (Hiragana)	Hiragana is used
noConversion (No Conversion)	Any characters are allowed. In this case the spreadsheet application shall leave the text as entered.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PhoneticType) is located in §A.2. *end note*]

18.18.58 ST_PivotAreaType (Rule Type)

Indicates the type of rule being used to describe an area or aspect of the PivotTable.

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 (All)	Refers to the whole PivotTable.
button (Field Button)	Refers to a field button.
data (Data)	Refers to something in the data area.
none (None)	Refers to no Pivot area.
normal (Normal)	Refers to a header or item.
origin (Origin)	Refers to the blank cells at the top-left of the PivotTable (top-left to LTR sheets, top-right for RTL sheets).
topEnd (Top End)	Refers to the blank cells at the top of the PivotTable, on its trailing edge (top-right for LTR sheets, top-left for RTL sheets).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PivotAreaType) is located in §A.2. *end note*]

18.18.59 ST_PivotFilterType (Pivot Filter Types)

This simple type defines filters that can be applied to PivotTables.

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
captionBeginsWith (Caption Begins With)	Indicates the "begins with" filter for field captions.
captionBetween (Caption Is Between)	Indicates the "is between" filter for field captions.
captionContains (Caption Contains)	Indicates the "contains" filter for field captions.
captionEndsWith (Caption Ends With)	Indicates the "ends with" filter for field captions.
captionEqual (Caption Equals)	Indicates the "equal" filter for field captions.
captionGreaterThan (Caption Is Greater Than)	Indicates the "is greater than" filter for field captions.
captionGreaterThanOrEqual (Caption Is Greater Than Or Equal To)	Indicates the "is greater than or equal to" filter for field captions.
captionLessThan (Caption Is Less Than)	Indicates the "is less than" filter for field captions.
captionLessThanOrEqual (Caption Is Less Than Or Equal To)	Indicates the "is less than or equal to" filter for field captions.
captionNotBeginsWith (Caption Does Not Begin With)	Indicates the "does not begin with" filter for field captions.
captionNotBetween (Caption Is Not Between)	Indicates the "is not between" filter for field captions.
captionNotContains (Caption Does Not Contain)	Indicates the "does not contain" filter for field captions.
captionNotEndsWith (Caption Does Not End With)	Indicates the "does not end with" filter for field captions.
captionNotEqual (Caption Not Equal)	Indicates the "not equal" filter for field captions.
count (Count)	Indicates the "count" filter.
dateBetween (Date Between)	Indicates the "between" filter for date values.
dateEqual (Date Equals)	Indicates the "equals" filter for date values.
dateNewerThan (Date Newer Than)	Indicates the "newer than" filter for date values.
dateNewerThanOrEqual (Date Newer Than or Equal To)	Indicates the "newer than or equal to" filter for date values.
dateNotBetween (Date Not Between)	Indicates the "not between" filter for date values.
dateNotEqual (Date Does Not Equal)	Indicates the "does not equal" filter for date values.
dateOlderThan (Date Older Than)	Indicates the "older than" filter for date values.
dateOlderThanOrEqual (Date Older Than Or Equal)	Indicates the "older than or equal to" filter for date values.
lastMonth (Last Month)	Indicates the "last month" filter for date values.
lastQuarter (Last Quarter)	Indicates the "last quarter" filter for date values.
lastWeek (Last Week)	Indicates the "last week" filter for date values.
lastYear (Last Year)	Indicates the "last year" filter for date values.
M1 (January)	Indicates the "January" filter for date values.

Enumeration Value	Description
M10 (Dates in October)	Indicates the "October" filter for date values.
M11 (Dates in November)	Indicates the "November" filter for date values.
M12 (Dates in December)	Indicates the "December" filter for date values.
M2 (Dates in February)	Indicates the "February" filter for date values.
M3 (Dates in March)	Indicates the "March" filter for date values.
M4 (Dates in April)	Indicates the "April" filter for date values.
M5 (Dates in May)	Indicates the "May" filter for date values.
M6 (Dates in June)	Indicates the "June" filter for date values.
M7 (Dates in July)	Indicates the "July" filter for date values.
M8 (Dates in August)	Indicates the "August" filter for date values.
M9 (Dates in September)	Indicates the "September" filter for date values.
nextMonth (Next Month)	Indicates the "next month" filter for date values.
nextQuarter (Next Quarter)	Indicates the "next quarter" for date values.
nextWeek (Next Week)	Indicates the "next week" for date values.
nextYear (Next Year)	Indicates the "next year" filter for date values.
percent (Percent)	Indicates the "percent" filter for numeric values.
Q1 (First Quarter)	Indicates the "first quarter" filter for date values.
Q2 (Second Quarter)	Indicates the "second quarter" filter for date values.
Q3 (Third Quarter)	Indicates the "third quarter" filter for date values.
Q4 (Fourth Quarter)	Indicates the "fourth quarter" filter for date values.
sum (Sum)	Indicates the "sum" filter for numeric values.
thisMonth (This Month)	Indicates the "this month" filter for date values.
thisQuarter (This Quarter)	Indicates the "this quarter" filter for date values.
thisWeek (This Week)	Indicates the "this week" filter for date values.
thisYear (This Year)	Indicate the "this year" filter for date values.
today (Today)	Indicates the "today" filter for date values.
tomorrow (Tomorrow)	Indicates the "tomorrow" filter for date values.
unknown (Unknown)	Indicates the PivotTable filter is unknown to the application.
valueBetween (Value Between)	Indicates the "Value between" filter for text and numeric values.
valueEqual (Value Equal)	Indicates the "value equal" filter for text and numeric values.
valueGreaterThan (Value Greater Than)	Indicates the "value greater than" filter for text and numeric values.

Enumeration Value	Description
valueGreaterThanOrEqual (Value Greater Than Or Equal To)	Indicates the "value greater than or equal to" filter for text and numeric values.
valueLessThan (Value Less Than)	Indicates the "value less than" filter for text and numeric values.
valueLessThanOrEqual (Value Less Than Or Equal To)	Indicates the "value less than or equal to" filter for text and numeric values
valueNotBetween (Value Not Between)	Indicates the "value not between" filter for text and numeric values.
valueNotEqual (Value Not Equal)	Indicates the "value not equal" filter for text and numeric values.
yearToDate (Year-To-Date)	Indicates the "year-to-date" filter for date values.
yesterday (Yesterday)	Indicates the "yesterday" filter for date values.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PivotFilterType) is located in §A.2. *end note*]

18.18.60 ST_PrintError (Print Errors)

This enumeration specifies how to display cells with errors when printing the worksheet.

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
blank (Show Cell Errors As Blank)	Display cell errors as blank.
dash (Dash Cell Errors)	Display cell errors as dashes.
displayed (Display Cell Errors)	Display cell errors as displayed on screen.
NA (NA)	Display cell errors as #N/A.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PrintError) is located in §A.2. *end note*]

18.18.61 ST_Qualifier (Qualifier)

Qualifier to use to denote string data types in when text is imported from an external file.

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
doubleQuote (Double Quote)	Quotation mark -- double quote (").
none (No Text Qualifier)	No text string qualifier used.
singleQuote (Single Quote)	Apostrophe mark -- single quote (').

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Qualifier) is located in §A.2. *end note*]

18.18.62 ST_Ref (Cell References)

This simple type defines a reference to a range of cells within a sheet in the workbook. A reference identifies a cell or a range of cells on a worksheet and tells the application where to look for the values or data you want to use in a formula. With references, you can use data contained in different parts of a worksheet in one formula or use the value from one cell in several formulas. You can also refer to cells on other sheets in the same workbook, and to other workbooks. References to cells in other workbooks are called links.

SpreadsheetML defines two reference styles defined in the ST_RefMode simple type.

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_Ref) is located in §A.2. *end note*]

18.18.63 ST_RefA (Single Cell Reference)

This simple type specifies a single cell reference that might be absolute.

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_RefA) is located in §A.2. *end note*]

18.18.64 ST_RefMode (Reference Mode)

This simple type defines the supported reference styles or modes for a workbook in SpreadsheetML.

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
A1 (A1 Mode)	Indicates that the workbook uses A1 reference style. This is the default for SpreadsheetML. A1 reference style refers to columns with letters and refers to rows with numbers. For example, A1 refers to the cell at the intersection of column A and row 1.

Enumeration Value	Description
R1C1 (R1C1 Reference Mode)	Indicates that the workbook uses the R1C1 reference style. R1C1 reference style refers to both the rows and the columns on the worksheet with numbers. The location of a cell is indicated with an "R" followed by a row number and a "C" followed by a column number. For example, R1C1 refers to the cell at the intersection of row R1 and column C1.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_RefMode) is located in §A.2. *end note*]

18.18.65 ST_RevisionAction (Revision Action Types)

Identifies what kind of action the user performed. Applies to Comment and Custom View revision record.

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
add (Add)	Add action.
delete (Delete)	Delete action.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_RevisionAction) is located in §A.2. *end note*]

18.18.66 ST_rwColActionType (Row Column Action Type)

Identifies what kind of an action was applied to a row or column.

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
deleteCol (Delete Column)	Column delete revision.
deleteRow (Delete Row)	Row delete revision.
insertCol (Column Insert)	Column insert revision.
insertRow (Insert Row)	Row insert revision.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_rwColActionType) is located in §A.2. *end note*]

18.18.67 ST_Scope (Conditional Formatting Scope)

This simple type defines the scope of conditional formatting applied in the PivotTable.

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
data (Data Fields)	Indicates that conditional formatting is applied to the selected data fields.
field (Field Intersections)	Indicates that conditional formatting is applied to the selected PivotTable field intersections.
selection (Selection)	Indicates that conditional formatting is applied to the selected cells.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Scope) is located in §A.2. *end note*]

18.18.68 ST_SheetState (Sheet Visibility Types)

This simple type defines the possible states for sheet visibility.

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
hidden (Hidden)	Indicates the workbook window is hidden, but can be shown by the user via the user interface.
veryHidden (Very Hidden)	Indicates the sheet is hidden and cannot be shown in the user interface (UI). This state is only available programmatically.
visible (Visible)	Indicates the sheet is visible.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SheetState) is located in §A.2. *end note*]

18.18.69 ST_SheetViewType (Sheet View Type)

Defines the view setting of the sheet.

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
normal (Normal View)	Normal view
pageBreakPreview (Page Break Preview)	Page break preview
pageLayout (Page Layout View)	Page Layout View

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SheetViewType) is located in §A.2. *end note*]

18.18.70 ST_ShowDataAs (Show Data As)

This simple type defines the data formats for a field in the PivotTable.

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
difference (Difference)	Indicates the field is shown as the "difference from" a value.
index (Index)	Indicates the field is shown as the "index.
normal (Normal Data Type)	Indicates that the field is shown as its normal data type.
percent (Percentage Of)	Indicates the field is show as the "percentage of
percentDiff (Percentage Difference)	Indicates the field is shown as the "percentage difference from" a value.
percentOfCol (Percent of Column)	Indicates the field is shown as the percentage of column.
percentOfRow (Percentage of Row)	Indicates the field is shown as the percentage of row
percentOfTotal (Percentage of Total)	Indicates the field is shown as percentage of total.
runTotal (Running Total)	Indicates the field is shown as running total in the table.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_ShowDataAs) is located in §A.2. *end note*]

18.18.71 ST_SmartTagShow (Smart Tag Display Types)

This simple type defines options for displaying smart tags in the user interface.

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 (All)	Indicates that smart tags are enabled and shown in the user interface.
noIndicator (No Smart Tag Indicator)	Indicates that the smart tags are enabled but the indicator not be shown in the user interface.
none (None)	Indicates that smart tags are disabled and not displayed in the user interface.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SmartTagShow) is located in §A.2. *end note*]

18.18.72 ST_SortBy (Sort By)

Specifies what to sort by. In many cases a range of cells are sorted by their values. However, cells can also be sorted by their background color, font color, and type of icon in the cell.

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
cellColor (Sort by Cell Color)	Sort by cell color
fontColor (Sort by Font Color)	Sort by font color
icon (Sort by Icon)	Sort by icon
value (Value)	Sort by value

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SortBy) is located in §A.2. *end note*]

18.18.73 ST_SortMethod (Sort Method)

Sort method. Chinese Simplified, Chinese Traditional, and Japanese support alternate sort methods (multiple sort options are available). All other languages support only 1 sort option. In that case, the value `pinYin` is used.

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
none (None)	Not specified, use default sort method.
pinYin (PinYin Sort)	Default sort method. This is the only sort option for most languages.

Enumeration Value	Description
	For Chinese Simplified, Chinese Traditional, and Japanese, `pinYin` means sort by phonetic value. •
stroke (Sort by Stroke)	Sort by stroke count of the characters. Only applies to Chinese Simplified, Chinese Traditional, and Japanese.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SortMethod) is located in §A.2. *end note*]

18.18.74 ST_SortType (Set Sort Order)

This simple type defines the possible sort order for the PivotTable.

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
ascending (Ascending)	Indicates that the PivotTable data is sorted in ascending order.
ascendingAlpha (Ascending Alpha)	Indicates that the PivotTable data is sorted in alphabetic order with ascending values.
ascendingNatural (Ascending Natural)	Indicates that the PivotTable data is sorted in natural order with ascending.
descending (Descending)	Indicates that the PivotTable data is sorted in descending.
descendingAlpha (Alphabetic Order Descending)	Indicates that the PivotTable data is sorted in alphabetic order with descending values.
descendingNatural (Natural Order Descending)	Indicates that the PivotTable data is sorted in natural order with descending values.
none (None)	Indicates that the PivotTable data is not sorted.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SortType) is located in §A.2. *end note*]

18.18.75 ST_SourceType (PivotCache Type)

This simple type defines the cache types for PivotTables.

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
consolidation (Consolidation Ranges)	Indicates that the cache contains data that consolidates ranges.
external (External)	Indicates that the cache contains data from an external data source.
scenario (Scenario Summary Report)	Indicates that the cache contains a scenario summary report
worksheet (Worksheet)	Indicates that the cache contains worksheet data.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SourceType) is located in §A.2. *end note*]

18.18.76 ST_Sqref (Reference Sequence)

A sequence of cell references, space delimited.

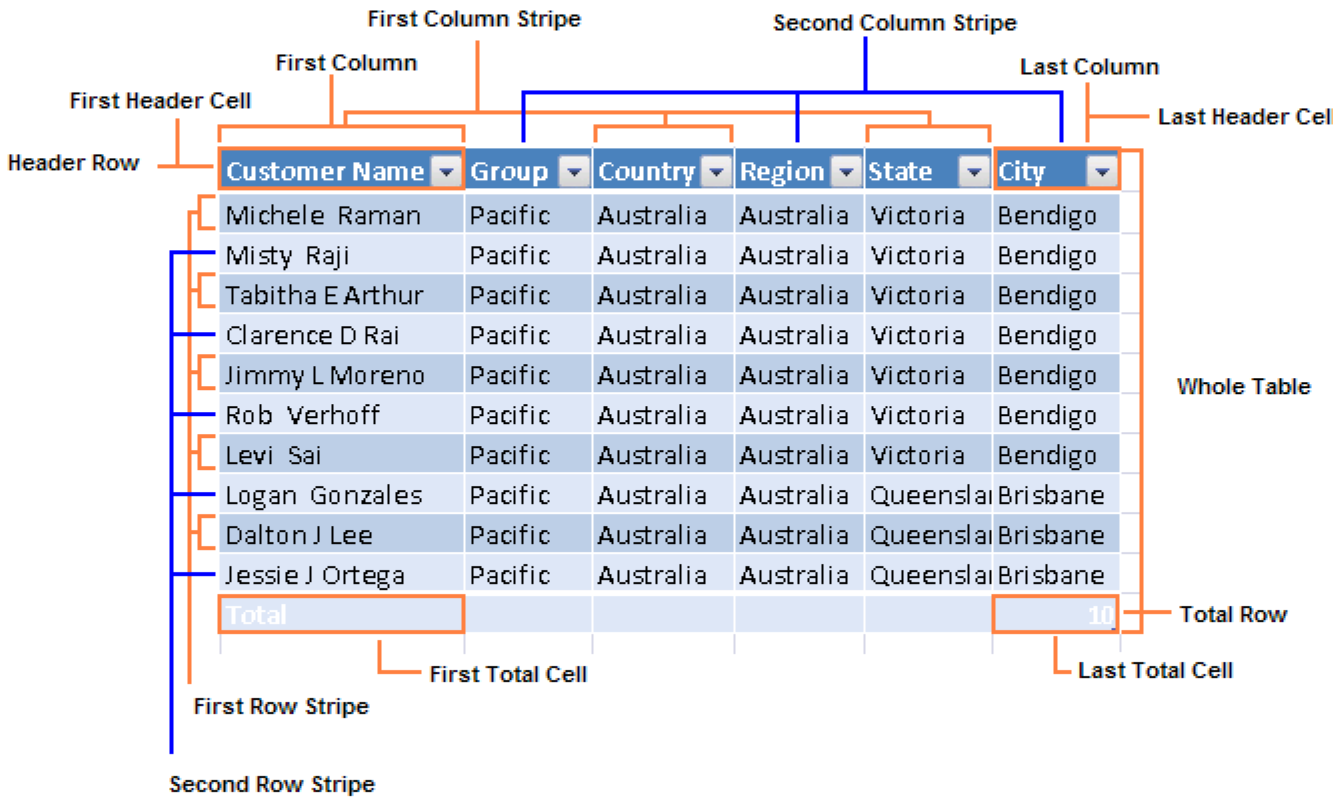
This simple type allows a list of items of the ST_Ref simple type (§18.18.62).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Sqref) is located in §A.2. *end note*]

18.18.77 ST_TableStyleType (Table Style Type)

Enumeration of the different structured regions of a Table or PivotTable which can be formatted. Specifies which region is being formatted by this table style element.

Table Regions



Customer Name	Group	Country	Region	State	City
Michele Raman	Pacific	Australia	Australia	Victoria	Bendigo
Misty Raji	Pacific	Australia	Australia	Victoria	Bendigo
Tabitha E Arthur	Pacific	Australia	Australia	Victoria	Bendigo
Clarence D Rai	Pacific	Australia	Australia	Victoria	Bendigo
Jimmy L Moreno	Pacific	Australia	Australia	Victoria	Bendigo
Rob Verhoff	Pacific	Australia	Australia	Victoria	Bendigo
Levi Sai	Pacific	Australia	Australia	Victoria	Bendigo
Logan Gonzales	Pacific	Australia	Australia	Queensla	Brisbane
Dalton J Lee	Pacific	Australia	Australia	Queensla	Brisbane
Jessie J Ortega	Pacific	Australia	Australia	Queensla	Brisbane
Total					10

PivotTable Regions

Blank Row

Only applies when "Insert blank row after each item" is ON.

Country	(All)	
Sum of Sales Amount	Column Labels	
	⊟2001	
	⊟3	
	⊟July	July Total
Row Labels	No Discount	
⊟New South Wales	75625.4664	75625.4664
⊟Coffs Harbour	699.0982	699.0982
⊟Bikes	699.0982	699.0982
⊟Road Bikes	699.0982	699.0982
Road-150 Red, 48		
Road-150 Red, 52		
Road-150 Red, 62		
Road-650 Red, 44	699.0982	699.0982
⊟Darlinghurst	3578.27	3578.27
⊟Bikes	3578.27	3578.27
⊟Road Bikes	3578.27	3578.27
Road-150 Red, 48		
Road-150 Red, 56	3578.27	3578.27
Road-150 Red, 62		
⊟Goulburn	18412.1682	18412.1682
⊟Bikes	18412.1682	18412.1682
⊟Mountain Bikes	3399.99	3399.99
Mountain-100 Silver, 44	3399.99	3399.99
⊟Road Bikes	15012.1782	15012.1782

Whole Table

Country	(All)	
Sum of Sales Amount	Column Labels	
	2001	
	3	
	July	July Total
Row Labels	No Discount	
New South Wales	75625.4664	75625.4664
Coffs Harbour	699.0982	699.0982
Bikes	699.0982	699.0982
Road Bikes	699.0982	699.0982
Road-150 Red, 48		
Road-150 Red, 52		
Road-150 Red, 62		
Road-650 Red, 44	699.0982	699.0982
Darlinghurst	3578.27	3578.27
Bikes	3578.27	3578.27
Road Bikes	3578.27	3578.27
Road-150 Red, 48		
Road-150 Red, 56	3578.27	3578.27
Road-150 Red, 62		

Page Field Labels

Country	(All)	
Sum of Sales Amount	Column Labels	
	2001	
	3	
	July	July Total
Row Labels	No Discount	
New South Wales	75625.4664	75625.4664
Coffs Harbour	699.0982	699.0982
Bikes	699.0982	699.0982
Road Bikes	699.0982	699.0982

Page Field Values

Country	(All)	
Sum of Sales Amount	Column Labels	
	2001	
	3	
	July	July Total
Row Labels	No Discount	
New South Wales	75625.4664	75625.4664
Coffs Harbour	699.0982	699.0982
Bikes	699.0982	699.0982
Road Bikes	699.0982	699.0982

First Column Stripe

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
	3				
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54
Road-150 Red, 48					3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27	

Second Column Stripe

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
	3				
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54
Road-150 Red, 48					3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27	
Road-650 Red, 44	699.0982	699.0982			
Darlinghurst	3578.27	3578.27	3578.27	3578.27	7156.54
Bikes	3578.27	3578.27	3578.27	3578.27	7156.54
Road Bikes	3578.27	3578.27	3578.27	3578.27	7156.54
Road-150 Red, 48			3578.27	3578.27	
Road-150 Red, 56	3578.27	3578.27			3578.27
Road-150 Red, 62					3578.27

First Row Stripe

Country	(All)			
Sum of Sales Amount	Column Labels			
	2001			
	3			
	July	July Total	August	August Total
Row Labels	No Discount		No Discount	
New South Wales	75625.4664	75625.4664	119823.881	119823.881
Coffs Harbour	699.0982	699.0982	7156.54	7156.54
Bikes	699.0982	699.0982	7156.54	7156.54
Road Bikes	699.0982	699.0982	7156.54	7156.54
Road-150 Red, 48				
Road-150 Red, 52			3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982		
Darlinghurst	3578.27	3578.27	3578.27	3578.27
Bikes	3578.27	3578.27	3578.27	3578.27
Road Bikes	3578.27	3578.27	3578.27	3578.27
Road-150 Red, 48			3578.27	3578.27
Road-150 Red, 56	3578.27	3578.27		
Road-150 Red, 62				

Second Row Stripe

Country	(All)			
Sum of Sales Amount	Column Labels			
	⊟2001			
	⊟3			
	⊟July	July Total	⊟August	August Total
Row Labels	No Discount		No Discount	
⊟New South Wales	75625.4664	75625.4664	119823.881	119823.881
⊟Coffs Harbour	699.0982	699.0982	7156.54	7156.54
⊟Bikes	699.0982	699.0982	7156.54	7156.54
⊟Road Bikes	699.0982	699.0982	7156.54	7156.54
Road-150 Red, 48				
Road-150 Red, 52			3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982		
⊟Darlinghurst	3578.27	3578.27	3578.27	3578.27
⊟Bikes	3578.27	3578.27	3578.27	3578.27
⊟Road Bikes	3578.27	3578.27	3578.27	3578.27
Road-150 Red, 48			3578.27	3578.27
Road-150 Red, 56	3578.27	3578.27		
Road-150 Red, 62				

First Column

Country	(All)		
Sum of Sales Amount	Column Labels		
	2001		
	3		
	July	July Total	August
Row Labels	No Discount		No Discount
New South Wales	75625.4664	75625.4[illegible] July Total	[illegible]9823.881
Coffs Harbour	699.0982	699.0982	7156.54
Bikes	699.0982	699.0982	7156.54
Road Bikes	699.0982	699.0982	7156.54
Road-150 Red, 48			
Road-150 Red, 52			3578.27
Road-150 Red, 62			3578.27
Road-650 Red, 44	699.0982	699.0982	
Darlinghurst	3578.27	3578.27	3578.27
Bikes	3578.27	3578.27	3578.27
Road Bikes	3578.27	3578.27	3578.27
Road-150 Red, 48			3578.27
Road-150 Red, 56	3578.27	3578.27	
Road-150 Red, 62			

Header Row

Country	(All)		
Sum of Sales Amount	Column Labels		
	2001		
	3		
	July	July Total	August
Row Labels	No Discount		No Discount
New South Wales	75625.4664	75625.4664	119823.881
Coffs Harbour	699.0982	699.0982	7156.54
Bikes	699.0982	699.0982	7156.54
Road Bikes	699.0982	699.0982	7156.54
Road-150 Red, 48			
Road-150 Red, 52			3578.27
Road-150 Red, 62			3578.27
Road-650 Red, 44	699.0982	699.0982	

First Header Cell

Country	(All)		
Sum of Sales Amount	Column Labels		
	2001		
	3		
	July	July Total	August
Row Labels	No Discount		No Discount
New South Wales	75625.4664	75625.4664	119823.881
Coffs Harbour	699.0982	699.0982	7156.54
Bikes	699.0982	699.0982	7156.54
Road Bikes	699.0982	699.0982	7156.54
Road-150 Red, 48			
Road-150 Red, 52			3578.27
Road-150 Red, 62			3578.27
Road-650 Red, 44	699.0982	699.0982	

Subtotal Column 1

Country	(All)							
Sum of Sales Amount	Column Labels							
	2001							2001 Total
	3						3 Total	
	July	July Total	August	August Total	September	September Total		
Row Labels	No Discount		No Discount		No Discount			
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road-150 Red, 48					3578.27	3578.27	3578.27	3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27	3578.27	7156.54	7156.54
Road-150 Red, 62			3578.27	3578.27			3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982					699.0982	699.0982

Subtotal Column 2

Country	(All)							
Sum of Sales Amount	Column Labels							
	2001							2001 Total
	3						3 Total	
	July	July Total	August	August Total	September	September Total		
Row Labels	No Discount		No Discount		No Discount			
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road-150 Red, 48					3578.27	3578.27	3578.27	3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27	3578.27	7156.54	7156.54
Road-150 Red, 62			3578.27	3578.27			3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982					699.0982	699.0982

Subtotal Column 3

Country	(All)							
Sum of Sales Amount	Column Labels							
	2001							2001 Total
	3						3 Total	
	July	July Total	August	August Total	September	September Total		
Row Labels	No Discount		No Discount		No Discount			
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road-150 Red, 48					3578.27	3578.27	3578.27	3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27	3578.27	7156.54	7156.54
Road-150 Red, 62			3578.27	3578.27			3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982					699.0982	699.0982

Subtotal Row 1

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
	3				
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
Bendigo					
Mountain Bikes					
Mountain-100 Black, 38	3374.99	3374.99			
Mountain-100 Silver, 42	3399.99	3399.99			
Mountain-100 Silver, 48	3399.99	3399.99			
Mountain Bikes Total	10174.97	10174.97			
Road Bikes					
Road-150 Red, 44	3578.27	3578.27			
Road-150 Red, 52	3578.27	3578.27			
Road-150 Red, 62	3578.27	3578.27			3578.27
Road Bikes Total	10734.81	10734.81			3578.27
Bendigo Total	20909.78	20909.78			3578.27
Brisbane					
Road Bikes					
Road-150 Red, 44	3578.27	3578.27	3578.27	3578.27	
Road-150 Red, 62			3578.27	3578.27	
Road Bikes Total	3578.27	3578.27	7156.54	7156.54	
Brisbane Total	3578.27	3578.27	7156.54	7156.54	

Subtotal Row 2

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
	3				
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
Bendigo					
Mountain Bikes					
Mountain-100 Black, 38	3374.99	3374.99			
Mountain-100 Silver, 42	3399.99	3399.99			
Mountain-100 Silver, 48	3399.99	3399.99			
Mountain Bikes Total	10174.97	10174.97			
Road Bikes					
Road-150 Red, 44	3578.27	3578.27			
Road-150 Red, 52	3578.27	3578.27			
Road-150 Red, 62	3578.27	3578.27			3578.27
Road Bikes Total	10734.81	10734.81			3578.27
Bendigo Total	20909.78	20909.78			3578.27
Brisbane					
Road Bikes					
Road-150 Red, 44	3578.27	3578.27	3578.27	3578.27	
Road-150 Red, 62			3578.27	3578.27	
Road Bikes Total	3578.27	3578.27	7156.54	7156.54	

Subtotal Row 3

Country	(All)				
Sum of Sales Amount	Column Labels				
	⊟2001				
	⊟3				
	⊟July	July Total	⊟August	August Total	⊟September
Row Labels	No Discount		No Discount		No Discount
⊟New South Wales					
⊟Coffs Harbour					
⊟Road Bikes					
Road-150 Red, 48					3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27	
Road-650 Red, 44	699.0982	699.0982			
Road Bikes Total	699.0982	699.0982	7156.54	7156.54	7156.54
Coffs Harbour Total	699.0982	699.0982	7156.54	7156.54	7156.54
⊟Darlinghurst					
⊟Road Bikes					
Road-150 Red, 48			3578.27	3578.27	
Road-150 Red, 56	3578.27	3578.27			3578.27
Road-150 Red, 62					3578.27
Road Bikes Total	3578.27	3578.27	3578.27	3578.27	7156.54
Darlinghurst Total	3578.27	3578.27	3578.27	3578.27	7156.54

Column Subheading 1

Country	(All)							
Sum of Sales Amount	Column Labels							
	⊟2001							2001 Total
	⊟3						3 Total	
	⊟July	July Total	⊟August	August Total	⊟September	September Total		
Row Labels	No Discount		No Discount		No Discount			
⊟New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538
⊟Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
⊟Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
⊟Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782

Column Subheading 2

Country	(All)						
Sum of Sales Amount	Column Labels						
	2001						
	3						3 Total
	July	July Total	August	August Total	September	September Total	
Row Labels	No Discount		No Discount		No Discount		
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782

Column Subheading 3

Country	(All)							
Sum of Sales Amount	Column Labels							
	2001							2001 Total
	3						3 Total	
	July	July Total	August	August Total	September	September Total		
Row Labels	No Discount		No Discount		No Discount			
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782

Row Subheading 1

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
	3				
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
New South Wales					
Coffs Harbour					
Road Bikes					
Road-150 Red, 48					3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27	
Road-650 Red, 44	699.0982	699.0982			
Road Bikes Total	699.0982	699.0982	7156.54	7156.54	7156.54
Coffs Harbour Total	699.0982	699.0982	7156.54	7156.54	7156.54

Row Subheading 2

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
		3			
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
New South Wales					
Coffs Harbour					
Road Bikes					
Road-150 Red, 48					3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27	
Road-650 Red, 44	699.0982	699.0982			
Road Bikes Total	699.0982	699.0982	7156.54	7156.54	7156.54
Coffs Harbour Total	699.0982	699.0982	7156.54	7156.54	7156.54
Darlinghurst					
Road Bikes					
Road-150 Red, 48			3578.27	3578.27	
Road-150 Red, 56	3578.27	3578.27			3578.27
Road-150 Red, 62					3578.27
Road Bikes Total	3578.27	3578.27	3578.27	3578.27	7156.54
Darlinghurst Total	3578.27	3578.27	3578.27	3578.27	7156.54
Goulburn					
Mountain Bikes					
Mountain-100 Silver, 44	3399.99	3399.99			

Row Subheading 3

Country	(All)				
Sum of Sales Amount	Column Labels				
	2001				
	3				
	July	July Total	August	August Total	September
Row Labels	No Discount		No Discount		No Discount
New South Wales					
Coffs Harbour					
Road Bikes					
Road-150 Red, 48					3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27
Road-150 Red, 62			3578.27	3578.27	
Road-650 Red, 44	699.0982	699.0982			
Road Bikes Total	699.0982	699.0982	7156.54	7156.54	7156.54
Coffs Harbour Total	699.0982	699.0982	7156.54	7156.54	7156.54
Darlinghurst					
Road Bikes					
Road-150 Red, 48			3578.27	3578.27	
Road-150 Red, 56	3578.27	3578.27			3578.27
Road-150 Red, 62					3578.27
Road Bikes Total	3578.27	3578.27	3578.27	3578.27	7156.54
Darlinghurst Total	3578.27	3578.27	3578.27	3578.27	7156.54
Goulburn					
Mountain Bikes					
Mountain-100 Silver, 44	3399.99	3399.99			

Grand Total Column

Country	(All)								
Sum of Sales Amount	Column Labels								
	2001							2001 Total	Grand Total
	3						3 Total		
	July	July Total	August	August Total	September	September Total			
Row Labels	No Discount		No Discount		No Discount				
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538	279147.7538
Coffs Harbour	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782	15012.1782
Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782	15012.1782
Road Bikes	699.0982	699.0982	7156.54	7156.54	7156.54	7156.54	15012.1782	15012.1782	15012.1782
Road-150 Red, 48					3578.27	3578.27	3578.27	3578.27	3578.27
Road-150 Red, 52			3578.27	3578.27	3578.27	3578.27	7156.54	7156.54	7156.54
Road-150 Red, 62			3578.27	3578.27			3578.27	3578.27	3578.27
Road-650 Red, 44	699.0982	699.0982					699.0982	699.0982	699.0982

Grand Total Row

Country	(All)								
Sum of Sales Amount	Column Labels								
	2001							2001 Total	Grand Total
	3						3 Total		
	July	July Total	August	August Total	September	September Total			
Row Labels	No Discount		No Discount		No Discount				
New South Wales	75625.4664	75625.4664	119823.881	119823.881	83698.4064	83698.4064	279147.7538	279147.7538	279147.7538
Queensland	53228.4682	53228.4682	35376.14	35376.14	29821.0764	29821.0764	118425.6846	118425.6846	118425.6846
South Australia	7156.54	7156.54	11255.6282	11255.6282	6978.26	6978.26	25390.4282	25390.4282	25390.4282
Tasmania	6978.26	6978.26	10531.53	10531.53	3578.27	3578.27	21088.06	21088.06	21088.06
Victoria	66664.17	66664.17	45551.11	45551.11	49917.5	49917.5	162132.78	162132.78	162132.78
Grand Total	209652.9046	209652.9046	222538.2892	222538.2892	173993.5128	173993.5128	606184.7066	606184.7066	606184.7066

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
blankRow (Blank Row Style)	Table style element that applies to PivotTable's blank rows.
firstColumn (First Column Style)	Table style element that applies to table's first column.
firstColumnStripe (First Column Stripe Style)	Table style element that applies to table's first column stripes.
firstColumnSubheading (First Column Subheading Style)	Table style element that applies to PivotTable's first column subheading.
firstHeaderCell (First Header Row Style)	Table style element that applies to table's first header row cell.
firstRowStripe (First Row Stripe Style)	Table style element that applies to table's first row stripes.
firstRowSubheading (First Row Subheading Style)	Table style element that applies to PivotTable's first row subheading.
firstSubtotalColumn (First Subtotal Column Style)	Table style element that applies to PivotTable's first subtotal column.
firstSubtotalRow (First Subtotal Row Style)	Table style element that applies to pivot table's first subtotal row.
firstTotalCell (First Total Row Style)	Table style element that applies to table's first total row cell.

Enumeration Value	Description
headerRow (Header Row Style)	Table style element that applies to table's header row.
lastColumn (Last Column Style)	Table style element that applies to table's last column.
lastHeaderCell (Last Header Style)	Table style element that applies to table's last header row cell.
lastTotalCell (Last Total Row Style)	Table style element that applies to table's last total row cell.
pageFieldLabels (Page Field Labels Style)	Table style element that applies to pivot table's page field labels.
pageFieldValues (Page Field Values Style)	Table style element that applies to pivot table's page field values.
secondColumnStripe (Second Column Stipe Style)	Table style element that applies to table's second column stripes.
secondColumnSubheading (Second Column Subheading Style)	Table style element that applies to pivot table's second column subheading.
secondRowStripe (Second Row Stripe Style)	Table style element that applies to table's second row stripes.
secondRowSubheading (Second Row Subheading Style)	Table style element that applies to pivot table's second row subheading.
secondSubtotalColumn (Second Subtotal Column Style)	Table style element that applies to PivotTable's second subtotal column.
secondSubtotalRow (Second Subtotal Row Style)	Table style element that applies to PivotTable's second subtotal row.
thirdColumnSubheading (Third Column Subheading Style)	Table style element that applies to PivotTable's third column subheading.
thirdRowSubheading (Third Row Subheading Style)	Table style element that applies to PivotTable's third row subheading.
thirdSubtotalColumn (Third Subtotal Column Style)	Table style element that applies to pivot table's third subtotal column.
thirdSubtotalRow (Third Subtotal Row Style)	Table style element that applies to PivotTable's third subtotal row.
totalRow (Total Row Style)	Table style element that applies to table's total row.
wholeTable (Whole Table Style)	Table style element that applies to table's entire content.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TableStyleType) is located in §A.2. *end note*]

18.18.78 ST_TableType (Table Type)

An enumeration that specifies what the table data is based on.

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
queryTable (Query Table)	A table based on an external data query.
worksheet (Worksheet)	A table based on a worksheet data range.
xml (XML)	A table based on an XML mapping.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TableType) is located in §A.2. *end note*]

18.18.79 ST_TargetScreenSize (Target Screen Size Types)

This simple type defines the collection of screen resolutions that are supported for this workbook.

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 (1024 x 768 Resolution)	Sets the target screen resolution to 1024x768 pixels.
1152x882 (1152 x 882 Resolution)	Sets the target screen resolution to 1152x882 pixels.
1152x900 (1152 x 900 Resolution)	Sets the target screen resolution to 1152x900 pixels
1280x1024 (1280 x 1024 Resolution)	Sets the target screen resolution to 1280x1024 pixels.
1600x1200 (1600 x 1200 Resolution)	Sets the target screen resolution to 1600x1200 pixels.
1800x1440 (1800 x 1440 Resolution)	Sets the target screen resolution to 1800x1440 pixels.
1920x1200 (1920 x 1200 Resolution)	Sets the target screen resolution to 1920x1200 pixels.
544x376 (544 x 376 Resolution)	Sets the target screen resolution to 544x376 pixels.
640x480 (640 x 480 Resolution)	Sets the target screen resolution to 640x480 pixels.
720x512 (720 x 512 Resolution)	Sets the target screen resolution to 720x512 pixels.
800x600 (800 x 600 Resolution)	Sets the target screen resolution to 800x600 pixels.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TargetScreenSize) is located in §A.2. *end note*]

18.18.80 ST_TextHAlign (Comment Text Horizontal Alignment)

This simple type specifies the horizontal alignment of the text within a comment text field.

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 Alignment)	Specifies that the text is centered horizontally.
distributed (Distributed Alignment)	Specifies that the text is distributed horizontally.
justify (Justify Alignment)	Specifies that the text is justified horizontally.
left (Left Alignment)	Specifies that the text is left-aligned.
right (Right Alignment)	Specifies that the text is right-aligned.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TextHAlign) is located in §A.2. *end note*]

18.18.81 ST_TextVAlign (Comment Text Vertical Alignment)

This simple type specifies the vertical alignment of the text within a comment text field.

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
bottom (Bottom Alignment)	Specifies that the text is bottom-aligned.
center (Center Alignment)	Specifies that the text is centered vertically.
distributed (Distributed Alignment)	Specifies that the text is distributed vertically.
justify (Justify Alignment)	Specifies that the text is justified vertically.
top (Top Alignment)	Specifies that the text is top-aligned.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TextVAlign) is located in §A.2. *end note*]

18.18.82 ST_TimePeriod (Time Period Types)

Used in a "contains dates" conditional formatting rule. These are dynamic time periods, which change based on the date the conditional formatting is refreshed / applied.

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
last7Days (Last 7 Days)	A date in the last seven days.
lastMonth (Last Month)	A date occuring in the last calendar month.

Enumeration Value	Description
lastWeek (Last Week)	A date occuring last week.
nextMonth (Next Month)	A date occuring in the next calendar month.
nextWeek (Next Week)	A date occuring next week.
thisMonth (This Month)	A date occuring in this calendar month.
thisWeek (This Week)	A date occuring this week.
today (Today)	Today's date.
tomorrow (Tomorrow)	Tomorrow's date.
yesterday (Yesterday)	Yesterday's date.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TimePeriod) is located in §A.2. *end note*]

18.18.83 ST_TotalsRowFunction (Totals Row Function Types)

An enumeration that specifies what function is used to aggregate the data in a column before it is displayed in the totals row.

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
average (Average)	Represents the arithmetic mean.
count (Non Empty Cell Count)	Represents a count of the number of non-empty cells.
countNums (Count Numbers)	Represents the number of cells that contain numbers.
custom (Custom Formula)	Represents the formula provided in `totalsRowFormula`.
max (Maximum)	Represents the largest value.
min (Minimum)	Represents the smallest value.
none (None)	No total row.
stdDev (StdDev)	Represents the estimated standard deviation.
sum (Sum)	Represents the arithmetic sum.
var (Var)	Represents the estimated variance.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TotalsRowFunction) is located in §A.2. *end note*]

18.18.84 ST_Type (Top N Evaluation Type)

This simple type defines the values for the Top N conditional formatting evaluation for the PivotTable. For more information on Top N conditional formatting, see the Sheet (§18.3.1) reference material.

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 (All)	Indicates that Top N conditional formatting is evaluated across the entire scope range.
column (Column Top N)	Indicates that Top N conditional formatting is evaluated for each column.
none (Top N None)	Indicates that Top N conditional formatting is not evaluated
row (Row Top N)	Indicates that Top N conditional formatting is evaluated for each row.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Type) is located in §A.2. *end note*]

18.18.85 ST_UnderlineValues (Underline Types)

Represents the different types of possible underline formatting.

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
double (Double Underline)	Double-line underlining under each character in the cell. underlines are drawn through the descenders of characters such as g and p. 1.23
doubleAccounting (Accounting Double Underline)	Double-line accounting underlining under each character in the cell. The underlines are drawn under the descenders of characters such as g and p. 1.23
none (None)	No underline.
single (Single Underline)	Single-line underlining under each character in the cell. The underline is drawn through the descenders of characters such as g and p. 1.23
singleAccounting (Accounting Single Underline)	Single-line accounting underlining under each

Enumeration Value	Description
	character in the cell. The underline is drawn under the descenders of characters such as g and p. 

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_UnderlineValues) is located in §A.2. *end note*]

18.18.86 ST_UnsignedIntHex (Hex Unsigned Integer)

This simple type represents the Hex representation of an unsigned integer.

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_UnsignedIntHex) is located in §A.2. *end note*]

18.18.87 ST_UpdateLinks (Update Links Behavior Types)

This simple type defines when the application updates links to other workbooks when the workbook is opened.

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 (Always Update Links)	Indicates that links to other workbooks are always updated when the workbook is opened. The application will not display an alert in the user interface (UI).
never (Never Update Links)	Indicates that links to other workbooks are never updated when the workbook is opened. The application will not display an alert in the user interface.
userSet (User Set)	Indicates that the end-user specified whether they receive an alert to update links to other workbooks when the workbook is opened. [*Example*: The application can expose this option in an application settings dialog. *end example*]

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_UpdateLinks) is located in §A.2. *end note*]

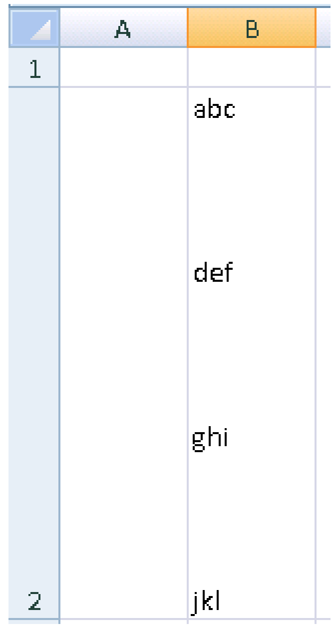
18.18.88 ST_VerticalAlignment (Vertical Alignment Types)

This enumeration value indicates the type of vertical alignment for a cell, i.e., whether it is aligned top, bottom, vertically centered, justified or distributed.

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:

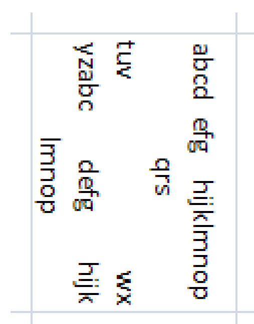
Enumeration Value	Description
bottom (Aligned To Bottom)	The vertical alignment is aligned-to-bottom.
center (Centered Vertical Alignment)	The vertical alignment is centered across the height of the cell.
distributed (Distributed Vertical Alignment)	When text direction is horizontal: the vertical alignment of lines of text is distributed vertically, where each line of text inside the cell is evenly distributed across the height of the cell, with flush top and bottom margins. When text direction is vertical: behaves exactly as distributed horizontal alignment. The first words in a line of text (appearing at the top of the cell) are flush with the top edge of the cell, and the last words of a line of text are flush with the bottom edge of the cell, and the line of text is distributed evenly from top to bottom. [*Example*: Horizontal text: this first example shows four lines of text (read horizontally from left to right) distributed vertically across the height of the cell. The first line is "abc", the second line is "def", the third line is "ghi" and the fourth line is "jkl".

<table>
<tr><th>Enumeration Value</th><th>Description</th></tr>
<tr><td></td><td>


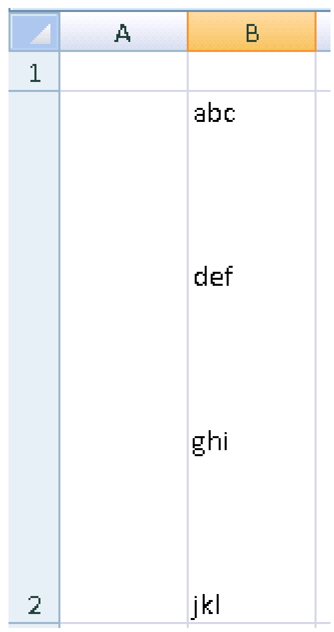
Vertical text: this second example shows three lines of text (read vertically from top to bottom) distributed vertically across the height of the cell. The lines of text are:

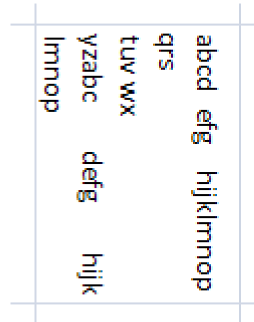
abcd efg hijklmnop qrs
tuv wx
yzabc defg hijk lmnop

The rendering looks like this:



end example]
</td></tr>
<tr><td>justify (Justified Vertically)</td><td>When text direction is horizontal: the vertical alignment of lines of text is distributed vertically, where each line of text inside the cell is evenly distributed across the height of the cell, with flush top</td></tr>
</table>

Enumeration Value	Description
	and bottom margins. When text direction is vertical: similar behavior as horizontal justification. The alignment is justified (flush top and bottom in this case). For each line of text, each line of the wrapped text in a cell is aligned to the top and bottom (except the last line). If no single line of text wraps in the cell, then the text is not justified. [*Example*: Horizontal text: this first example shows four lines of text (read horizontally from left to right) justified vertically across the height of the cell. The first line is "abc", the second line is "def", the third line is "ghi" and the fourth line is "jkl".  Vertical text: this second example shows three lines of text (read vertically from top to bottom) distributed vertically across the height of the cell. The lines of text are: abcd efg hijklmnop qrs tuv wx yzabc defg hijk lmnop The rendering looks like this:

Enumeration Value	Description
	 *end example*]
top (Align Top)	The vertical alignment is aligned-to-top.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_VerticalAlignment) is located in §A.2. *end note*]

18.18.89 ST_Visibility (Visibility Types)

This simple type defines the possible states for sheet visibility.

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
hidden (Hidden)	Indicates the workbook window is hidden, but can be shown by the user via the user interface.
veryHidden (Very Hidden)	Indicates the sheet is hidden and cannot be shown in the user interface (UI). This state is only available programmatically.
visible (Visible)	Indicates the workbook window is visible.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_Visibility) is located in §A.2. *end note*]

18.18.90 ST_VolDepType (Volatile Dependency Types)

This simple type defines the dependency types available for this workbook.

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
olapFunctions (OLAP Formulas)	Indicates that the type is Cube Functions.
realTimeData (Real Time Data)	Indicates that the type is Real Time Data (RTD).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_VolDepType) is located in §A.2. *end note*]

18.18.91 ST_VolValueType (Volatile Dependency Value Types)

This simple type defines the data type of the values in the dependency cache.

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
b (Boolean)	Indicates topic value is a boolean.
e (Error)	Indicates topic value is an error.
n (Real Number)	Indicates topic value is a real number.
s (String)	Indicates topic value is a string.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_VolValueType) is located in §A.2. *end note*]

18.18.92 ST_WebSourceType (Web Source Type)

This is an enumeration of types of objects which can be selected from the workbook to be published as HTML. For example, the entire sheet can be published, or a narrower set of objects on the sheet can be published, like a chart or a range.

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
autoFilter (AutoFilter)	Auto filter
chart (Chart)	Chart
label (Label)	Label
pivotTable (PivotTable)	PivotTable
printArea (Print Area)	Print area
query (QueryTable)	Query Table

Enumeration Value	Description
range (Range)	Range of cells
sheet (All Sheet Content)	All content of a sheet

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_WebSourceType) is located in §A.2. *end note*]

18.18.93 ST_XmlDataType (XML Data Types)

Represents a W3C XML built-in datatype name (http://www.w3.org/TR/xmlschema-2/). The values permitted by this type are the names of the simple datatypes defined by the XMLSchema Library, http://www.w3.org/2001/XML-Schema-datatypes.

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_XmlDataType) is located in §A.2. *end note*]

18.18.94 ST_FontFamily (Font Family)

This simple type specifies a font family. A font family is a set of fonts having common stroke width and serif characteristics. This is system-level font information.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

This simple type is restricted to the values listed in the following table:

Value	Font Family
0	Not applicable.
1	Roman
2	Swiss
3	Modern
4	Script
5	Decorative
6	Reserved for future use
7	Reserved for future use
8	Reserved for future use
9	Reserved for future use
10	Reserved for future use
11	Reserved for future use
12	Reserved for future use
13	Reserved for future use

Value	Font Family
14	Reserved for future use

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_FontFamily) is located in §A.2. *end note*]

19. PresentationML 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 PresentationML document, as defined by §13 of this Part of ISO/IEC 29500.

19.1 Table of Contents

This subclause is informative.

End of informative text.

19.2 Presentation

The Presentation portion of the PresentationML framework houses a set of elements that describe the storing of presentation-wide and view-specific properties. The presentation-wide properties are those that pertain to the entire presentation. The view-specific properties assist the generating application and viewing application by storing parameters that pertain to the final delivery of the presentation.

19.2.1 Presentation Properties

This section contains all presentation-level properties that pertain to a presentation document:

19.2.1.1 bold (Bold Embedded Font)

This element specifies a bold embedded font that is linked to a parent typeface. Once specified, this bold version of the given typeface name is available for use within the presentation. The actual font data is referenced using a relationships file that contains links to all fonts available. This font data contains font information for each of the characters to be made available.

[*Example*: Consider the following embedded font with a bold version specified.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
  <p:bold r:id="rId2"/>
</p:embeddedFont>
```

end example]

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

Attributes	Description
id (Relationship Identifier) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this embedded font that is referenced in a presentation. 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_EmbeddedFontDataId) is located in §A.3. *end note*]

19.2.1.2 boldItalic (Bold Italic Embedded Font)

This element specifies a bold italic embedded font that is linked to a parent typeface. Once specified, this bold italic version of the given typeface name is available for use within the presentation. The actual font data is referenced using a relationships file that contains links to all fonts available. This font data contains font information for each of the characters to be made available.

[*Example*: Consider the following embedded font with a bold italic version specified.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
  <p:boldItalic r:id="rId2"/>
</p:embeddedFont>
```

end example]

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

Attributes	Description
id (Relationship Identifier) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this embedded font that is referenced in a presentation. 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_EmbeddedFontDataId) is located in §A.3. *end note*]

19.2.1.3 browse (Browse Slide Show Mode)

This element specifies that the presentation slide show should be viewed in a single window or browse mode, instead of full screen.

[*Example*: Consider the following presentation that is to be viewed in a browse mode.

```
<p:presentationPr xmlns:a="…" xmlns:r="…" xmlns:p="…">
  <p:showPr>
    …
    <p:browse showScrollbar="0"/>
    …
  </p:showPr>
</p:presentationPr>
```

end example]

Attributes	Description
showScrollbar (Show Scroll Bar in Window)	Specifies whether to show the scroll bar in the viewing window. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ShowInfoBrowse) is located in §A.3. *end note*]

19.2.1.4 clrMru (Color MRU)

This specifies the most recently used user-selected colors within the presentation. This list contains custom user-selected colors outside the presentation's theme colors, enabling the application to expose these additional color choices for easy reuse. The first item in the list is the most recently used color.

[*Example*: Consider the following presentation with two user-selected colors in the color MRU list.

```
<p:presentationPr xmlns:a="…" xmlns:r="…" xmlns:p="…">
  …
  <p:clrMru>
    <a:srgbClr val="5361EB"/>
    <a:srgbClr val="CCECFF"/>
  </p:clrMru>
  …
</p:presentationPr>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_ColorMRU) is located in §A.4.1. *end note*]

19.2.1.5 custShow (Custom Show)

This element specifies a custom show which is an ordered list of a group of slides that are contained within the presentation. The custom show element allows for the specification of a presentation order that is different from the order in which the slides themselves are stored.

[*Example*: Consider the following custom show list that outlines a couple custom shows for a given set of slides.

```
<p:custShowLst>
  <p:custShow name="Custom Show 1" id="0">
    <p:sldLst>
      <p:sld r:id="rId4"/>
      <p:sld r:id="rId3"/>
      <p:sld r:id="rId2"/>
      <p:sld r:id="rId5"/>
    </p:sldLst>
  </p:custShow>
  <p:custShow name="Custom Show 2" id="1">
    <p:sldLst>
      <p:sld r:id="rId4"/>
      <p:sld r:id="rId5"/>
    </p:sldLst>
  </p:custShow>
</p:custShowLst>
```

In the above example there are two custom shows specified. The first specifies to present the slides in the order of 4, 3, 2 then 5 while the second specifies to play only slide 4 then 5. *end example*]

Attributes	Description
id (Custom Show Identifier)	This attribute specifies the custom show identification number. This is a number given that should be unique within the presentation document. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomShowId) is located in §A.3. *end note*]

19.2.1.6 custShow (Custom Show)

This element specifies a custom show that defines a specific slide sequence that the slides are displayed in. This allows for many variants of the same set of slides to be presented.

[*Example*: Consider the following custom show using three slides.

```
<p:custShow name="Custom Show 1" id="0">
  <p:sldLst>
    <p:sld r:id="rId5"/>
    <p:sld r:id="rId2"/>
    <p:sld r:id="rId4"/>
  </p:sldLst>
</p:custShow>
```

Notice here that the custom show specifies a show, or presentation, where slide 5 is shown first, then slide 2 and finally slide 4. *end example*]

Attributes	Description
id (Custom Show ID)	Specifies the identification number for this custom show. This should be unique among all the custom shows within the corresponding presentation. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (Custom Show Name)	Specifies a name for the custom show. The possible values for this attribute are defined by the ST_Name simple type (§19.7.5).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomShow) is located in §A.3. *end note*]

19.2.1.7 custShowLst (List of Custom Shows)

This element specifies a list of all custom shows that are available within the corresponding presentation. A custom show is a defined slide sequence that allows for the displaying of the slides with the presentation in any arbitrary order.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomShowList) is located in §A.3. *end note*]

19.2.1.8 defaultTextStyle (Presentation Default Text Style)

This element specifies the default text styles that are to be used within the presentation. The text style defined here can be referenced when inserting a new slide if that slide is not associated with a master slide or if no styling information has been otherwise specified for the text within the presentation slide.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextListStyle) is located in §A.4.1. *end note*]

19.2.1.9 embeddedFont (Embedded Font)

This element specifies an embedded font. Once specified, this font is available for use within the presentation. Within a font specification there can be regular, bold, italic and boldItalic versions of the font specified. The actual font data for each of these is referenced using a relationships file that contains links to all available fonts. This font data contains font information for each of the characters to be made available in each version of the font.

[*Example*: Consider the following embedded font.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
  <p:regular r:id="rId2"/>
</p:embeddedFont>
```

end example]

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_EmbeddedFontListEntry) is located in §A.3. *end note*]

19.2.1.10 embeddedFontLst (Embedded Font List)

This element specifies a list of fonts that are embedded within the corresponding presentation. The font data for these fonts is stored alongside the other document parts within the document container. The actual font data is referenced within the embeddedFont element.

[*Note*: The W3C XML Schema definition of this element's content model (CT_EmbeddedFontList) is located in §A.3. *end note*]

19.2.1.11 ext (Extension)

This element specifies an extension that is used for future extensions to the current version of DrawingML. This allows for the specifying of currently unknown elements for later versions of generating applications.

[*Note*: This element is not intended to reintroduce transitional schema into the strict conformance class. *end note*]

Attributes	Description
uri (Uniform Resource Identifier)	This attribute specifies the URI, or uniform resource identifier that represents the data stored under this tag. The URI is used to identify the correct 'server' that can process the contents of this tag.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema token datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Extension) is located in §A.3. *end note*]

19.2.1.12 extLst (Extension List)

This element specifies the extension list within which all future extensions of element type ext are defined. The extension list along with corresponding future extensions is used to extend the storage capabilities of the PresentationML framework. This allows for various new kinds of data to be stored natively within the framework.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExtensionList) is located in §A.3. *end note*]

19.2.1.13 font (Embedded Font Name)

This element specifies specific properties describing an embedded font. Once specified, this font is available for use within the presentation. Within a font specification there can be regular, bold, italic and boldItalic versions of the font specified. The actual font data for each of these is referenced using a relationships file that contains links to all available fonts. This font data contains font information for each of the characters to be made available in each version of the font.

[*Example*: Consider the following embedded font.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
  <p:regular r:id="rId2"/>
</p:embeddedFont>
```

end example]

Font Substitution Logic:

If the specified font is not available on a system being used for rendering, then the attributes of this element are to be utilized in selecting an alternate font.

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

Attributes	Description
typeface (Text Typeface)	Specifies the typeface, or name of the font that is to be used. The typeface is a string name of the specific font that should be used in rendering the presentation. If this font is

Attributes	Description
Namespace: http://purl.oclc.org/ooxml/drawingml/main	not available within the font list of the generating application than font substitution logic should be utilized in order to select an alternate font. The possible values for this attribute are defined by the ST_TextTypeface simple type (§20.1.10.80).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextFont) is located in §A.4.1. *end note*]

19.2.1.14 handoutMasterId (Handout Master ID)

This element specifies a handout master that is available within the corresponding presentation. A handout master is a slide that is specifically designed for printing as a handout.

[*Example*: Consider the following specification of a handout master within a presentation

```
<p:presentation xmlns:a="…" xmlns:r="…" xmlns:p="…" embedTrueTypeFonts="1">
  …
  <p:handoutMasterIdLst>
    <p:handoutMasterId r:id="rId8"/>
  </p:handoutMasterIdLst>
  …
</p:presentation>
```

end example]

Attributes	Description
id (Relationship Identifier) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location within a presentation of the handoutMaster element defining this handout master. 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_HandoutMasterIdListEntry) is located in §A.3. *end note*]

19.2.1.15 handoutMasterIdLst (List of Handout Master IDs)

This element specifies a list of identification information for the handout master slides that are available within the corresponding presentation. A handout master is a slide that is specifically designed for printing as a handout.

[*Note*: The W3C XML Schema definition of this element's content model (CT_HandoutMasterIdList) is located in §A.3. *end note*]

19.2.1.16 italic (Italic Embedded Font)

This element specifies an italic embedded font that is linked to a parent typeface. Once specified, this italic version of the given typeface name is available for use within the presentation. The actual font data is referenced using a relationships file that contains links to all fonts available. This font data contains font information for each of the characters to be made available.

[*Example*: Consider the following embedded font with a italic version specified.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
  <p:italic r:id="rId2"/>
</p:embeddedFont>
```

end example]

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

Attributes	Description
id (Relationship Identifier) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this embedded font that is referenced in a presentation. 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_EmbeddedFontDataId) is located in §A.3. *end note*]

19.2.1.17 kinsoku (Kinsoku Settings)

This element specifies the presentation-wide kinsoku settings that define the line breaking behaviour of East Asian text within the corresponding presentation.

Attributes	Description
invalEndChars (Invalid Kinsoku End Characters)	Specifies the characters that cannot end a line of text. The possible values for this attribute are defined by the W3C XML Schema string datatype.
invalStChars	Specifies the characters that cannot start a line of text.

Attributes	Description
(Invalid Kinsoku Start Characters)	The possible values for this attribute are defined by the W3C XML Schema string datatype.
lang (Language)	Specifies the corresponding East Asian language that these settings apply to. The possible values for this attribute are defined by the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Kinsoku) is located in §A.3. *end note*]

19.2.1.18 kiosk (Kiosk Slide Show Mode)

This element specifies that the presentation slide show should be viewed in a full-screen kiosk mode. A presentation viewed in kiosk mode should have user input disabled and restarts after a specified interval.

[*Example*: Consider the following presentation that is set to be viewed in a looping kiosk mode.

```
<p:presentationPr xmlns:a="…" xmlns:r="…" xmlns:p="…">
  <p:showPr loop="1" showNarration="1">
    …
    <p:kiosk/>
    …
  </p:showPr>
</p:presentationPr>
```

end example]

Attributes	Description
restart (Restart Show)	Specifies the time length that the presentation should run until it is to be restarted. That is, the presentation should loop back to the first slide specified in the presentation or custom show. This value is specified in 1/1000ths of a second and measured from the most recent time the presentation started or restarted. [*Note*: The counter is reset when a presentation is restarted due to automatic looping at the end of a show, if specified by the loop attribute of showPr. *end note*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ShowInfoKiosk) is located in §A.3. *end note*]

19.2.1.19 modifyVerifier (Modification Verifier)

This element specifies the write protection settings which have been applied to a PresentationML document. *Write protection* refers to a mode in which the document's contents should not be modified, and the document should not be resaved using the same file name.

When present, the application shall require a password to enable modifications to the document. If the supplied password does not match the hash value in this attribute, then write protection shall be enabled. If this element is omitted, then no write protection shall be applied to the current document. Since this protection does not encrypt the document, malicious applications might circumvent its use.

The password supplied to the algorithm is to be a UTF-16LE encoded string; strings longer than 510 octets are truncated to 510 octets. If there is a leading BOM character (U+FEFF) in the encoded password it is removed before hash calculation. The attributes of this element specify the algorithm to be used to verify the password provided by the user.

[*Example*: Consider a PresentationML document that can only be opened 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 PresentationML:

```
<p:modifyVerifier p:algorithmName="SHA-512" …
p:hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" … />
```

…In order for the hosting application to enable edits to the document, the hosting application would have to be provided with a password that the hosting application would then hash using the algorithm specified by the algorithm attributes and compare to the value of the hashValue attribute (`9oN7nWkCAyEZib1RomSJTjmPpCY=`). If the two values matched, the file would be opened in an editable state. *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.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</td></tr>
<tr><td>MD4</td><td>Specifies that the MD4 algorithm, as defined by RFC 1320, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>breaks. end note]</td></tr>
<tr><td><code>MD5</code></td><td>Specifies that the MD5 algorithm, as defined by RFC 1321, shall be used.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</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.

[Note: It is recommended that applications should avoid using this algorithm to store new hash values, due to publically known breaks. end note]</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>

[Example: Consider an Office Open XML document with the following information stored in one of its protection elements:

<code>< … algorithmName="SHA-1"
  hashValue="9oN7nWkCAyEZib1RomSJTjmPpCY=" /></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. end example]

The possible values for this attribute are defined by the W3C XML Schema string datatype.</td></tr>
<tr><td>hashValue (Password Hash Value)</td><td>Specifies the hash value for the password required to edit this chartsheet. 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.</td></tr>
</table>

Attributes	Description
	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.
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.
spinValue (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.

Attributes	Description
	[*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 hashValue attribute. *end example*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ModifyVerifier) is located in §A.3. *end note*]

19.2.1.20 notesMasterId (Notes Master ID)

This element specifies a notes master that is available within the corresponding presentation. A notes master is a slide that is specifically designed for the printing of the slide along with any attached notes.

[*Example*: Consider the following specification of a notes master within a presentation

```
<p:presentation xmlns:a="…" xmlns:r="…" xmlns:p="…" embedTrueTypeFonts="1">
  …
  <p:notesMasterIdLst>
    <p:notesMasterId r:id="rId8"/>
  </p:notesMasterIdLst>
  …
</p:presentation>
```

end example]

Attributes	Description
id (Relationship Identifier) Namespace: http://purl.oclc.or	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location within a presentation of the notesMaster element defining this notes master. The possible values for this attribute are defined by the ST_RelationshipId simple type

Attributes	Description
g/ooxml/officeDocument/relationships	(§22.8.2.1).

[*Note*: The W3C XML Schema definition of this element's content model (CT_NotesMasterIdListEntry) is located in §A.3. *end note*]

19.2.1.21 notesMasterIdLst (List of Notes Master IDs)

This element specifies a list of identification information for the notes master slides that are available within the corresponding presentation. A notes master is a slide that is specifically designed for the printing of the slide along with any attached notes.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NotesMasterIdList) is located in §A.3. *end note*]

19.2.1.22 notesSz (Notes Slide Size)

This element specifies the size of slide surface used for notes slides and handout slides. Objects within a notes slide can be specified outside these extents, but the notes slide has a background surface of the specified size when presented or printed. This element is intended to specify the region to which content is fitted in any special format of printout the application might choose to generate, such as an outline handout.

[*Example*: Consider the following specifying of the size of a notes slide.

```
<p:presentation xmlns:a="…" xmlns:r="…" xmlns:p="…" embedTrueTypeFonts="1">
  …
  <p:notesSz cx="9144000" cy="6858000"/>
  …
</p:presentation>
```

end example]

Attributes	Description
cx (Extent Length) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies the length of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `<… cx="1828800" cy="200000"/>` The cx attributes specifies that this object has a height of `1828800` EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple

Attributes	Description
	type (§20.1.10.41).
cy (Extent Width) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies the width of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `< … cx="1828800" cy="200000"/>` The cy attribute specifies that this object has a width of `200000` EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§20.1.10.41).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PositiveSize2D) is located in §A.4.1. *end note*]

19.2.1.23 penClr (Pen Color for Slide Show)

This element specifies the pen color that should be used to make markings on the slides while in a presentation.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Color) is located in §A.4.1. *end note*]

19.2.1.24 photoAlbum (Photo Album Information)

This element specifies that the corresponding presentation contains a photo album. A photo album specifies a list of images within the presentation that spread across one or more slides, all of which share a consistent layout. Each image in the album is formatted with a consistent style. This functionality enables the application to manage all of the images together and modify their ordering, layout, and formatting as a set.

This element does not enforce the specified properties on individual photo album images; rather, it specifies common settings that should be applied by default to all photo album images and their containing slides. Images that are part of the photo album are identified by the presence of the isPhoto element in the definition of the picture.

[*Example*: Consider the following presentation that has been specified as a photo album

```
<p:presentation xmlns:a="…" xmlns:r="…" xmlns:p="…" embedTrueTypeFonts="1">
  …
  <p:photoAlbum bw="1" layout="2pic"/>
  …
</p:presentation>
```

end example]

Attributes	Description
bw (Black and White)	Specifies whether all pictures in the photo album are to be displayed as black and white. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
frame (Frame Type)	Specifies the frame type that is to be used on all the pictures in the photo album. The possible values for this attribute are defined by the ST_PhotoAlbumFrameShape simple type (§19.7.7).
layout (Photo Album Layout)	Specifies the layout that is to be used to arrange the pictures in the photo album on individual slides. The possible values for this attribute are defined by the ST_PhotoAlbumLayout simple type (§19.7.8).
showCaptions (Show/Hide Captions)	Specifies whether to show captions for pictures in the photo album. Captions are text boxes grouped with each image, with the group set to not allow ungrouping. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PhotoAlbum) is located in §A.3. *end note*]

19.2.1.25 present (Presenter Slide Show Mode)

This element specifies that the presentation slide show should be viewed in a full-screen presenter mode. In this mode, the presentation is displayed on one monitor while a different monitor displays notes and provides navigation controls intended to be viewed only by the presenter.

[*Example*: Consider the following presentation that is set to be viewed in a present mode.

```
<p:presentationPr xmlns:a="…" xmlns:r="…" xmlns:p="…">
  <p:showPr>
    …
    <p:present/>
    …
  </p:showPr>
</p:presentationPr>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.3. *end note*]

19.2.1.26 presentation (Presentation)

This element specifies within it fundamental presentation-wide properties.

[*Example*: Consider the following presentation with a single slide master and two slides. In addition to these commonly used elements there can also be the specification of other properties such as slide size, notes size and default text styles.

```
<p:presentation xmlns:a="…" xmlns:r="…" xmlns:p="…">
    <p:sldMasterIdLst>
        <p:sldMasterId id="2147483648" r:id="rId1"/>
    </p:sldMasterIdLst>
    <p:sldIdLst>
        <p:sldId id="256" r:id="rId3"/>
        <p:sldId id="257" r:id="rId4"/>
    </p:sldIdLst>
    <p:sldSz cx="9144000" cy="6858000" type="screen4x3"/>
    <p:notesSz cx="6858000" cy="9144000"/>
    <p:defaultTextStyle>
      …
    </p:defaultTextStyle>
</p:presentation>
```

end example]

Attributes	Description
autoCompressPictures (Automatically Compress Pictures)	Specifies whether the generating application should automatically compress all pictures for this presentation. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
bookmarkIdSeed (Bookmark ID Seed)	Specifies a seed for generating bookmark IDs to ensure IDs remain unique across the document. This value specifies the number to be used as the ID for the next new bookmark created. The possible values for this attribute are defined by the ST_BookmarkIdSeed simple type (§19.7.1).
compatMode (Compatibility Mode)	Specifies whether the generating application is to be in a compatibility mode which serves to inform the user of any loss of content or functionality when working with older formats. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
conformance (Document Conformance Class)	Specifies the conformance class (§2.1) to which the PresentationML document conforms. If this attribute is omitted, its default value is `transitional`. [*Example*: Consider the following PresentationML Presentation part markup:

Attributes	Description
	`<p:presentation conformance="strict">` `…` `</p:presentation>` This document has a conformance attribute value of `strict`, therefore it conforms to the PML Strict conformance class. *end example*] The possible values for this attribute are defined by the ST_ConformanceClass simple type (§22.9.2.2).
embedTrueTypeFonts (Embed True Type Fonts)	Specifies whether the generating application should automatically embed true type fonts or not. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
firstSlideNum (First Slide Number)	Specifies the first slide number in the presentation. The possible values for this attribute are defined by the W3C XML Schema int datatype.
removePersonalInfoOnSave (Remove Personal Information on Save)	Specifies whether to automatically remove personal information when the presentation document is saved. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rtl (Right-To-Left Views)	Specifies if the current view of the user interface is oriented right-to-left or left-to-right. The view is right-to-left is this value is set to true, and left-to-right otherwise. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
saveSubsetFonts (Save Subset Fonts)	Specifies to save only the subset of characters used in the presentation when a font is embedded. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
serverZoom (Server Zoom)	Specifies the scaling to be used when the presentation is embedded in another document. The embedded slides are to be scaled by this percentage. The possible values for this attribute are defined by the ST_Percentage simple type (§20.1.10.40).
showSpecialPlsOnTitleSld (Show Header and Footer Placeholders on Titles)	Specifies whether to show the header and footer placeholders on the title slides. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
strictFirstAndLastC	Specifies whether to use strict characters for starting and ending lines of Japanese text.

Attributes	Description
hars (Strict First and Last Characters)	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Presentation) is located in §A.3. *end note*]

19.2.1.27 presentationPr (Presentation-wide Properties)

This element functions as a parent element within which additional presentation-wide document properties are contained. All properties and their corresponding settings are defined within the child elements.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PresentationProperties) is located in §A.3. *end note*]

19.2.1.28 prnPr (Printing Properties)

This element specifies the default printing properties associated with this presentation document.

Attributes	Description
clrMode (Print Color Mode)	Specifies the color mode to be used when printing. The possible values for this attribute are defined by the ST_PrintColorMode simple type (§19.7.11).
frameSlides (Frame slides when printing)	Specifies whether slides should be framed when printing. When framed, an outline border is printed for each slide. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
hiddenSlides (Print Hidden Slides)	Specifies whether hidden slides should be printed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
prnWhat (Print Output)	Specifies what the default print output is in terms of content layout. The possible values for this attribute are defined by the ST_PrintWhat simple type (§19.7.12).
scaleToFitPaper (Scale to Fit Paper when printing)	Specifies whether the print output should be scaled to fit the paper being used. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_PrintProperties) is located in §A.3. *end note*]

19.2.1.29 regular (Regular Embedded Font)

This element specifies a regular embedded font that is linked to a parent typeface. Once specified, this regular version of the given typeface name is available for use within the presentation. The actual font data is referenced using a relationships file that contains links to all fonts available. This font data contains font information for each of the characters to be made available.

[*Example*: Consider the following embedded font with a regular version specified.

```
<p:embeddedFont>
  <p:font typeface="MyFont" pitchFamily="34" charset="0"/>
  <p:regular r:id="rId2"/>
</p:embeddedFont>
```

end example]

[*Note*: Not all characters for a typeface must be stored. It is up to the generating application to determine which characters are to be stored in the corresponding font data files. *end note*]

Attributes	Description
id (Relationship Identifier) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this embedded font that is referenced in a presentation. 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_EmbeddedFontDataId) is located in §A.3. *end note*]

19.2.1.30 showPr (Presentation-wide Show Properties)

This element functions as a parent element within which all presentation-wide show properties are contained. All properties and their corresponding settings are defined within the child elements.

Attributes	Description
loop (Loop Slide Show)	Specifies whether the slide show should be set to loop at the end. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showAnimation	Specifies whether slide show animation should be shown when presenting.

Attributes	Description
(Show Animation in Slide Show)	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showNarration (Show Narration in Slide Show)	Specifies whether slide show narration should be played when presenting. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
useTimings (Use Timings in Slide Show)	Specifies whether slide transition timings should be used to advance slides when presenting. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ShowProperties) is located in §A.3. *end note*]

19.2.1.31 sld (Presentation Slide)

This element specifies a slide within a slide list. The slide list is used to specify an ordering of slides.

[*Example*: Consider the following custom show with an ordering of slides.

```
<p:custShowLst>
  <p:custShow name="Custom Show 1" id="0">
    <p:sldLst>
      <p:sld r:id="rId4"/>
      <p:sld r:id="rId3"/>
      <p:sld r:id="rId2"/>
      <p:sld r:id="rId5"/>
    </p:sldLst>
  </p:custShow>
</p:custShowLst>
```

In the above example the order specified to present the slides is slide 4, then 3, 2 and finally 5. *end example*]

Attributes	Description
id (Relationship ID) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	This attribute specifies the relationship id that is used to reference to the actual slide XML file that contains all the information to the slide listed within the slide list. 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_SlideRelationshipListEntry) is located in §A.3. *end note*]

19.2.1.32 sldAll (All Slides)

This attribute specifies all slides instead of a given range of slides for use within the html publishing properties as well as the show properties.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.3. *end note*]

19.2.1.33 sldId (Slide ID)

This element specifies a presentation slide that is available within the corresponding presentation. A slide contains the information that is specific to a single slide such as slide-specific shape and text information.

[*Example*: Consider the following specification of a slide master within a presentation

```
<p:presentation xmlns:a="…" xmlns:r="…" xmlns:p="…" embedTrueTypeFonts="1">
  ..
  <p:sldIdLst>
    <p:sldId id="256" r:id="rId3"/>
    <p:sldId id="257" r:id="rId4"/>
    <p:sldId id="258" r:id="rId5"/>
    <p:sldId id="259" r:id="rId6"/>
    <p:sldId id="260" r:id="rId7"/>
  </p:sldIdLst>
  ..
</p:presentation>
```

end example]

Attributes	Description
id (Relationship Identifier) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location within a presentation of the sld element defining this slide. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
id (Slide Identifier)	Specifies the slide identifier that is to contain a value that is unique throughout the presentation. The possible values for this attribute are defined by the ST_SlideId simple type (§19.7.13).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideIdListEntry) is located in §A.3. *end note*]

19.2.1.34 sldIdLst (List of Slide IDs)

This element specifies a list of identification information for the slides that are available within the corresponding presentation. A slide contains the information that is specific to a single slide such as slide-specific shape and text information.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideIdList) is located in §A.3. *end note*]

19.2.1.35 sldLst (List of Presentation Slides)

This element specifies a list of presentation slides. A presentation slide contains the information that is specific to a single slide such as slide-specific shape and text information.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideRelationshipList) is located in §A.3. *end note*]

19.2.1.36 sldMasterId (Slide Master ID)

This element specifies a slide master that is available within the corresponding presentation. A slide master is a slide that is specifically designed to be a template for all related child layout slides.

[*Example*: Consider the following specification of a slide master within a presentation

```
<p:presentation xmlns:a="…" xmlns:r="…" xmlns:p="…" embedTrueTypeFonts="1">
  …
  <p:sldMasterIdLst>
    <p:sldMasterId id="2147483648" r:id="rId1"/>
  </p:sldMasterIdLst>
  …
</p:presentation>
```

end example]

Attributes	Description
id (Relationship Identifier) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location within a presentation of the sldMaster element defining this slide master. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
id (Slide Master Identifier)	Specifies the slide master identifier that is to contain a value that is unique throughout the presentation.

Attributes	Description
	The possible values for this attribute are defined by the ST_SlideMasterId simple type (§19.7.16).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideMasterIdListEntry) is located in §A.3. *end note*]

19.2.1.37 sldMasterIdLst (List of Slide Master IDs)

This element specifies a list of identification information for the slide master slides that are available within the corresponding presentation. A slide master is a slide that is specifically designed to be a template for all related child layout slides.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideMasterIdList) is located in §A.3. *end note*]

19.2.1.38 sldRg (Slide Range)

This element specifies a slide range for use within the html publishing properties as well as the show properties.

[*Note*: The indexes used here correlate directly with the presentation slide numbers which they reference to. That is the slide range must be greater than or equal to 1 and also less than or equal to the number of slides in the presentation document. *end note*]

Attributes	Description
end (End)	This attribute defines the end of the index range. The possible values for this attribute are defined by the ST_Index simple type (§19.7.3).
st (Start)	This attribute defines the start of the index range. The possible values for this attribute are defined by the ST_Index simple type (§19.7.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_IndexRange) is located in §A.3. *end note*]

19.2.1.39 sldSz (Presentation Slide Size)

This element specifies the size of the presentation slide surface. Objects within a presentation slide can be specified outside these extents, but this is the size of background surface that is shown when the slide is presented or printed..

[*Example*: Consider the following specifying of the size of a presentation slide.

```
<p:presentation xmlns:a="…" xmlns:r="…" xmlns:p="…" embedTrueTypeFonts="1">
```



```
    …
    <p:sldSz cx="9144000" cy="6858000" type="screen4x3"/>
    …
  </p:presentation>
```

end example]

Attributes	Description
cx (Extent Length)	Specifies the length of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `<… cx="1828800" cy="200000"/>` The cx attributes specifies that this object has a height of `1828800` EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_SlideSizeCoordinate simple type (§19.7.17).
cy (Extent Width)	Specifies the width of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `< … cx="1828800" cy="200000"/>` The cy attribute specifies that this object has a width of `200000` EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_SlideSizeCoordinate simple type (§19.7.17).
type (Type of Size)	Specifies the kind of slide size that should be used. This identifies in particular the expected delivery platform for this presentation. The possible values for this attribute are defined by the ST_SlideSizeType simple type (§19.7.18).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideSize) is located in §A.3. *end note*]

19.2.1.40 smartTags (Smart Tags)

This element specifies that references to smart tags exist within this document. [*Note*: For a complete definition of smart tags, which are semantically identical throughout Office Open XML, see §17.5.1. *end note*] To denote the location of smart tags on individual runs of text, there smart tag identifier attributes are specified for each run to which a smart tag applies. These are further specified in the run property attributes within DrawingML.

[*Example*: Consider the following PresentationML markup:

```
<p:presentation>
  …
  <p:smartTags r:id="rId1"/>
</p:presentation>
```

The presence of the smartTags element specifies that there is smart tag information within the PresentationML package. Individual runs are then inspected for the value of the smtId attribute to determine where smart tags might apply, for example:

```
<p:txBody>
  <a:bodyPr/>
  <a:lstStyle/>
  <a:p>
    <a:r>
      <a:rPr lang="en-US" dirty="0" smtId="1"/>
      <a:t>CNTS</a:t>
    </a:r>
    <a:endParaRPr lang="en-US" dirty="0"/>
  </a:p>
</p:txBody>
```

In the sample above there is a smart tag identifier of 1 specified for this run of text to denote that the text should be inspected for smart tag information. *end example*]

Attributes	Description
id (Relationship Identifier) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this smart tag. 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_SmartTags) is located in §A.3. *end note*]

19.2.2 View Properties

This section contains all properties that pertain to the viewing of the presentation.

19.2.2.1 cSldViewPr (Common Slide View Properties)

This element functions as a container for slide view properties that are common across multiple view property elements. The specific properties and associated values for these view properties reside within the child elements and attributes.

Attributes	Description
showGuides (Show Guides in View)	Specifies whether to show guides when editing the presentation. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
snapToGrid (Snap Objects to Grid)	Specifies whether objects should snap to underlying presentation grid when editing. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
snapToObjects (Snap Objects to Objects)	Specifies whether objects should snap to other objects when editing the presentation. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CommonSlideViewProperties) is located in §A.3. *end note*]

19.2.2.2 cViewPr (Common View Properties)

This element specifies the view properties that are common across multiple view property elements.

Attributes	Description
varScale (Variable Scale)	Specifies that the view content should automatically scale to best fit the current window size. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CommonViewProperties) is located in §A.3. *end note*]

19.2.2.3 gridSpacing (Grid Spacing)

This element specifies the grid spacing that should be used for the grid underlying the presentation document. The grid can be used to align objects on the slide and to display visual positioning cues.

Attributes	Description
cx (Extent Length) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies the length of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `<… cx="1828800" cy="200000"/>` The cx attributes specifies that this object has a height of `1828800` EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§20.1.10.41).
cy (Extent Width) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies the width of the extents rectangle in EMUs. This rectangle shall dictate the size of the object as displayed (the result of any scaling to the original object). [*Example*: Consider a DrawingML object specified as follows: `< … cx="1828800" cy="200000"/>` The cy attribute specifies that this object has a width of `200000` EMUs (English Metric Units). *end example*] The possible values for this attribute are defined by the ST_PositiveCoordinate simple type (§20.1.10.41).

[*Note*: The W3C XML Schema definition of this element's content model (CT_PositiveSize2D) is located in §A.4.1. *end note*]

19.2.2.4 guide (A Guide)

This element specifies a guide within the presentation. Guides are lines used for arranging layouts and content and never appear except as an aid in editing slides.

Attributes	Description
orient (Guide Orientation)	Specifies the orientation for a guide. The possible values for this attribute are defined by the ST_Direction simple type (§19.7.2).
pos (Guide Position)	Specifies the position information for a guide. The possible values for this attribute are defined by the ST_Coordinate32 simple type (§20.1.10.17).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Guide) is located in §A.3. *end note*]

19.2.2.5 guideLst (List of Guides)

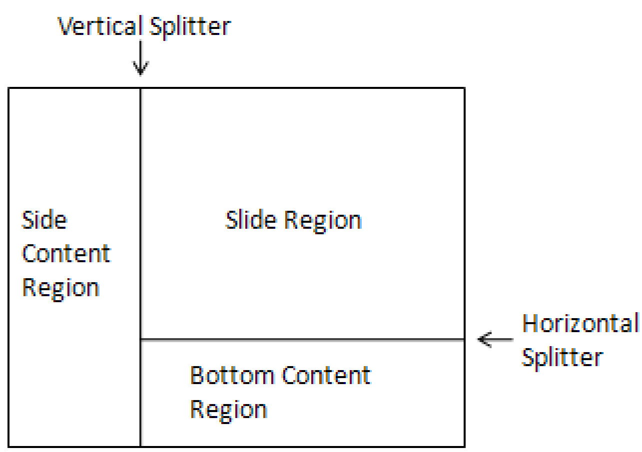
This element specifies a list of guides for a particular view of the presentation.

[*Note*: The W3C XML Schema definition of this element's content model (CT_GuideList) is located in §A.3. *end note*]

19.2.2.6 normalViewPr (Normal View Properties)

This element specifies the view properties associated with the normal view mode. The normal view consists of three content regions: the slide itself, a side content region, and a bottom content region. The content of the side content region and bottom content region is determined by the generating application. Properties pertaining to the positioning of the different content regions are stored in this element. This information allows the application to save its view state to the file, so that when reopened the view is in the same state as when the presentation was last saved.

A vertical splitter bar separates the slide from the side content region. A horizontal splitter bar separates the slide from the content region below the slide. If the presentation is set to left-to-right, the side content region is to the left of the slide. If the presentation is set to right-to-left, the side content region is to the right of the slide.



Attributes	Description
horzBarState (State of the Horizontal Splitter Bar)	Specifies the state that the horizontal splitter bar should be in when in normal view mode. The region to be maximized or minimized is the side content region. The possible values for this attribute are defined by the ST_SplitterBarState simple type (§19.7.19).
preferSingleView (Prefer Single View)	Specifies whether the user prefers to see a full-window single-content region over the standard normal view with three content regions. If enabled, the application can choose to display one of the content regions in the entire window.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showOutlineIcons (Show Outline Icons in Normal View)	Specifies whether the application should show icons if displaying outline content in any of the content regions of normal view mode. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
snapVertSplitter (Snap Vertical Splitter)	Specifies whether the vertical splitter should snap to a minimized state when the side region is sufficiently small. The specific parameters of this behaviour are left to the generating application. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
vertBarState (State of the Vertical Splitter Bar)	Specifies the state that the vertical splitter bar should be in when in normal view mode. The region to be maximized or minimized is the slide region. The possible values for this attribute are defined by the ST_SplitterBarState simple type (§19.7.19).

[*Note*: The W3C XML Schema definition of this element's content model (CT_NormalViewProperties) is located in §A.3. *end note*]

19.2.2.7 notesTextViewPr (Notes Text View Properties)

This element functions as a parent element within which all properties associated with the notes text view are contained. All properties are defined within the child elements.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NotesTextViewProperties) is located in §A.3. *end note*]

19.2.2.8 notesViewPr (Notes View Properties)

This element functions as a parent element within which all view properties associated with notes are contained. All properties are defined within the child elements.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NotesViewProperties) is located in §A.3. *end note*]

19.2.2.9 origin (View Origin)

This element specifies the origin of the slide when it is being viewed with various scaling factors using the scale element.

Attributes	Description
x (X-Axis	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified

Attributes	Description
Coordinate) Namespace: http://purl.oclc.org/ooxml/drawingml/main	by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<… x="0" y="100" />` The x attribute defines an x-coordinate of `0`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§20.1.10.16).
y (Y-Axis Coordinate) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<… x="0" y="100" />` The y attribute defines a y-coordinate of `100`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§20.1.10.16).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Point2D) is located in §A.4.1. *end note*]

19.2.2.10 outlineViewPr (Outline View Properties)

This element functions as a parent element within which all view properties associated with the outline view mode are contained. All properties are defined within the child elements.

Outline view displays only the textual content of a presentation. The presentation is formatted as an outline, with slide titles as the first level of the outline. Body text on slides is indented below the slide title.

[*Note*: The W3C XML Schema definition of this element's content model (CT_OutlineViewProperties) is located in §A.3. *end note*]

19.2.2.11 restoredLeft (Normal View Restored Left Properties)

This element specifies the sizing of the side content region of the normal view, when the region is of a variable restored size (neither minimized nor maximized).

Attributes	Description
autoAdjust (Auto	Specifies whether the size of the side content region should compensate for the new size

Attributes	Description
Adjust Normal View)	when resizing the window containing the view within the application. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sz (Normal View Dimension Size)	Specifies the size of the slide region (width when a child of restoredTop, height when a child of restoredLeft). The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§20.1.10.44).

[*Note*: The W3C XML Schema definition of this element's content model (CT_NormalViewPortion) is located in §A.3. *end note*]

19.2.2.12 restoredTop (Normal View Restored Top Properties)

This element specifies the sizing of the top slide region of the normal view, when the region is of a variable restored size (neither minimized nor maximized).

Attributes	Description
autoAdjust (Auto Adjust Normal View)	Specifies whether the size of the side content region should compensate for the new size when resizing the window containing the view within the application. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sz (Normal View Dimension Size)	Specifies the size of the slide region (width when a child of restoredTop, height when a child of restoredLeft). The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§20.1.10.44).

[*Note*: The W3C XML Schema definition of this element's content model (CT_NormalViewPortion) is located in §A.3. *end note*]

19.2.2.13 scale (View Scale)

This element specifies the view scaling factors that the presentation was last viewed with.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Scale2D) is located in §A.4.1. *end note*]

19.2.2.14 sld (Presentation Slide)

This element specifies a presentation slide and properties specific to the slide's appearance in outline view.

[*Example*: Consider the following presentation slide that has been collapsed in outline view.

```
<p:viewPr xmlns:a="…" xmlns:r="…" xmlns:p="…" lastView="outlineView">
  …
  <p:outlineViewPr>
    …
    <p:sldLst>
      <p:sld r:id="rId1" collapse="1"/>
    </p:sldLst>
    …
  </p:outlineViewPr>
  …
</p:viewPr>
```

end example]

Attributes	Description
collapse (Collapsed)	Specifies whether this presentation slide is to be shown as collapsed within outline view. That is, all text other than the slide title is not shown to the user. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
id (Relationship Identifier) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship identifier that is used in conjunction with a corresponding relationship file to resolve the location of this presentation slide within a presentation. 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_OutlineViewSlideEntry) is located in §A.3. *end note*]

19.2.2.15 sldLst (List of Presentation Slides)

This element specifies a list of presentation slides. A presentation slide contains the information that is specific to a single slide such as slide-specific shape and text information.

[*Note*: The W3C XML Schema definition of this element's content model (CT_OutlineViewSlideList) is located in §A.3. *end note*]

19.2.2.16 slideViewPr (Slide View Properties)

This element functions as a parent element within which all view properties associated with the slide view mode are contained. All properties are defined within the child elements.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideViewProperties) is located in §A.3. *end note*]

19.2.2.17 sorterViewPr (Slide Sorter View Properties)

This element functions as a parent element within which all view properties associated with the slide sorter view mode are contained. All properties are defined within the child elements.

The slide sorter view displays thumbnails of multiple slides at once; the number of slides and size of thumbnails depends on the scaling factor of the view.

Attributes	Description
showFormatting (Show Formatting)	Specifies whether to show associated slide formatting when in slide sorter view mode. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideSorterViewProperties) is located in §A.3. *end note*]

19.2.2.18 viewPr (Presentation-wide View Properties)

This element functions as a parent element within which all presentation-wide view properties are contained. All properties and their corresponding settings are defined within the child elements.

Attributes	Description
lastView (Last View)	Specifies the view mode that was used when the presentation document was last saved. The possible values for this attribute are defined by the ST_ViewType simple type (§19.7.55).
showComments (Show Comments)	Specifies whether the slide comments should be shown. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ViewProperties) is located in §A.3. *end note*]

19.3 Slides

The Slides portion of the PresentationML framework stores all information pertaining specifically to slides of various slide types. These slide types and corresponding parts can be broken down into three distinct parts, namely slides, embedded objects, and programmable tags.

19.3.1 Slides

Being the main segment of this section of PresentationML, the slides elements encompass all data that is to be contained within a slide. The best way to think of a slide is a container for all data that is to be on that slide. The specific shapes, images and relations within a slide do not come into play here. The elements here pertain to the six different slide types that can be described within PresentationML, namely slide, slide layout, slide master, handout master, notes master and notes slide.

19.3.1.1 bg (Slide Background)

This element specifies the background appearance information for a slide. The slide background covers the entire slide and is visible where no objects exist and as the background for transparent objects.

Attributes	Description
bwMode (Black and White Mode)	Specifies that the background should be rendered using only black and white coloring. That is, the coloring information for the background should be converted to either black or white when rendering the picture. [*Note*: No gray is to be used in rendering this background, only stark black and stark white. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§20.1.10.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Background) is located in §A.3. *end note*]

19.3.1.2 bgPr (Background Properties)

This element specifies visual effects used to render the slide background. This includes any fill, image, or effects that are to make up the background of the slide.

Attributes	Description
shadeToTitle (Shade to Title)	Specifies whether the background of the slide is of a shade to title background type. This kind of gradient fill is on the slide background and changes based on the placement of the slide title placeholder. An example is shown below.

Attributes	Description
	 The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_BackgroundProperties) is located in §A.3. *end note*]

19.3.1.3 bgRef (Background Style Reference)

This element specifies the slide background is to use a fill style defined in the style matrix. The idx attribute refers to the index of a background fill style or fill style within the presentation's style matrix, defined by the fmtScheme element. A value of 0 or 1000 indicates no background, values 1-999 refer to the index of a fill style within the fillStyleLst element, and values 1001 and above refer to the index of a background fill style within the bgFillStyleLst element. The value 1001 corresponds to the first background fill style, 1002 to the second background fill style, and so on.

[*Example*:

```
<p:bgRef idx="2">
  <a:schemeClr val="bg2"/>
</p:bgRef>
```

The above code indicates a slide background with the style's second fill style using the second background color of the color scheme.

end example]

[*Example*:

```
<p:bgRef idx="1001">
  <a:schemeClr val="bg2"/>
</p:bgRef>
```

The above code indicates a slide background with the style's first background fill style using the second background color of the color scheme.

end example]

Attributes	Description
idx (Style Matrix Index) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies the style matrix index of the style referred to. The possible values for this attribute are defined by the ST_StyleMatrixColumnIndex simple type (§20.1.10.56).

[*Note*: The W3C XML Schema definition of this element's content model (CT_StyleMatrixReference) is located in §A.4.1. *end note*]

19.3.1.4 blipFill (Picture Fill)

This element specifies the kind of picture fill that the picture object has. Because a picture has a picture fill already by default, it is possible to have two fills specified for a picture object. An example of this is shown below.

[*Example*: Consider the picture below that has a blip fill applied to it. The image used to fill this picture object has transparent pixels instead of white pixels.

```
<p:pic>
  …
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  …
</p:pic>
```


The above picture object is shown as an example of this fill type. *end example*]

[*Example*: Consider now the same picture object but with an additional gradient fill applied within the shape properties portion of the picture.

```
<p:pic>
  …
  <p:blipFill>
    <a:blip r:embed="rId2"/>
    <a:stretch>
      <a:fillRect/>
    </a:stretch>
  </p:blipFill>
  <p:spPr>
    <a:gradFill>
      <a:gsLst>
        <a:gs pos="0">
          <a:schemeClr val="tx2">
            <a:shade val="50000"/>
          </a:schemeClr>
        </a:gs>
        <a:gs pos="39999">
          <a:schemeClr val="tx2">
            <a:tint val="20000"/>
          </a:schemeClr>
        </a:gs>
        <a:gs pos="70000">
          <a:srgbClr val="C4D6EB"/>
        </a:gs>
        <a:gs pos="100000">
          <a:schemeClr val="bg1"/>
```



```
        </a:gs>
      </a:gsLst>
    </a:gradFill>
  </p:spPr>
  …
</p:pic>
```

The above picture object is shown as an example of this double fill type. *end example*]

Attributes	Description
dpi (DPI Setting) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies the DPI (dots per inch) used to calculate the size of the blip. If not present or zero, the DPI in the blip is used. [*Note*: This attribute is primarily used to keep track of the picture quality within a document. There are different levels of quality needed for print than on-screen viewing and thus a need to track this information. *end note*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rotWithShape (Rotate With Shape) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies that the fill should rotate with the shape. That is, when the shape that has been filled with a picture and the containing shape (say a rectangle) is transformed with a rotation then the fill is transformed with the same rotation. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_BlipFillProperties) is located in §A.4.1. *end note*]

19.3.1.5 bodyStyle (Slide Master Body Text Style)

This element specifies the text formatting style for all body text within a master slide. This formatting is used on all body text within presentation slides related to this master. The text formatting is specified by utilizing the DrawingML framework just as within a regular presentation slide. Within the bodyStyle element there can be many different style types defined as there are different kinds of text stored within the body of a slide.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextListStyle) is located in §A.4.1. *end note*]

19.3.1.6 clrMap (Color Scheme Map)

This element specifies the mapping layer that transforms one color scheme definition to another. Each attribute represents a color name that can be referenced in this master, and the value is the corresponding color in the theme.

[*Example*: Consider the following mapping of colors that applies to a slide master:

```
<p:clrMap bg1="dk1" tx1="lt1" bg2="dk2" tx2="lt2" accent1="accent1"
accent2="accent2" accent3="accent3" accent4="accent4" accent5="accent5"
accent6="accent6" hlink="hlink" folHlink="folHlink"/>
```

end example]

Attributes	Description
accent1 (Accent 1) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a color defined which is associated as the accent 1 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§20.1.10.14).
accent2 (Accent 2) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a color defined which is associated as the accent 2 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§20.1.10.14).
accent3 (Accent 3) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a color defined which is associated as the accent 3 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§20.1.10.14).
accent4 (Accent 4)	Specifies a color defined which is associated as the accent 4 color.

Attributes	Description
Namespace: http://purl.oclc.org/ooxml/drawingml/main	The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§20.1.10.14).
accent5 (Accent 5) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a color defined which is associated as the accent 5 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§20.1.10.14).
accent6 (Accent 6) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a color defined which is associated as the accent 6 color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§20.1.10.14).
bg1 (Background 1) Namespace: http://purl.oclc.org/ooxml/drawingml/main	A color defined which is associated as the first background color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§20.1.10.14).
bg2 (Background 2) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a color defined which is associated as the second background color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§20.1.10.14).
folHlink (Followed Hyperlink) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a color defined which is associated as the color for a followed hyperlink. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§20.1.10.14).
hlink (Hyperlink) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a color defined which is associated as the color for a hyperlink. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§20.1.10.14).
tx1 (Text 1) Namespace:	Specifies a color defined which is associated as the first text color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple

Attributes	Description
http://purl.oclc.org/ooxml/drawingml/main	type (§20.1.10.14).
tx2 (Text 2) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a color defined which is associated as the second text color. The possible values for this attribute are defined by the ST_ColorSchemeIndex simple type (§20.1.10.14).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ColorMapping) is located in §A.4.1. *end note*]

19.3.1.7 clrMapOvr (Color Scheme Map Override)

This element provides a mechanism with which to override the color schemes listed within the ClrMap element. If the masterClrMapping element is present, the color scheme defined by the master is used. If the overrideClrMapping element is present, it defines a new color scheme specific to the parent notes slide, presentation slide, or slide layout.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ColorMappingOverride) is located in §A.4.1. *end note*]

19.3.1.8 cNvCxnSpPr (Non-Visual Connector Shape Drawing Properties)

This element specifies the non-visual drawing properties specific to a connector shape. This includes information specifying the shapes to which the connector shape is connected.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NonVisualConnectorProperties) is located in §A.4.1. *end note*]

19.3.1.9 cNvGraphicFramePr (Non-Visual Graphic Frame Drawing Properties)

This element specifies the non-visual drawing properties for a graphic frame. These non-visual properties are properties that the generating application would utilize when rendering the slide surface.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NonVisualGraphicFrameProperties) is located in §A.4.1. *end note*]

19.3.1.10 cNvGrpSpPr (Non-Visual Group Shape Drawing Properties)

This element specifies the non-visual drawing properties for a group shape. These non-visual properties are properties that the generating application would utilize when rendering the slide surface.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NonVisualGroupDrawingShapeProps) is located in §A.4.1. *end note*]

19.3.1.11 cNvPicPr (Non-Visual Picture Drawing Properties)

This element specifies the non-visual properties for the picture canvas. These properties are to be used by the generating application to determine how certain properties are to be changed for the picture object in question.

[*Example*: Consider the following DrawingML.

```
<p:pic>
  …
  <p:nvPicPr>
    <p:cNvPr id="4" name="Lilly_by_Lisher.jpg"/>
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
    </p:cNvPicPr>
    <p:nvPr/>
  </p:nvPicPr>
  …
</p:pic>
```

end example]

Attributes	Description
preferRelativeResize (Relative Resize Preferred) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies if the user interface should show the resizing of the picture based on the picture's current size or its original size. If this attribute is set to true, then scaling is relative to the original picture size as opposed to the current picture size. [*Example*: Consider the case where a picture has been resized within a document and is now 50% of the originally inserted picture size. Now if the user chooses to make a later adjustment to the size of this picture within the generating application, then the value of this attribute should be checked. If this attribute is set to true then a value of 50% is shown. Similarly, if this attribute is set to false, then a value of 100% should be shown because the picture has not yet been resized from its current (smaller) size. *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NonVisualPictureProperties) is located in §A.4.1. *end note*]

19.3.1.12 cNvPr (Non-Visual Drawing Properties)

This element specifies non-visual canvas properties. This allows for additional information that does not affect the appearance of the picture to be stored.

[*Example*: Consider the following DrawingML.

```
<p:pic>
  …
  <p:nvPicPr>
    <p:cNvPr id="4" name="Lilly_by_Lisher.jpg"/>
  </p:nvPicPr>
  …
</p:pic>
```

end example]

Attributes	Description
descr (Alternative Text for Object) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies alternative text for the current DrawingML object, for use by assistive technologies or applications which do not display the current object. If this element is omitted, then no alternative text is present for the parent object. [*Example*: Consider a DrawingML object defined as follows: `<… descr="A picture of a bowl of fruit">` The descr attribute contains alternative text which can be used in place of the actual DrawingML object. *end example*] The possible values for this attribute are defined by the W3C XML Schema string datatype.
hidden (Hidden) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies whether this DrawingML object is displayed. When a DrawingML object is displayed within a document, that object can be hidden (i.e., present, but not visible). This attribute determines whether the object is rendered or made hidden. [*Note*: An application can have settings which allow this object to be viewed. *end note*] If this attribute is omitted, then the parent DrawingML object shall be displayed (i.e., not hidden). [*Example*: Consider an inline DrawingML object which must be hidden within the document's content. This setting would be specified as follows: `<… hidden="true" />` The hidden attribute has a value of `true`, which specifies that the DrawingML object is hidden and not displayed when the document is displayed. *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
id (Unique	Specifies a unique identifier for the current DrawingML object within the current

Attributes	Description
Identifier) Namespace: http://purl.oclc.org/ooxml/drawingml/main	document. This ID can be used to assist in uniquely identifying this object so that it can be referred to by other parts of the document. If multiple objects within the same document share the same id attribute value, then the document shall be considered non-conformant. [*Example*: Consider a DrawingML object defined as follows: `<… id="10" … >` The id attribute has a value of `10`, which is the unique identifier for this DrawingML object. *end example*] The possible values for this attribute are defined by the ST_DrawingElementId simple type (§20.1.10.21).
name (Name) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies the name of the object. [*Note*: Typically, this is used to store the original file name of a picture object. *end note*] [*Example*: Consider a DrawingML object defined as follows: `< … name="foo.jpg" >` The name attribute has a value of `foo.jpg`, which is the name of this DrawingML object. *end example*] The possible values for this attribute are defined by the W3C XML Schema string datatype.
title (Title) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies the title (caption) of the current DrawingML object. If this attribute is omitted, then no title text is present for the parent object. [*Example*: Consider a DrawingML object defined as follows: `<… title="Process Flow Diagram">` *end example*] The possible values for this attribute are defined by the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NonVisualDrawingProps) is located in §A.4.1. *end note*]

19.3.1.13 cNvSpPr (Non-Visual Drawing Properties for a Shape)

This element specifies the non-visual drawing properties for a shape. These properties are to be used by the generating application to determine how the shape should be dealt with

[*Example*: Consider the shape that has a shape lock applied to it.

```
<p:sp>
  <p:nvSpPr>
    <p:cNvPr id="2" name="Rectangle 1"/>
    <p:cNvSpPr>
      <a:spLocks noGrp="1"/>
    </p:cNvSpPr>
   </p:nvSpPr>
   …
</p:sp>
```

This shape lock is stored within the non-visual drawing properties for this shape. *end example*]

Attributes	Description
txBox (Text Box) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies that the corresponding shape is a text box and thus should be treated as such by the generating application. If this attribute is omitted then it is assumed that the corresponding shape is not specifically a text box. [*Note*: Because a shape is not specified to be a text box does not mean that it cannot have text attached to it. A text box is merely a specialized shape with specific properties. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NonVisualDrawingShapeProps) is located in §A.4.1. *end note*]

19.3.1.14 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.

[*Note*: For better interoperability, only standard XML formats should be used. *end note*]

[*Example*: Consider a PresentationML document which includes the following SMIL markup in a part named smil1.xml:

```
<!--
            Copyright: Copyright 1998-2001 W3C (MIT, INRIA, Keio), All Rights
Reserved.
See http://www.w3.org/Consortium/Legal/.
Author: Aaron Cohen (Intel)
           Version: February 7, 2001
            Module: Animation Module
           Feature: animation
         File Name: animation-add-BE-05.smil
  Media Components: none
 Expected Behavior: Nine red rectangles numbered 1 to 9 shrink to squares
                    over 2s as follows:
                    at 2s #1 shrinks.
                    at 5s #2 shrinks, 1s after #1 completes
                    at 8s #3 shrinks.
                    #4 shrinks when it is clicked on.
                    #5 shrinks 1s after it is clicked on.
                    #6 shrinks 2s after it is clicked on.
                    #7 shrinks when the accesskey '1' is pressed.
                    #8 should be shrunk from 0s since it's wallclock time is in
the past.
                    #9 will not shrink unless a DOM call causes it to begin.
-->
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <layout>
      <root-layout width="640" height="480" backgroundColor="white"/>
      <region id="whole" width="640" height="480" z-index="0"/>
      <region id="rect1" top="50px" left="90px" height="50px" width="30px"
backgroundColor="red" z-index="1"/>
      <region id="rect2" top="50px" left="234px" height="50px" width="30px"
backgroundColor="red" z-index="1"/>
```



```
      <region id="rect4" top="160px" left="90px" height="50px" width="30px"
backgroundColor="transparent" z-index="1"/>
      <region id="rect5" top="160px" left="234px" height="50px" width="30px"
backgroundColor="transparent" z-index="1"/>
      <region id="rect6" top="160px" left="380px" height="50px" width="30px"
backgroundColor="transparent" z-index="1"/>
      <region id="rect7" top="270px" left="90px" height="50px" width="30px"
backgroundColor="red" z-index="1"/>
      <region id="rect8" top="270px" left="234px" height="50px" width="30px"
backgroundColor="red" z-index="1"/>
      <region id="rect9" top="270px" left="380px" height="50px" width="30px"
backgroundColor="red" z-index="1"/>
    </layout>
  </head>
<!-- Copyright 1998-2001 W3C (MIT, INRIA, Keio), All Rights Reserved.
See http://www.w3.org/Consortium/Legal/. -->
  <body>
    <par dur="indefinite">
      <img src="../images/animation-timing-BE-05.jpg" region="whole"/>
      <animate id="anim1" targetElement="rect1" attributeName="height" from="50"
to="25" begin="2s" dur="2s" fill="freeze"/>
      <animate id="anim2" targetElement="rect2" attributeName="height" from="50"
to="25" begin="anim1.end+1s" dur="2s" fill="freeze"/>
      <brush id="brush4" color="red" region="rect4" height="50px" width="30px"/>
      <animate id="anim4" targetElement="brush4" attributeName="height"
from="50" to="25" begin="brush4.activateEvent" dur="2s" fill="freeze"/>
      <brush id="brush5" color="red" region="rect5" height="50px" width="30px"/>
      <animate id="anim5" targetElement="brush5" attributeName="height"
from="50" to="25" begin="brush5.activateEvent+1s" dur="2s" fill="freeze"/>
      <brush id="brush6" color="red" region="rect6" height="50px" width="30px"/>
      <animate id="anim6a" targetElement="brush6" attributeName="width"
repeatCount="3" from="30" to="30" begin="brush6.activateEvent" dur="1s"
fill="freeze"/>
      <animate id="anim6b" targetElement="brush6" attributeName="height"
from="50" to="25" begin="anim6a.repeat(2)" dur="2s" fill="freeze"/>
      <animate id="anim7" targetElement="rect7" attributeName="height" from="50"
to="25" begin="accesskey(1)" dur="2s" fill="freeze"/>
      <animate id="anim8" targetElement="rect8" attributeName="height" from="50"
to="25" begin="wallclock(2000-01-01T00:00:00Z)" dur="2s" fill="freeze"/>
      <animate id="anim9" targetElement="rect9" attributeName="width" from="30"
to="30" begin="indefinite" dur="1s" fill="freeze"/>
    </par>
  </body>
```



```
</smil>
```

A Slide Part would reference this content as follows:

```
<p:spTree>
  …
  <p:contentPart r:id="smil01"/>
  …
</p:spTree>
```

The contentPart element specifies that the content targeted by the relationship with an ID of `smil01` is part of the PresentationML document. Examining the contents of the corresponding relationship part item, we can see the targets for that relationship:

```
<Relationships … >
  …
  <Relationship Id="smil01" TargetMode="Internal"
Type="http://purl.oclc.org/ooxml/officeDocument/relationships/customXml"
Target="smil1.xml" />
  …
</Relationships>
```

The corresponding relationship part item shows that the SMIL content is located next to the slide and is named `smil1.xml`. *end example*]

ttributes	Description
id (Relationship to Part) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship ID to a content part. [*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` 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.3. *end note*]

19.3.1.15 controls (List of controls)

This element specifies a list of embedded controls for the corresponding slide. Custom embedded controls can be embedded on slides.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ControlList) is located in §A.3. *end note*]

19.3.1.16 cSld (Common Slide Data)

This element specifies a container for slide information that is relevant to all of the slide types. All slides share a common set of properties that is independent of the slide type; the description of these properties for any particular slide is stored within the slide's cSld container. Slide data specific to the slide type indicated by the parent element is stored elsewhere.

[*Note*: The actual data in cSld describe only the particular parent slide; it is only the kind of information stored that is common across all slides. *end note*]

[*Example*: Consider the following PresentationML slide

```
<p:sld>
  <p:cSld>
    <p:spTree>
      …
    </p:spTree>
  </p:cSld>
  …
</p:sld>
```

As the above example shows, the shape tree of a slide (spTree) is a child element of cSld because all slide types can contain a shape tree. Other slide properties specific to the slide type (such as transitions for sld slides) are specified elsewhere. *end example*]

Attributes	Description
name (Name)	Specifies the slide name property that is used to further identify this unique configuration of common slide data. This might be used to aid in distinguishing different slide layouts or various other slide types. The possible values for this attribute are defined by the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CommonSlideData) is located in §A.3. *end note*]

19.3.1.17 custData (Customer Data)

This element specifies customer data which allows for the specifying and persistence of customer specific data within the presentation.

Attributes	Description
id (Relationship ID) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	This attribute specifies the relationship id for referencing other resources outside the scope of the current PresentationML file. 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_CustomerData) is located in §A.3. *end note*]

19.3.1.18 custDataLst (Customer Data List)

This element allows for the specifying of customer defined data within the PresentationML framework. References to custom data or tags can be defined within this list.

[*Note*: The W3C XML Schema definition of this element's content model (CT_CustomerDataList) is located in §A.3. *end note*]

19.3.1.19 cxnSp (Connection Shape)

This element specifies a connection shape that is used to connect two sp elements. Once a connection is specified using a cxnSp, it is left to the generating application to determine the exact path the connector takes. That is the connector routing algorithm is left up to the generating application as the desired path might be different depending on the specific needs of the application.

[*Example*: Consider the following connector shape that connects two regular shapes.

```
<p:spTree>
  …
  <p:sp>
    <p:nvSpPr>
      <p:cNvPr id="1" name="Rectangle 1"/>
      <p:cNvSpPr/>
      <p:nvPr/>
    </p:nvSpPr>
    …
  </p:sp>
  <p:sp>
    <p:nvSpPr>
```



```
      <p:cNvPr id="2" name="Rectangle 2"/>
      <p:cNvSpPr/>
      <p:nvPr/>
    </p:nvSpPr>
    …
  </p:sp>
  <p:cxnSp>
    <p:nvCxnSpPr>
      <p:cNvPr id="3" name="Elbow Connector 3"/>
      <p:cNvCxnSpPr>
        <a:stCxn id="1" idx="3"/>
        <a:endCxn id="2" idx="1"/>
      </p:cNvCxnSpPr>
      <p:nvPr/>
    </p:nvCxnSpPr>
    …
  </p:cxnSp>
</p:spTree>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Connector) is located in §A.3. *end note*]

19.3.1.20 extLst (Extension List with Modification Flag)

This element specifies the extension list with modification ability within which all future extensions of element type ext are defined. The extension list along with corresponding future extensions is used to extend the storage capabilities of the PresentationML framework. This allows for various new kinds of data to be stored natively within the framework.

[*Note*: Using this extLst element allows the generating application to store whether this extension property has been modified. *end note*]

Attributes	Description
mod (Modify)	This attribute specifies whether the data contained within this element has been modified and should thus be processed again by the generating application. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ExtensionListModify) is located in §A.3. *end note*]

19.3.1.21 graphicFrame (Graphic Frame)

This element specifies the existence of a graphics frame. This frame contains a graphic that was generated by an external source and needs a container in which to be displayed on the slide surface.

Attributes	Description
bwMode (Black and White Mode) Namespace: .../drawingml/2006/main	Specifies how the graphical object should be rendered, using color, black or white, or grayscale. [*Note*: This does not mean that the graphical object itself is stored with only black and white or grayscale information. This attribute instead sets the rendering mode that the graphical object uses. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§20.1.10.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_GraphicalObjectFrame) is located in §A.3. *end note*]

19.3.1.22 grpSp (Group Shape)

This element specifies a group shape that represents many shapes grouped together. This shape is to be treated just as if it were a regular shape but instead of being described by a single geometry it is made up of all the shape geometries encompassed within it. Within a group shape each of the shapes that make up the group are specified just as they normally would. The idea behind grouping elements however is that a single transform can apply to many shapes at the same time.

[*Example*: Consider the following group shape.

```
<p:grpSp>
  <p:nvGrpSpPr>
    <p:cNvPr id="10" name="Group 9"/>
    <p:cNvGrpSpPr/>
    <p:nvPr/>
  </p:nvGrpSpPr>
  <p:grpSpPr>
    <a:xfrm>
      <a:off x="838200" y="990600"/>
      <a:ext cx="2426208" cy="978408"/>
      <a:chOff x="838200" y="990600"/>
      <a:chExt cx="2426208" cy="978408"/>
    </a:xfrm>
  </p:grpSpPr>
  <p:sp>
    …
```



```
    </p:sp>
    <p:sp>
      …
    </p:sp>
    <p:sp>
      …
    </p:sp>
  </p:grpSp>
```

In the above example we see three shapes specified within a single group. These three shapes have their position and sizes specified just as they normally would within the shape tree. The generating application should apply the transformation after the bounding box for the group shape has been calculated. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_GroupShape) is located in §A.3. *end note*]

19.3.1.23 grpSpPr (Group Shape Properties)

This element specifies the properties that are to be common across all of the shapes within the corresponding group. If there are any conflicting properties within the group shape properties and the individual shape properties then the individual shape properties should take precedence.

Attributes	Description
bwMode (Black and White Mode) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies that the group shape should be rendered using only black and white coloring. That is the coloring information for the group shape should be converted to either black or white when rendering the corresponding shapes. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the group shapes themselves are stored with only black and white color information. This attribute instead sets the rendering mode that the shapes use when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§20.1.10.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_GroupShapeProperties) is located in §A.4.1. *end note*]

19.3.1.24 handoutMaster (Handout Master)

This element specifies an instance of a handout master slide. Within a handout master slide are contained all elements that describe the objects and their corresponding formatting for within a handout slide. Within a handout master slide the cSld element specifies the common slide elements such as shapes and their attached text bodies. There are other properties within a handout master slide but cSld encompasses the majority of the intended purpose for a handoutMaster slide.

[*Note*: The W3C XML Schema definition of this element's content model (CT_HandoutMaster) is located in §A.3. *end note*]

19.3.1.25 hf (Header/Footer information for a slide master)

This element specifies the header and footer information for a slide. Headers and footers consist of placeholders for text that should be consistent across all slides and slide types, such as a date and time, slide numbering, and custom header and footer text.

Attributes	Description
dt (Date/Time Placeholder)	Specifies whether the Date/Time placeholder is enabled for this master. If this attribute is not specified, a value of true should be assumed by the generating application. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
ftr (Footer Placeholder)	Specifies whether the Footer placeholder is enabled for this master. If this attribute is not specified, a value of true should be assumed by the generating application. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
hdr (Header Placeholder)	Specifies whether the Header placeholder is enabled for this master. If this attribute is not specified, a value of true should be assumed by the generating application. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
sldNum (Slide Number Placeholder)	Specifies whether the slide number placeholder is enabled. If this attribute is not specified, a value of true should be assumed by the generating application. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_HeaderFooter) is located in §A.3. *end note*]

19.3.1.26 notes (Notes Slide)

This element specifies the existence of a notes slide along with its corresponding data. Contained within a notes slide are all the common slide elements along with addition properties that are specific to the notes element.

[*Example*: Consider the following PresentationML notes slide

```
<p:notes>
  <p:cSld>
```



```
    …
  </p:cSld>
  …
</p:notes>
```

In the above example a notes element specifies the existence of a notes slide with all of its parts. Notice the cSld element, that specifies the common elements that can appear on any slide type and then any elements specify additional non-common properties for this notes slide. *end example*]

Attributes	Description
showMasterPhAnim (Show Master Placeholder Animations)	Specifies whether or not to display animations on placeholders from the master slide. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showMasterSp (Show Master Shapes)	Specifies if shapes on the master slide should be shown on slides or not. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NotesSlide) is located in §A.3. *end note*]

19.3.1.27 notesMaster (Notes Master)

This element specifies an instance of a handout master slide. Within a handout master slide are contained all elements that describe the objects and their corresponding formatting for within a handout slide. Within a handout master slide the cSld element specifies the common slide elements such as shapes and their attached text bodies. There are other properties within a handout master slide but cSld encompasses the majority of the intended purpose for a handoutMaster slide.

[*Note*: The W3C XML Schema definition of this element's content model (CT_NotesMaster) is located in §A.3. *end note*]

19.3.1.28 notesStyle (Notes Text Style)

This element specifies the text formatting style for the all other text within a notes slide. This formatting is used on all text within the corresponding notes slides. The text formatting is specified by utilizing the DrawingML framework just as within a regular presentation slide. Within the notesStyle element there can be many different style types defined as there are different kinds of text stored within a notes slide.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextListStyle) is located in §A.4.1. *end note*]

19.3.1.29 nvCxnSpPr (Non-Visual Properties for a Connection Shape)

This element specifies all non-visual properties for a connection shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a connection shape. This allows for additional information that does not affect the appearance of the connection shape to be stored.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ConnectorNonVisual) is located in §A.3. *end note*]

19.3.1.30 nvGraphicFramePr (Non-Visual Properties for a Graphic Frame)

This element specifies all non-visual properties for a graphic frame. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a graphic frame. This allows for additional information that does not affect the appearance of the graphic frame to be stored.

[*Note*: The W3C XML Schema definition of this element's content model (CT_GraphicalObjectFrameNonVisual) is located in §A.3. *end note*]

19.3.1.31 nvGrpSpPr (Non-Visual Properties for a Group Shape)

This element specifies all non-visual properties for a group shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a group shape. This allows for additional information that does not affect the appearance of the group shape to be stored.

[*Note*: The W3C XML Schema definition of this element's content model (CT_GroupShapeNonVisual) is located in §A.3. *end note*]

19.3.1.32 nvPicPr (Non-Visual Properties for a Picture)

This element specifies all non-visual properties for a picture. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a picture. This allows for additional information that does not affect the appearance of the picture to be stored.

[*Example*: Consider the following PresentationML.

```
<p:pic>
  …
  <p:nvPicPr>
    …
  </p:nvPicPr>
  …
</p:pic>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_PictureNonVisual) is located in §A.3. *end note*]

19.3.1.33 nvPr (Non-Visual Properties)

This element specifies non-visual properties for objects. These properties include multimedia content associated with an object and properties indicating how the object is to be used or displayed in different contexts.

Attributes	Description
isPhoto (Is a Photo Album)	Specifies whether the picture belongs to a photo album and should thus be included when editing a photo album within the generating application. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
userDrawn (Is User Drawn)	Specifies if the corresponding object has been drawn by the user and should thus not be deleted. This allows for the flagging of slides that contain user drawn data. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ApplicationNonVisualDrawingProps) is located in §A.3. *end note*]

19.3.1.34 nvSpPr (Non-Visual Properties for a Shape)

This element specifies all non-visual properties for a shape. This element is a container for the non-visual identification properties, shape properties and application properties that are to be associated with a shape. This allows for additional information that does not affect the appearance of the shape to be stored.

[*Note*: The W3C XML Schema definition of this element's content model (CT_ShapeNonVisual) is located in §A.3. *end note*]

19.3.1.35 otherStyle (Slide Master Other Text Style)

This element specifies the text formatting style for the all other text within a master slide. This formatting is used on all text not covered by the titleStyle or bodyStyle elements within related presentation slides. The text formatting is specified by utilizing the DrawingML framework just as within a regular presentation slide. Within the otherStyle element there can be many different style types defined as there are different kinds of text stored within a slide.

[*Note*: The otherStyle element is to be used for specifying the text formatting of text within a slide shape but not within a text box. Text box styling is handled from within the bodyStyle element. *end note*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextListStyle) is located in §A.4.1. *end note*]

19.3.1.36 ph (Placeholder Shape)

This element specifies that the corresponding shape should be represented by the generating application as a placeholder. When a shape is considered a placeholder by the generating application it can have special properties to alert the user that they can enter content into the shape. Different placeholder types are allowed and can be specified by using the placeholder type attribute for this element.

Attributes	Description
hasCustomPrompt (Placeholder has custom prompt)	Specifies whether the corresponding placeholder should have a custom prompt or not. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
idx (Placeholder Index)	Specifies the placeholder index. This is used when applying templates or changing layouts to match a placeholder on one template/master to another. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
orient (Placeholder Orientation)	Specifies the orientation of a placeholder. The possible values for this attribute are defined by the ST_Direction simple type (§19.7.2).
sz (Placeholder Size)	Specifies the size of a placeholder. The possible values for this attribute are defined by the ST_PlaceholderSize simple type (§19.7.9).
type (Placeholder Type)	Specifies what content type a placeholder is intended to contain. The possible values for this attribute are defined by the ST_PlaceholderType simple type (§19.7.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Placeholder) is located in §A.3. *end note*]

19.3.1.37 pic (Picture)

This element specifies the existence of a picture object within the document.

[*Example*: Consider the following PresentationML that specifies the existence of a picture within a document. This picture can have non-visual properties, a picture fill as well as shape properties attached to it.

```
<p:pic>
  <p:nvPicPr>
    <p:cNvPr id="4" name="lake.JPG" descr="Picture of a Lake" />
    <p:cNvPicPr>
      <a:picLocks noChangeAspect="1"/>
```



```
      </p:cNvPicPr>
      <p:nvPr/>
    </p:nvPicPr>
    <p:blipFill>
      …
    </p:blipFill>
    <p:spPr>
      …
    </p:spPr>
  </p:pic>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Picture) is located in §A.3. *end note*]

19.3.1.38 sld (Presentation Slide)

This element specifies a slide within a slide list. The slide list is used to specify an ordering of slides.

[*Example*: Consider the following custom show with an ordering of slides.

```
  <p:custShowLst>
    <p:custShow name="Custom Show 1" id="0">
      <p:sldLst>
        <p:sld r:id="rId4"/>
        <p:sld r:id="rId3"/>
        <p:sld r:id="rId2"/>
        <p:sld r:id="rId5"/>
      </p:sldLst>
    </p:custShow>
  </p:custShowLst>
```

In the above example the order specified to present the slides is slide 4, then 3, 2 and finally 5. *end example*]

Attributes	Description
show (Show Slide in Slide Show)	Specifies that the current slide should be shown in slide show. If this attribute is omitted then a value of true is assumed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showMasterPhAnim (Show Master Placeholder	Specifies whether or not to display animations on placeholders from the master slide. The possible values for this attribute are defined by the W3C XML Schema boolean

Attributes	Description
Animations)	datatype.
showMasterSp (Show Master Shapes)	Specifies if shapes on the master slide should be shown on slides or not. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Slide) is located in §A.3. *end note*]

19.3.1.39 sldLayout (Slide Layout)

This element specifies an instance of a slide layout. The slide layout contains in essence a template slide design that can be applied to any existing slide. When applied to an existing slide all corresponding content should be mapped to the new slide layout.

Attributes	Description
matchingName (Matching Name)	Specifies a name to be used in place of the name attribute within the cSld element. This is used for layout matching in response to layout changes and template applications. The possible values for this attribute are defined by the W3C XML Schema string datatype.
preserve (Preserve Slide Layout)	Specifies whether the corresponding slide layout is deleted when all the slides that follow that layout are deleted. If this attribute is not specified then a value of false should be assumed by the generating application. This would mean that the slide would in fact be deleted if no slides within the presentation were related to it. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showMasterPhAnim (Show Master Placeholder Animations)	Specifies whether or not to display animations on placeholders from the master slide. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
showMasterSp (Show Master Shapes)	Specifies if shapes on the master slide should be shown on slides or not. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
type (Slide Layout Type)	Specifies the slide layout type that is used by this slide. The possible values for this attribute are defined by the ST_SlideLayoutType simple type (§19.7.15).
userDrawn (Is User Drawn)	Specifies if the corresponding object has been drawn by the user and should thus not be deleted. This allows for the flagging of slides that contain user drawn data.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideLayout) is located in §A.3. *end note*]

19.3.1.40 sldLayoutId (Slide Layout Id)

This element specifies the relationship information for each slide layout that is used within the slide master. The slide master has relationship identifiers that it uses internally for determining the slide layouts that should be used. Then, to resolve what these slide layouts should be the sldLayoutId elements in the sldLayoutIdLst are utilized.

Attributes	Description
id (ID Tag) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship id value that the generating application can use to resolve which slide layout is used in the creation of the slide. This relationship id is used within the relationship file for the master slide to expose the location of the corresponding layout file within the presentation. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
id (ID Tag)	Specifies the identification number that uniquely identifies this slide layout within the presentation file. The possible values for this attribute are defined by the ST_SlideLayoutId simple type (§19.7.14).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideLayoutIdListEntry) is located in §A.3. *end note*]

19.3.1.41 sldLayoutIdLst (List of Slide Layouts)

This element specifies the existence of the slide layout identification list. This list is contained within the slide master and is used to determine which layouts are being used within the slide master file. Each layout within the list of slide layouts has its own identification number and relationship identifier that uniquely identifies it within both the presentation document and the particular master slide within which it is used.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideLayoutIdList) is located in §A.3. *end note*]

19.3.1.42 sldMaster (Slide Master)

This element specifies an instance of a slide master slide. Within a slide master slide are contained all elements that describe the objects and their corresponding formatting for within a presentation slide. Within a slide master slide are two main elements. The cSld element specifies the common slide elements such as shapes and their attached text bodies. Then the txStyles element specifies the formatting for the text within each of these shapes. The other properties within a slide master slide specify other properties for within a presentation slide such as color information, headers and footers, as well as timing and transition information for all corresponding presentation slides.

Attributes	Description
preserve (Preserve Slide Master)	Specifies whether the corresponding slide layout is deleted when all the slides that follow that layout are deleted. If this attribute is not specified then a value of false should be assumed by the generating application. This would mean that the slide would in fact be deleted if no slides within the presentation were related to it. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideMaster) is located in §A.3. *end note*]

19.3.1.43 sp (Shape)

This element specifies the existence of a single shape. A shape can either be a preset or a custom geometry, defined using the DrawingML framework. In addition to a geometry each shape can have both visual and non-visual properties attached. Text and corresponding styling information can also be attached to a shape. This shape is specified along with all other shapes within either the shape tree or group shape elements.

[*Note*: Shapes are the preferred mechanism for specifying text on a slide. *end note*]

Attributes	Description
useBgFill (Use Background Fill)	Specifies that the shape fill should be set to that of the slide background surface. [*Note*: This attribute does not set the fill of the shape to be transparent but instead sets it to be filled with the portion of the slide background that is directly behind it. *end note*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Shape) is located in §A.3. *end note*]

19.3.1.44 spPr (Shape Properties)

This element specifies the visual shape properties that can be applied to a shape. These properties include the shape fill, outline, geometry, effects, and 3D orientation.

Attributes	Description
bwMode (Black and White Mode) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies that the picture should be rendered using only black and white coloring. That is the coloring information for the picture should be converted to either black or white when rendering the picture. No gray is to be used in rendering this image, only stark black and stark white. [*Note*: This does not mean that the picture itself that is stored within the file is necessarily a black and white picture. This attribute instead sets the rendering mode that the picture has applied to when rendering. *end note*] The possible values for this attribute are defined by the ST_BlackWhiteMode simple type (§20.1.10.10).

[*Note*: The W3C XML Schema definition of this element's content model (CT_ShapeProperties) is located in §A.4.1. *end note*]

19.3.1.45 spTree (Shape Tree)

This element specifies all shape-based objects, either grouped or not, that can be referenced on a given slide. As most objects within a slide are shapes, this represents the majority of content within a slide. Text and effects are attached to shapes that are contained within the spTree element.

[*Example*: Consider the following PresentationML slide

```
<p:sld>
  <p:cSld>
    <p:spTree>
      <p:nvGrpSpPr>
        …
      </p:nvGrpSpPr>
      <p:grpSpPr>
        …
      </p:grpSpPr>
      <p:sp>
        …
      </p:sp>
    </p:spTree>
  </p:cSld>
  …
```

```
</p:sld>
```

In the above example the shape tree specifies all the shape properties for this slide. *end example*]

Each shape-based object within the shape tree, whether grouped or not, shall represent one unique level of z-ordering on the slide. The z-order for each shape-based object shall be determined by the lexical ordering of each shape-based object within the shape tree: the first shape-based object shall have the lowest z-order, while the last shape-based object shall have the highest z-order.

The z-ordering of shape-based objects within the shape tree shall also determine the navigation (tab) order of the shape-based objects: the shape-based object with the lowest z-order (the first shape in lexical order) shall be first in navigation order, with objects being navigated in ascending z-order.

[*Example*: Consider the following PresentationML slide with two shapes

```
<p:sld>
  <p:cSld>
    <p:spTree>
      …
      <p:sp>
        <p:nvSpPr>
          <p:cNvPr id="5" name="Oval 4" />
          …
        </p:nvSpPr>
        …
      </p:sp>
      <p:sp>
        <p:nvSpPr>
          <p:cNvPr id="4" name="Isosceles Triangle 3" />
          …
        </p:nvSpPr>
        …
      </p:sp>
    </p:spTree>
  </p:cSld>
  …
</p:sld>
```

In the above example the shape with name `Oval 4` has the lowest z-order value since that shape is the first shape in the shape tree. `Oval 4` is also the first shape in navigation order. The shape with name `Isosceles Triangle 3` has the highest z positioning value since that shape is the last shape in the shape tree. `Isosceles Triangle 3` is also the last shape in navigation order. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_GroupShape) is located in §A.3. *end note*]

19.3.1.46 style (Shape Style)

This element specifies the style information for a shape. This is used to define a shape's appearance in terms of the preset styles defined by the style matrix for the theme.

[*Example*:

```
<p:style>
  <a:lnRef idx="3">
    <a:schemeClr val="lt1"/>
  </a:lnRef>
  <a:fillRef idx="1">
    <a:schemeClr val="accent3"/>
  </a:fillRef>
  <a:effectRef idx="1">
    <a:schemeClr val="accent3"/>
  </a:effectRef>
  <a:fontRef idx="minor">
    <a:schemeClr val="lt1"/>
  </a:fontRef>
</p:style>
```

The parent shape of the above code is to have an outline that uses the third line style defined by the theme, use the first fill defined by the scheme, and be rendered with the first effect defined by the theme. Text inside the shape is to use the minor font defined by the theme.

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_ShapeStyle) is located in §A.4.1. *end note*]

19.3.1.47 tags (Customer Data Tags)

This element specifies the existence of customer data in the form of tags. This allows for the storage of customer data within the PresentationML framework. While this is similar to the ext tag in that it can be used store information, this tag mainly focuses on referencing to other parts of the presentation document. This is accomplished via the relationship identification attribute that is required for all specified tags.

Attributes	Description
id (Relationship ID) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	This attribute specifies the relationship identifier for the customer data tag. This allows for a link to a resource that is external from the current XML document but still contained within the presentation document. 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_TagsData) is located in §A.3. *end note*]

19.3.1.48 timing (Slide Timing Information for a Slide Layout)

This element specifies the timing information for handling all animations and timed events within the corresponding slide. This information is tracked via time nodes within the timing element. More information on the specifics of these time nodes and how they are to be defined can be found within the Animation section of the PresentationML framework.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideTiming) is located in §A.3. *end note*]

19.3.1.49 titleStyle (Slide Master Title Text Style)

This element specifies the text formatting style for the title text within a master slide. This formatting is used on all title text within related presentation slides. The text formatting is specified by utilizing the DrawingML framework just as within a regular presentation slide. Within a title style there can be many different style types defined as there are different kinds of text stored within a slide title.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextListStyle) is located in §A.4.1. *end note*]

19.3.1.50 transition (Slide Transition for a Slide Layout)

This element specifies the kind of slide transition that should be used to transition to the current slide from the previous slide. That is, the transition information is stored on the slide that appears after the transition is complete.

Attributes	Description
advClick (Advance on Click)	Specifies whether a mouse click advances the slide or not. If this attribute is not specified then a value of true is assumed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
advTm (Advance after time)	Specifies the time, in milliseconds, after which the transition should start. This setting can be used in conjunction with the advClick attribute. If this attribute is not specified then it is assumed that no auto-advance occurs. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
spd (Transition Speed)	Specifies the transition speed that is to be used when transitioning from the current slide to the next. The possible values for this attribute are defined by the ST_TransitionSpeed simple type (§19.7.54).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideTransition) is located in §A.3. *end note*]

19.3.1.51 txBody (Shape Text Body)

This element specifies the existence of text to be contained within the corresponding shape. All visible text and visible text related properties are contained within this element. There can be multiple paragraphs and within paragraphs multiple runs of text.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TextBody) is located in §A.4.1. *end note*]

19.3.1.52 txStyles (Slide Master Text Styles)

This element specifies the text styles within a slide master. Within this element is the styling information for title text, the body text and other slide text as well. This element is only for use within the Slide Master and thus sets the text styles for the corresponding presentation slides.

[*Example*: Consider the case where we would like to specify the title text for a master slide.

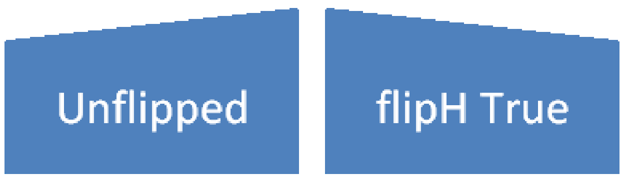
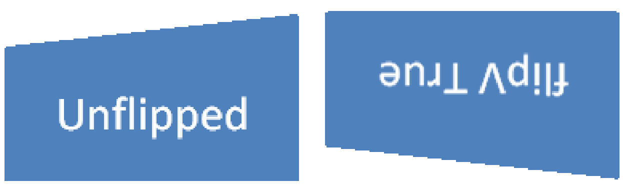
```
<p:txStyles>
  <p:titleStyle>
    <a:lvl1pPr algn="ctr" rtl="0" latinLnBrk="0">
      <a:spcBef>
        <a:spcPct val="0"/>
      </a:spcBef>
      <a:buNone/>
      <a:defRPr sz="4400" kern="1200">
        <a:solidFill>
          <a:schemeClr val="tx1"/>
        </a:solidFill>
        <a:latin typeface="+mj-lt"/>
        <a:ea typeface="+mj-ea"/>
        <a:cs typeface="+mj-cs"/>
      </a:defRPr>
    </a:lvl1pPr>
  </p:titleStyle>
</p:txStyles>
```

In the above example the title text is set according to the above formatting for all related slides within the presentation. *end example*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideMasterTextStyles) is located in §A.3. *end note*]

19.3.1.53 xfrm (2D Transform for Graphic Frame)

This element specifies the transform to be applied to the corresponding graphic frame. This transformation is applied to the graphic frame just as it would be for a shape or group shape.

Attributes	Description
flipH (Horizontal Flip) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a horizontal flip. When true, this attribute defines that the shape is flipped horizontally about the center of its bounding box. [*Example*: The following illustrates the effect of a horizontal flip.  *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
flipV (Vertical Flip) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a vertical flip. When true, this attribute defines that the group is flipped vertically about the center of its bounding box. [*Example*: The following illustrates the effect of a vertical flip.  *end example*] The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
rot (Rotation) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies the rotation of the Graphic Frame. The units for which this attribute is specified in reside within the simple type definition referenced below. The possible values for this attribute are defined by the ST_Angle simple type (§20.1.10.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Transform2D) is located in §A.4.1. *end note*]

19.3.2 Embedded Objects

Within the slides portion of PresentationML, there are the embedded elements. These are objects that can be embedded within a slide. As we defined a slide to be a container it can be seen that it does not just contain shapes, pictures and text but embedded objects as well that are not necessarily native to the PresentationML platform.

19.3.2.1 control (Embedded Control)

This element specifies the existence of an embedded control in the slide.

Attributes	Description
id (Relationship ID) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship id that is used to identify this Embedded object from within a slide. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
imgH (Image Height)	Specifies the height of the embedded control. The possible values for this attribute are defined by the ST_PositiveCoordinate32 simple type (§20.1.10.42).
imgW (Image Width)	Specifies the width of the embedded control. The possible values for this attribute are defined by the ST_PositiveCoordinate32 simple type (§20.1.10.42).
name (Embedded Object Name)	Specifies the identifying name class used by scripting languages. This name is also used to construct the clipboard name. The possible values for this attribute are defined by the W3C XML Schema string datatype.
showAsIcon (Show Embedded Object As Icon)	Specifies whether the Embedded object shows as an icon or using its native representation. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_Control) is located in §A.3. *end note*]

19.3.2.2 embed (Embedded Object or Control)

This element specifies an Embedded object or Control that is embedded within the presentation.

Attributes	Description
followColorScheme (Color Scheme Properties for Embedded object)	Specifies the Color Scheme Properties for the corresponding Embedded object being specified. The possible values for this attribute are defined by the ST_OleObjectFollowColorScheme simple type (§19.7.6).

[*Note*: The W3C XML Schema definition of this element's content model (CT_OleObjectEmbed) is located in §A.3. *end note*]

19.3.2.3 link (Linked Object or Control)

This element specifies a link to an external Embedded object or Control.

Attributes	Description
updateAutomatic (Update Linked Embedded Objects Automatically)	This attribute determines if linked embedded objects are automatically updated when the presentation is opened or printed. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_OleObjectLink) is located in §A.3. *end note*]

19.3.2.4 oleObj (Global Element for Embedded objects and Controls)

This element specifies a global element to be used for an Embedded object and Control.

When the oleObject element contains a pic child element, the identifier specified by the pic/nvPicPr/cNvPr@id attribute shall be ignored and the identifier specified by the graphicFrame/nvGraphicFramePr/cNvPr@id attribute shall be used when deciding which identifier to use for the OLE object.

Attributes	Description
id (Relationship ID) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the relationship id that is used to identify this Embedded object from within a slide. The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
imgH (Image Height)	Specifies the height of the embedded control. The possible values for this attribute are defined by the ST_PositiveCoordinate32 simple

Attributes	Description
	type (§20.1.10.42).
imgW (Image Width)	Specifies the width of the embedded control. The possible values for this attribute are defined by the ST_PositiveCoordinate32 simple type (§20.1.10.42).
name (Embedded Object Name)	Specifies the identifying name class used by scripting languages. This name is also used to construct the clipboard name. The possible values for this attribute are defined by the W3C XML Schema string datatype.
progId (Embedded Object ProgID)	Specifies the progid for an Embedded object. The possible values for this attribute are defined by the W3C XML Schema string datatype.
showAsIcon (Show Embedded Object As Icon)	Specifies whether the Embedded object shows as an icon or using its native representation. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_OleObject) is located in §A.3. *end note*]

19.3.3 Programmable Tags

Within the slides portion of PresentationML there are the tag elements. These are extensibility names and values that assist in the storage of legacy variables from older file formats.

19.3.3.1 tag (Programmable Extensibility Tag)

This element specifies a programmable extensibility tag to be used for storage of legacy variables.

Attributes	Description
name (Name)	Specifies the name associated with this specific programmable tag. The possible values for this attribute are defined by the W3C XML Schema string datatype.
val (Value)	Specifies the value associated with this specific programmable tag. The possible values for this attribute are defined by the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_StringTag) is located in §A.3. *end note*]

19.3.3.2 tagLst (Programmable Tab List)

This element specifies the list of programmable extensibility tags that are used to store variables from legacy file formats.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TagList) is located in §A.3. *end note*]

19.4 Comments

A comment is a text note attached to a slide, with the primary purpose of allowing readers of a presentation to provide feedback to the presentation author. Each comment contains an unformatted text string and information about its author, and is attached to a particular location on a slide. Comments can be visible while editing the presentation, but do not appear when a slide show is given. The displaying application decides when to display comments and determines their visual appearance.

19.4.1 cm (Comment)

This element specifies a single comment attached to a slide. It contains the text of the comment, its position on the slide, and attributes referring to its author and date.

[*Example:*

```
<p:cm authorId="0" dt="2006-08-28T17:26:44.129" idx="1">
  <p:pos x="10" y="10"/>
  <p:text>Add diagram to clarify.</p:text>
</p:cm>
```

end example]

Attributes	Description
authorId (Comment Author ID)	This attribute specifies the author of the comment. It refers to the ID of an author in the comment author list for the document. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
dt (Comment Date/Time)	This attribute specifies the date and time this comment was last modified. The possible values for this attribute are defined by the W3C XML Schema dateTime datatype.
idx (Comment Index)	This attribute specifies an identifier for this comment that is unique within a list of all comments by this author in this document. An author's first comment in a document has index 1.

Attributes	Description
	[*Note:* Because the index is unique only for the comment author, a document can contain multiple comments with the same index created by different authors. *end note*] The possible values for this attribute are defined by the ST_Index simple type (§19.7.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Comment) is located in §A.3. *end note*]

19.4.2 cmAuthor (Comment Author)

This element specifies a single author with comments in the document. It contains a unique author ID, the author's name and initials, the index of the author's last comment, and the index of a color associated with the author.

[*Example:*

```
<p:cmAuthor id="0" name="Julie Lee" initials="JL" lastIdx="1" clrIdx="0"/>
```

end example]

Attributes	Description
clrIdx (Comment Author Color Index)	This attribute specifies an index into the generating application's comments color table to allow for visual (color) differentiation of different author's comments. This color is used for all comments by this author. If more authors exist than there are entries in the color table, the color index wraps around to the beginning of the table. [*Note:* It is left entirely up to the generating application to determine the amount of colors used in the comments color table and in what order these are used when rendering comments on a slide surface. *end note*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
id (Comment Author ID)	This attribute specifies a unique (within the document) zero-based identifier that refers to a single comment author. [*Note:* The method of generating an author id is determined by the application and need not be sequential, provided each id is unique within the list of comment authors for the document. *end note*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
initials (Comment Author Initials)	This attribute specifies a string that represents the initials of this particular author. The value is not necessarily unique. It is intended for use by the application as an abbreviated version of the comment author's name.

Attributes	Description
	The possible values for this attribute are defined by the ST_Name simple type (§19.7.5).
lastIdx (Index of Comment Author's last comment)	Index of the last comment added to this document by this author. New comments by this author are counted starting with the value one greater than this index. [*Note:* The index of a deleted comment is not reused; therefore, this value is not an accurate count of the total number of comments by the author. *end note*] The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
name (Comment Author Name)	This attribute specifies the full name of this particular author. As a string, it has no security or authentication data. This value is not guaranteed to be unique across all document authors. The possible values for this attribute are defined by the ST_Name simple type (§19.7.5).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CommentAuthor) is located in §A.3. *end note*]

19.4.3 cmAuthorLst (List of Comment Authors)

This element specifies a list of authors with comments in the current document. Each comment in a document shall refer to an author in this list. To determine if a new author is in this list, the author's name and initials shall both match; otherwise, the new author is considered unique and a separate cmAuthor element is added.

[*Example:* A document contains comments left by two authors.

```
<p:cmAuthorLst>
    <p:cmAuthor id="0" name="Julie Lee" initials="JL" lastIdx="1" clrIdx="0"/>
    <p:cmAuthor id="1" name="Fred Jones" initials="FJ" lastIdx="2" clrIdx="1"/>
</p:cmAuthorLst>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_CommentAuthorList) is located in §A.3. *end note*]

19.4.4 cmLst (Comment List)

This element specifies a list of comments for a particular slide.

[*Example:* A slide contains two comments, each left by a different author. This example demonstrates that two comments can have the same index if they are created by different authors.

```
<p:cmLst>
```



```
    <p:cm authorId="0" dt="2006-08-28T17:26:44.129" idx="1">
        <p:pos x="10" y="10"/>
        <p:text>Add diagram to clarify.</p:text>
    </p:cm>
    <p:cm authorId="1" dt="2006-08-28T17:44:19.679" idx="1">
        <p:pos x="1426" y="660"/>
        <p:text>Clean up this text.</p:text>
    </p:cm>
</p:cmLst>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_CommentList) is located in §A.3. *end note*]

19.4.5 pos (Comment Position)

This element specifies the positioning information for the placement of a comment on a slide surface. In LTR versions of the generating application, this position information should refer to the upper left point of the comment shape. In RTL versions of the generating application, this position information should refer to the upper right point of the comment shape.

[*Note:* The anchoring point on the slide surface is unaffected by a right-to-left or left-to-right layout change. That is the anchoring point remains the same for all language versions. *end note*]

[*Note:* Because there is no specified size or formatting for comments, this UI widget used to display a comment can be any size and thus the lower right point of the comment shape is determined by how the viewing application chooses to display comments. *end note*]

[*Example:*

```
<p:pos x="1426" y="660"/>
```

end example]

Attributes	Description
x (X-Axis Coordinate) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<… x="0" y="100" />` The x attribute defines an x-coordinate of 0. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type

Attributes	Description
	(§20.1.10.16).
y (Y-Axis Coordinate) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies a coordinate on the x-axis. The origin point for this coordinate shall be specified by the parent XML element. [*Example*: Consider the following point on a basic wrapping polygon for a DrawingML object: `<… x="0" y="100" />` The y attribute defines a y-coordinate of `100`. *end example*] The possible values for this attribute are defined by the ST_Coordinate simple type (§20.1.10.16).

[*Note*: The W3C XML Schema definition of this element's content model (CT_Point2D) is located in §A.4.1. *end note*]

19.4.6 text (Comment's Text Content)

This element specifies the content of a comment. This is the text with which the author has annotated the slide.

[*Example:*

```
<p:text>Add diagram to clarify.</p:text>
```

end example]

The possible values for this element are defined by the W3C XML Schema string datatype.

19.5 Animation

The Animation section of the PresentationML framework stores the movement and related information of objects.

This schema is loosely based on the syntax and concepts from the Synchronized Multimedia Integration Language (SMIL), a W3C Recommendation for describing multimedia presentations using XML.

The schema describes all the animations effects that reside on a slide and also the animation that occurs when going from slide to slide (slide transition).

Animations on a slide are inherently time-based and consist of an animation effects on an object or text. Slide transitions however do not follow this concept and always appear before any animation on a slide.

All elements described in this schema are contained within the slide XML file. More specifically they are in the <transition> and the <timing> element as shown below:


```
<p:sld>
  <p:cSld> … </p:cSld>
  <p:clrMapOvr> … </p:clrMapOvr>
  <p:transition> … </p:transition>
  <p:timing> … </p:timing>
</p:sld>
```

19.5.1 anim (Animate)

This element is a generic animation element that requires little or no semantic understanding of the attribute being animated. It can animate text within a shape or even the shape itself.

[*Example*: Consider trying to emphasize text within a shape by changing the size of its font by 150%. The <anim> element should be used as follows:

```
<p:anim to="1.5" calcmode="lin" valueType="num">
  <p:cBhvr override="childStyle">
    <p:cTn id="1" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="1">
        <p:txEl>
          <p:charRg st="1" end="4"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>style.fontSize</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
</p:anim>
```

end example]

Attributes	Description
by (By)	This attribute specifies a relative offset value for the animation with respect to its position before the start of the animation. The possible values for this attribute are defined by the W3C XML Schema string datatype.
calcmode (Calculation Mode)	This attribute specifies the interpolation mode for the animation. The possible values for this attribute are defined by the ST_TLAnimateBehaviorCalcMode simple type (§19.7.20).
from (From)	This attribute specifies the starting value of the animation.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema string datatype.
to (To)	This attribute specifies the ending value for the animation as a percentage. The possible values for this attribute are defined by the W3C XML Schema string datatype.
valueType (Value Type)	This attribute specifies the type of property value. The possible values for this attribute are defined by the ST_TLAnimateBehaviorValueType simple type (§19.7.21).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimateBehavior) is located in §A.3. *end note*]

19.5.2 animClr (Animate Color Behavior)

This animation element is responsible for animating the color of an object.

[*Example*: Consider trying to emphasize a shape by changing its fill color to scheme color accent2. The <animClr> element should be used as follows:

```
<p:animClr clrSpc="rgb">
  <p:cBhvr>
    <p:cTn id="1" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="1"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>fillcolor</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:to>
    <a:schemeClr val="accent2"/>
  </p:to>
</p:animClr>
```

end example]

Attributes	Description
clrSpc (Color Space)	This attribute specifies the color space in which to interpolate the animation. Values for example can be HSL & RGB.

Attributes	Description
	The values for from/to/by/etc. can still be specified in any supported color format without affecting the color space within which the animation happens. The RGB color space is best used for doing animations between two different colors since it doesn't require going through any other hues between the two colors specified. The HSL space is useful for animating through a rainbow of colors or for modifying just the saturation by 30% for example. The possible values for this attribute are defined by the ST_TLAnimateColorSpace simple type (§19.7.23).
dir (Direction)	This attribute specifies which direction to cycle the hue around the color wheel. Values are clockwise or counter clockwise. Default is clockwise. The possible values for this attribute are defined by the ST_TLAnimateColorDirection simple type (§19.7.22).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimateColorBehavior) is located in §A.3. *end note*]

19.5.3 animEffect (Animate Effect)

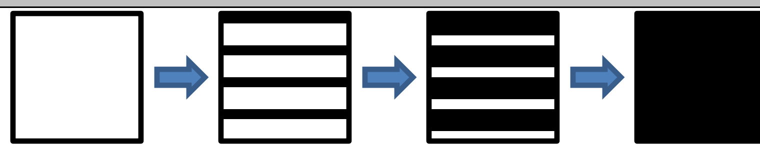
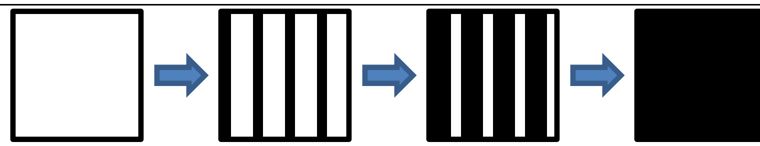
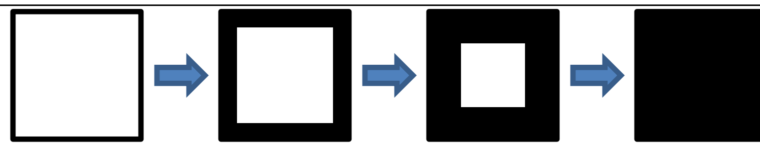
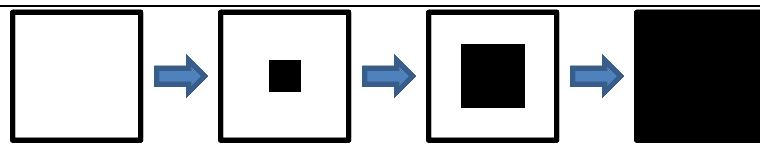
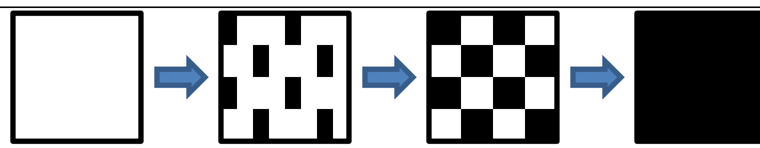
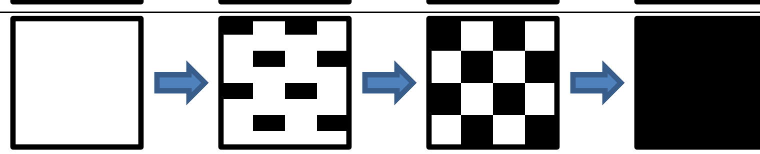
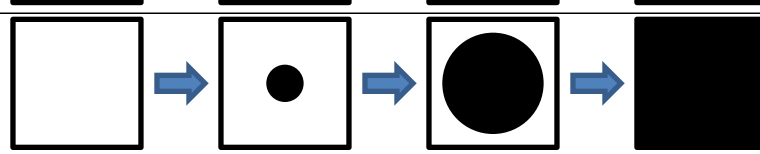
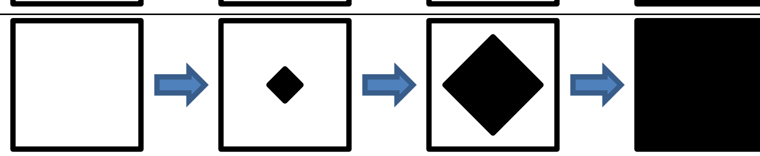
This animation behavior provides the ability to do image transform/filter effects on elements. Some visual effects are dynamic in nature and have a progress that animates from 0 to 1 over a period of time to do visual transitions between hidden and visible states. Other filters are static and apply a effects like a blur or drop-shadow which aren't inherently time-based.

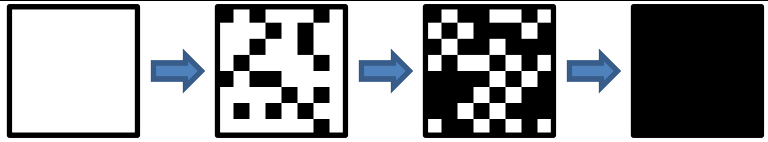
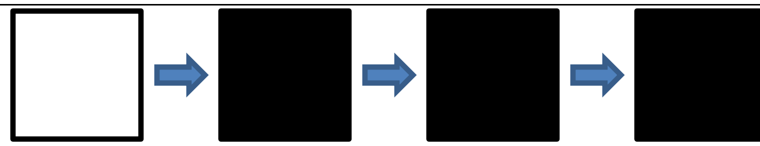
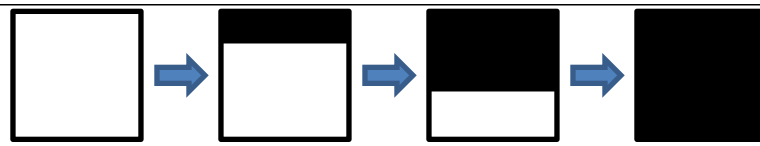
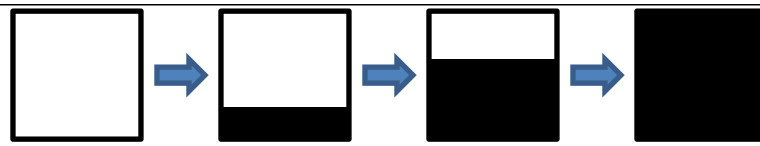
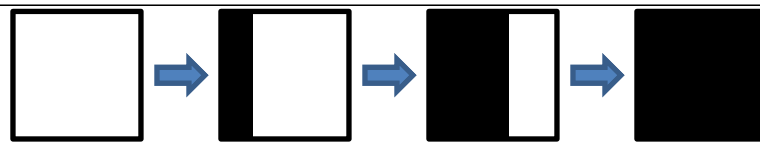
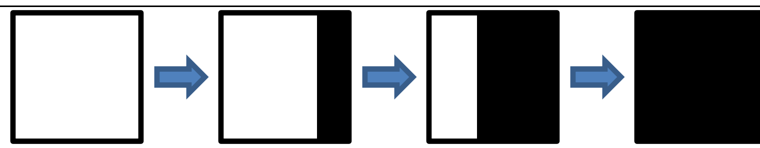
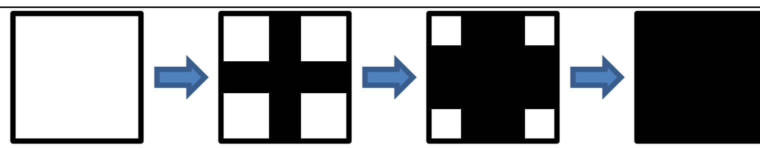
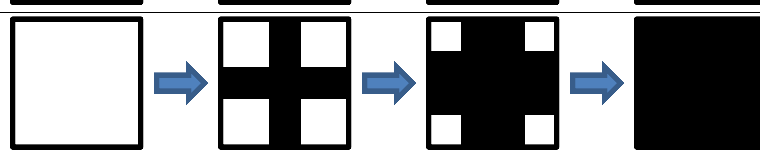
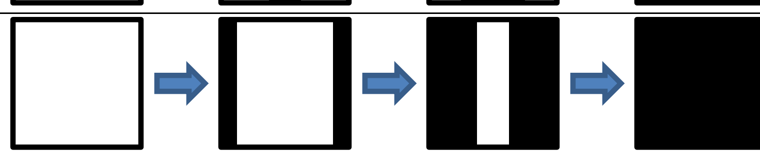
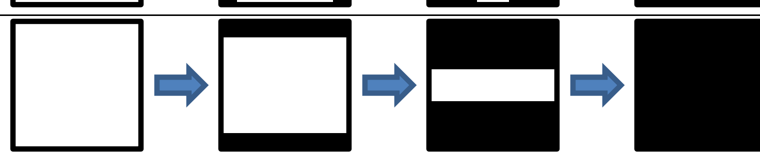
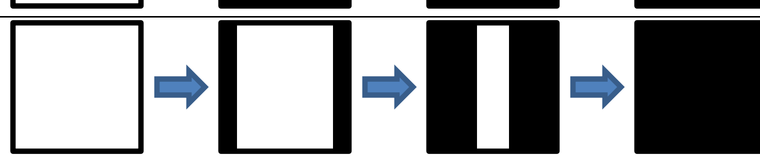
[*Example*: Consider trying to emphasize a shape by creating an entrance animation using a "blinds" motion.

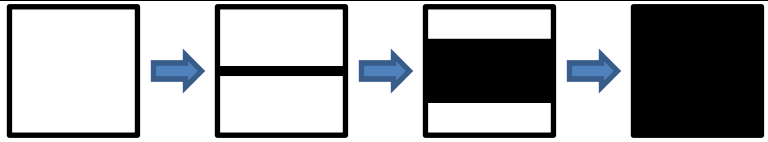
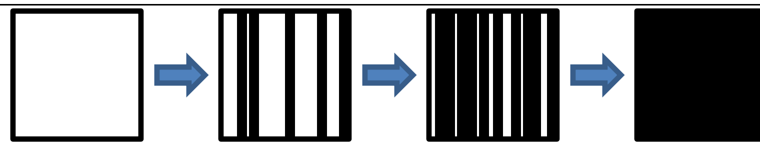
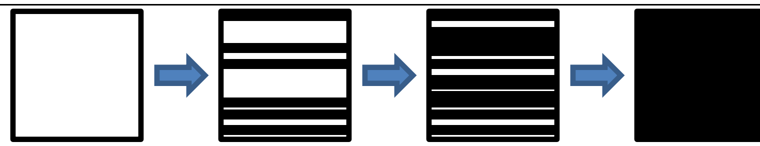
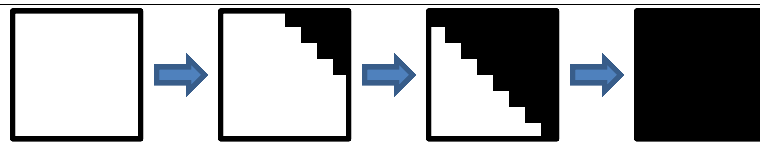
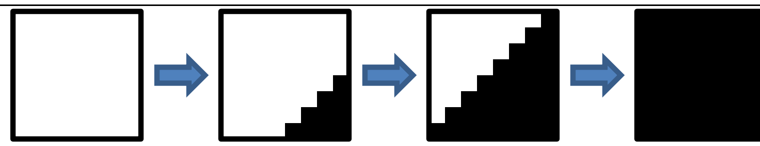
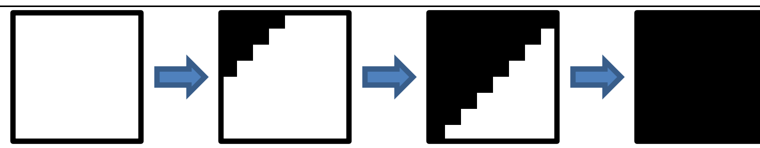
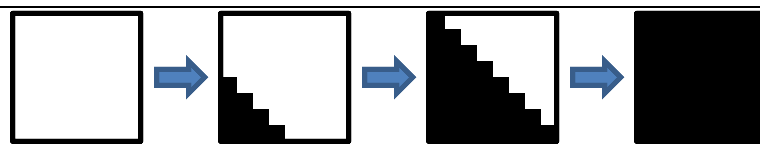
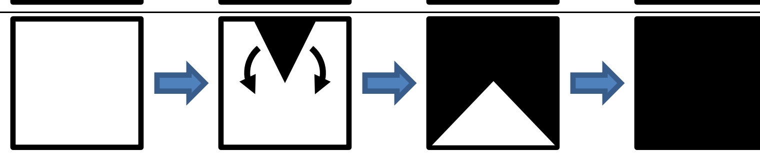
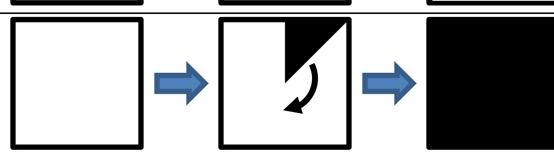
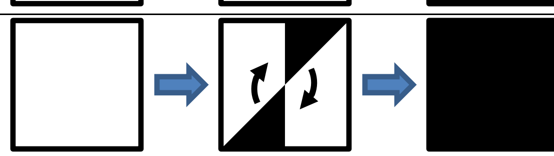
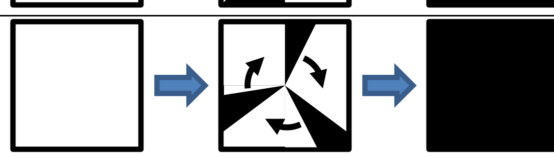
```
<p:animEffect transition="in" filter="blinds(horizontal)">
  <p:cBhvr>
    <p:cTn id="7" dur="500"/>
    <p:tgtEl>
      <p:spTgt spid="4"/>
    </p:tgtEl>
  </p:cBhvr>
</p:animEffect>
```

end example]

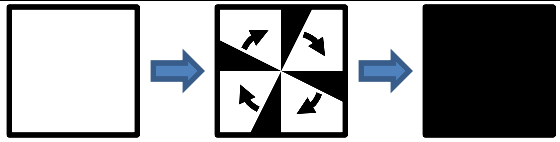
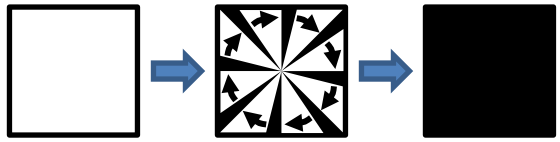
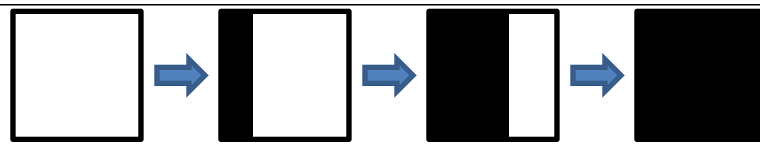
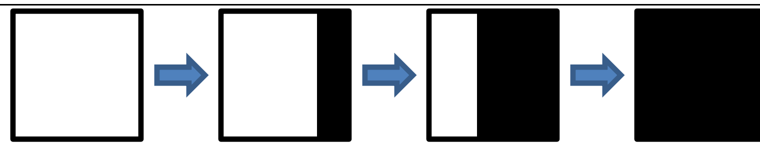
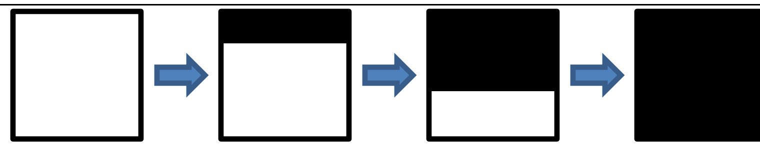
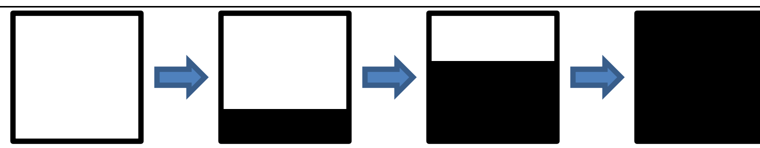
Attributes	Description
filter (Filter)	

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
This attribute specifies the animation types and subtypes to be used for the effect. Multiple animations are allowed to be listed so that in the event that a superseding animation (leftmost) cannot be rendered, a fallback animation is available. That is, the rendering application parses the list from left to right until a supported animation is found.
The syntax used for the filter attribute value is as follows: "type(subtype);type(subtype)". Subtype can be a string value such as "fromLeft" or a numerical value depending on the type specified.
Reserved Animation Types(subtypes):
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>blinds(horizontal)</td><td></td></tr>
<tr><td>blinds(vertical)</td><td></td></tr>
<tr><td>box(in)</td><td></td></tr>
<tr><td>box(out)</td><td></td></tr>
<tr><td>checkerboard(across)</td><td></td></tr>
<tr><td>checkerboard(down)</td><td></td></tr>
<tr><td>circle</td><td></td></tr>
<tr><td>diamond</td><td></td></tr>
</table>
</td></tr>
</table>

Attributes	Description	
	dissolve	
	fade	
	slide(fromTop)	
	slide(fromBottom)	
	slide(fromLeft)	
	slide(fromRight)	
	plus(in)	
	plus(out)	
	barn(inVertical)	
	barn(inHorizontal)	
	barn(outVertical)	

Attributes	Description	
	barn(outHorizontal)	
	randomBars(horizontal)	
	randomBars(vertical)	
	strips(downLeft)	
	strips(upLeft)	
	strips(downRight)	
	strips(upRight)	
	wedge	
	wheel(1)	
	wheel(2)	
	wheel(3)	

Attributes	Description

wheel(4)	
wheel(8)	
wipe(right)	
wipe(left)	
wipe(down)	
wipe(up)	

[*Note:* The renderings shown above are for example purposes only. Exact rendering of any animation is determined by the rendering application. As such, the same animation can have many variations, depending on the implementation. More detail for each rendering above can be found in transition (§19.3.1.50). *end note*]

[*Example*: Consider the following animation effect:

```
<p:animEffect transition="in"
filter="blinds(horizontal);blinds(vertical)">
  <p:cBhvr>
    <p:cTn id="7" dur="500"/>
      <p:tgtEl>
        <p:spTgtspid="5"/>
      </p:tgtEl>
  </p:cBhvr>
</p:animEffect>
```

There are two animation filters shown in this example. The first is the blinds (horizontal), which the rendering application is to use as the primary animation effect. If, however, the rendering application does not support this animation, the blinds (vertical) animation is used. In this example there are only two animation filters listed, a primary and a

Attributes	Description
	fallback, but it is possible to list multiple fallback filters using the syntax defined above. *end example*] The possible values for this attribute are defined by the W3C XML Schema string datatype.
prLst (Property List)	This attribute specifies a list of properties that coincide with the effect specified. Although there are many animation types allowed, this attribute allows the setting of specific property settings in order to describe an even wider variety of animation types. The syntax used for the prLst attribute value is as follows: “name:value;name:value”. When multiple animation types are listed in the filter attribute, the rendering application attempts to apply each property value even though some might not apply to it. Reserved Names(values): • opacity (float values of 0.0 - 1.0) [*Example*: Consider the following animation effect: `<p:animEffect filter="image" prLst="opacity: 0.5">` `  <p:cBhvr rctx="IE">` `    <p:cTn id="7" dur="indefinite"/>` `    <p:tgtEl>` `      <p:spTgtspid="3"/>` `    </p:tgtEl>` `  </p:cBhvr>` `</p:animEffect>` The animation filter specified is an image filter type that has a specific property called opacity set to a value of 0.5. *end example*] The possible values for this attribute are defined by the W3C XML Schema string datatype.
transition (Transition)	This attribute specifies whether to transition the element in or out or treat it as a static filter. The values are "none", "in" and "out", and the default value is "in". When a value of "in" is specified, the element is not visible at the start of the animation and is completely visible be the end of the duration. When "out" is specified, the element is visible at the start and not visible at the end of the effect. This visibility is in addition to the effect of setting CSS visibility or display attributes. The possible values for this attribute are defined by the ST_TLAnimateEffectTransition simple type (§19.7.24).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimateEffectBehavior) is located in §A.3. *end note*]

19.5.4 animMotion (Animate Motion)

Animate motion provides an abstracted way to move positioned elements. It provides the ability to specify from/to/by motion as well as to use more detailed path descriptions for motion over polylines or bezier curves.

[*Example*: Consider animating a shape from its original position to the right.. The <animMotion> element should be used as follows:

```
<p:animMotion origin="layout" path="M 0 0 L 0.25 0 E" pathEditMode="relative">
  <p:cBhvr>
    <p:cTn id="1" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="1"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>ppt_x</p:attrName>
      <p:attrName>ppt_y</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
</p:animMotion>
```

end example]

Attributes	Description
origin (Origin)	Specifies what the origin of the motion path is relative to such as the layout of the slide, or the parent. The possible values for this attribute are defined by the ST_TLAnimateMotionBehaviorOrigin simple type (§19.7.25).
path (Path)	Specifies the path primitive followed by coordinates for the animation motion. The allowed values that are understood within a path are as follows: M = move to, L = line to, C = curve to, Z=close loop, E=end UPPERCASE = absolute coords, lowercase = relative coords Thus total allowed set = {M,L,C,Z,E,m,l,c,z,e) [*Example*: The following string is a sample path. path: "M 0 0 L 1 1 c 1 2 3 4 4 4 Z" *end example*] The possible values for this attribute are defined by the W3C XML Schema string

Attributes	Description
	datatype.
pathEditMode (Path Edit Mode)	This attribute specifies how the motion path moves when the target element is moved. The possible values for this attribute are defined by the ST_TLAnimateMotionPathEditMode simple type (§19.7.26).
ptsTypes (Points Types)	This attribute describes the point type of the points in the path attribute. The allowed values that are understood for the ptsTypes attribute are as follows: A = Auto, F = Corner, T = Straight, S = Smooth UPPERCASE = Straight Line follows point, lowercase = curve follows point. Thus, the total allowed set = {A,F,T,S,a,f,t,s} The possible values for this attribute are defined by the W3C XML Schema string datatype.
rAng (Relative Angle)	The attribute describes the relative angle of the motion path. The possible values for this attribute are defined by the ST_Angle simple type (§20.1.10.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimateMotionBehavior) is located in §A.3. *end note*]

19.5.5 animRot (Animate Rotation)

This animation element is responsible for animating the rotation of an object. Rotation values set in the "by" , "to, and "from" attributes are specified in degrees measured to a 60,000th, i.e 1 degree is 60,000. Rotation values can be larger than 360°.

The sign of the rotation angle specifies the direction for rotation. A negative rotation specifies that the rotation should appear in the host to go counter-clockwise".

[*Example*: Consider trying to emphasize a shape by rotating it 360 degrees clockwise. The <animRot> element should be used as follows:

```
<p:animRot by="21600000">
  <p:cBhvr>
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="5"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>r</p:attrName>
    </p:attrNameLst>
```



```
    </p:cBhvr>
  </p:animRot>
```

end example]

Attributes	Description
by (By)	This attribute describes the relative offset value for the animation. The possible values for this attribute are defined by the ST_Angle simple type (§20.1.10.3).
from (From)	This attribute describes the starting value for the animation. The possible values for this attribute are defined by the ST_Angle simple type (§20.1.10.3).
to (To)	This attribute describes the ending value for the animation. The possible values for this attribute are defined by the ST_Angle simple type (§20.1.10.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimateRotationBehavior) is located in §A.3. *end note*]

19.5.6 animScale (Animate Scale)

This animation element is responsible for animating the scale of an object. When animating the scale, the element shall scale around the reference point of the element and the positioning system used should be consistent with the one used for motion paths. When animating the width and height of an element, all of the width/height animation values are calculated first then the scale animations are applied on top of that. So for example, an animation from 0 to 100 of the width with a concurrent scale from 100% to 200% would result in the element appearing to scale from 0 to 200.

[*Example*: Consider trying to emphasize a shape by scaling it larger by 150%. The <animScale> element should be used as follows:

```
  <p:childTnLst>
    <p:animScale>
      <p:cBhvr>
        <p:cTn id="6" dur="2000" fill="hold"/>
        <p:tgtEl>
          <p:spTgt spid="5"/>
        </p:tgtEl>
      </p:cBhvr>
      <p:by x="150000" y="150000"/>
    </p:animScale>
```



```
</p:childTnLst>
```

end example]

Attributes	Description
zoomContents (Zoom Content)	This attribute specifies whether to zoom the contents of an object when doing a scaling animation. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimateScaleBehavior) is located in §A.3. *end note*]

19.5.7 attrName (Attribute Name)

This element is used to contain an attribute value for an Attribute Name List. This value defines the specific attribute that an animation should be applied to, such as fill, style, and shadow, etc. A specific property is defined by using a "property.sub-property" format which is often extended to multiple sub properties as seen in the allowed values below.

Allowed property values:

```
style.opacity, style.rotation, style.visibility, style.color, style.fontSize,
style.fontWeight, style.fontStyle, style.fontFamily, style.textEffectEmboss,
style.textShadow, style.textTransform, style.textDecorationUnderline,
style.textEffectOutline, style.textDecorationLineThrough, style.sRotation,
imageData.cropTop, imageData.cropBottom, imageData.cropLeft,
imageData.cropRight, imageData.cropRight, imageData.gain, imageData.blacklevel,
imageData.gamma, imageData.grayscale, imageData.chromakey, fill.on, fill.type,
fill.color, fill.opacity, fill.color2, fill.method, fill.opacity2, fill.angle,
fill.focus, fill.focusposition.x, fill.focusposition.y, fill.focussize.x,
fill.focussize.y, stroke.on, stroke.color, stroke.weight, stroke.opacity,
stroke.linestyle, stroke.dashstyle, stroke.filltype, stroke.src, stroke.color2,
stroke.imagesize.x, stroke.imagesize.y, stroke.startArrow, stroke.endArrow,
stroke.startArrowWidth, stroke.startArrowLength, stroke.endArrowWidth,
stroke.endArrowLength, shadow.on, shadow.type, shadow.color, shadow.color2,
shadow.opacity, shadow.offset.x, shadow.offset.y, shadow.offset2.x,
shadow.offset2.y, shadow.origin.x, shadow.origin.y, shadow.matrix.xtox,
shadow.matrix.ytox, shadow.matrix.xtox, shadow.matrix.ytoy,
shadow.matrix.perspectiveX, shadow.matrix.perspectiveY, skew.on, skew.offset.x,
skew.offset.y, skew.origin.x, skew.origin.y, skew.matrix.xtox, skew.matrix.ytox,
skew.matrix.xtox, skew.matrix.ytoy, skew.matrix.perspectiveX,
```

skew.matrix.perspectiveY, extrusion.on, extrusion.type, extrusion.render, extrusion.viewpointorigin.x, extrusion.viewpointorigin.y, extrusion.viewpoint.x, extrusion.viewpoint.y, extrusion.viewpoint.z, extrusion.plane, extrusion.skewangle, extrusion.skewamt, extrusion.backdepth, extrusion.foredepth, extrusion.orientation.x, extrusion.orientation.y, extrusion.orientation.z, extrusion.orientationangle, extrusion.color, extrusion.rotationangle.x, extrusion.rotationangle.y, extrusion.lockrotationcenter, extrusion.autorotationcenter, extrusion.rotationcenter.x, extrusion.rotationcenter.y, extrusion.rotationcenter.z, and extrusion.colormode.

[*Example*: Consider trying to emphasize the txt font size within the body of a shape. The attribute would be 'style.fontSize' and this can be done by doing the following:

```
<p:anim to="1.5" calcmode="lin" valueType="num">
  <p:cBhvr override="childStyle">
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="3">
        <p:txEl>
          <p:charRg st="4294967295" end="4294967295"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>style.fontSize</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
</p:anim>
```

end example]

The possible values for this element are defined by the W3C XML Schema string datatype.

19.5.8 attrNameLst (Attribute Name List)

This element is used to describe a list of attributes in which to apply an animation to.

[*Example*: Consider trying to emphasize the txt font size within the body of a shape. The attribute would be 'style.fontSize' and this can be done by doing the following:

```
<p:anim to="1.5" calcmode="lin" valueType="num">
  <p:cBhvr override="childStyle">
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="3">
```

```
        <p:txEl>
          <p:charRg st="4294967295" end="4294967295"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>style.fontSize</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
</p:anim>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLBehaviorAttributeNameList) is located in §A.3. *end note*]

19.5.9 audio (Audio)

This element is used to include audio during an animation. This element specifies that this node within the animation tree triggers the playback of an audio file; the actual audio file used is specified by the sndTgt element (§19.5.70).

[*Example*: Consider adding applause sound to an animation sequence. The audio element is used as follows:

```
<p:cTn …>
  <p:stCondLst>…</p:stCondLst>
  <p:childTnLst>…</p:childTnLst>
  <p:subTnLst>
    <p:audio>
      <p:cMediaNode vol="50%">…
        <p:tgtEl>
          <p:sndTgt r:embed="rId2" />
        </p:tgtEl>
      </p:cMediaNode>
    </p:audio>
  </p:subTnLst>
</p:cTn>
```

The audio element specifies the location of the audio playback within the animation; its child sndTgt element specifies that the audio to be played is the target of the relationship with ID rId2.

end example]

Attributes	Description
isNarration (Is Narration)	This attribute indicates whether the audio is a narration for the slide.

Attributes	Description
	The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLMediaNodeAudio) is located in §A.3. *end note*]

19.5.10 bg (Background)

This element is used to specify animating the background of an object.

[*Example*: Consider adding animation to the background of Shape Id 3. The <bg> tag can be used as follows:

```
<p:tgtEl>
  <p:spTgt spid="3">
    <p:bg/>
  </p:spTgt>
</p:tgtEl>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.3. *end note*]

19.5.11 bldAsOne (Build As One)

This element specifies in the build list to build the entire graphical object as one entity.

[*Example*: Consider having a graph appear as on entity as opposed to by category. The <bldAsOne> element should be used as follows:

```
<p:bldLst>
  <p:bldGraphic spid="4" grpId="0">
    <p:bldAsOne/>
  </p:bldGraphic>
</p:bldLst>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.3. *end note*]

19.5.12 bldDgm (Build Diagram)

This element specifies how to build the animation for a diagram.

[*Example*: Consider the following example where a chart is specified to be animated by category rather than as one entity. Thus, the bldChart element should be used as follows:

```
<p:bdldLst>
```



```
    <p:bldGraphic spid="4" grpId="0">
      <p:bldSub>
        <a:bldChart bld="category"/>
      </p:bldSub>
    </p:bldGraphic>
  </p:bldLst>
```

end example]

Attributes	Description
bld (Diagram Build Types)	This attribute describes how the diagram is built. The animation animates the sub-elements in the container in the particular order defined by this attribute. The possible values for this attribute are defined by the ST_TLDiagramBuildType simple type (§19.7.33).
grpId (Group ID)	This attribute ties effects persisted in the animation to the build information. The attribute is used by the editor when changes to the build information are made. GroupIDs are unique for a given shape. They are not guaranteed to be unique IDs across all shapes on a slide. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
spid (Shape ID)	This attribute specifies the shape to which the build applies. The possible values for this attribute are defined by the ST_DrawingElementId simple type (§20.1.10.21).
uiExpand (Expand UI)	This attribute describes the view option indicating if the build should be displayed expanded. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLBuildDiagram) is located in §A.3. *end note*]

19.5.13 bldGraphic (Build Graphics)

This element specifies how to build a graphical element.

[*Example*: Consider having a chart graphical element appear as a whole as opposed to by a category. The <bldGraphic> element should be used as follows:

```
<p:bldLdst>
  <p:bldGraphic spid="3" grpId="0">
    <p:bldSub>
```



```
        <a:bldChart bld="category"/>
      </p:bldSub>
    </p:bldGraphic>
  </p:bldLst>
```

end example]

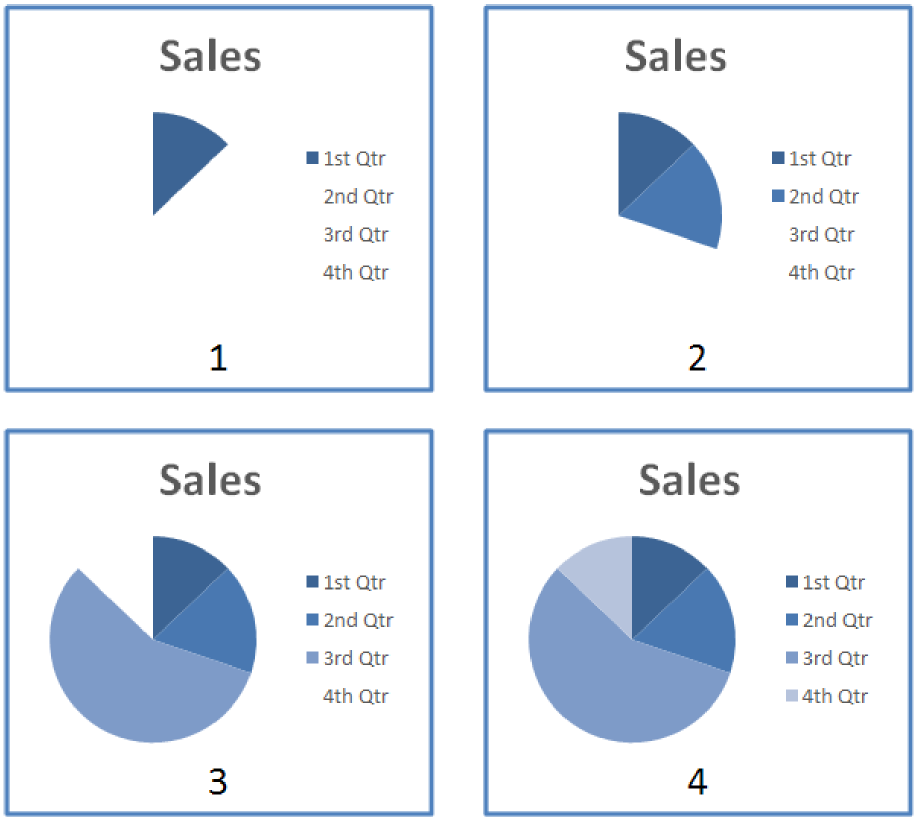
Attributes	Description
grpId (Group ID)	This attribute ties effects persisted in the animation to the build information. The attribute is used by the editor when changes to the build information are made. GroupIDs are unique for a given shape. They are not guaranteed to be unique IDs across all shapes on a slide. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
spid (Shape ID)	This attribute specifies the shape to which the build applies. The possible values for this attribute are defined by the ST_DrawingElementId simple type (§20.1.10.21).
uiExpand (Expand UI)	This attribute describes the view option indicating if the build should be displayed expanded. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLGraphicalObjectBuild) is located in §A.3. *end note*]

19.5.14 bldLst (Build List)

This element specifies the list of graphic elements to build. This refers to how the different sub-shapes or sub-components of a object are displayed. The different objects that can have build properties are text, diagrams, and charts.

[*Example*: Consider animating a pie chart but based on category as shown below:



The <bldList> element should be used as follows:

```
<p:bldLst>
  <p:bldGraphic spid="1" grpId="0">
    <p:bldSub>
      <a:bldChart bld="category"/>
    </p:bldSub>
  </p:bldGraphic>
</p:bldLst>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_BuildList) is located in §A.3. *end note*]

19.5.15 bldOleChart (Build Embedded Chart)

This element describes animation an a embedded Chart.

[*Example*: Consider displaying animation on a embedded graphical chart. The <bldOleChart>element should be use as follows:

```
<p:bldLst>
```



```
    <p:bldOleChart spid="1025" grpId="0"/>
  </p:bldLst>
```

end example]

Attributes	Description
animBg (Animate Background)	This attribute describes whether to animate the background of the shape. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
bld (Build)	This attribute describes how the diagram is built. The animation animates the sub-elements in the container in the particular order defined by this attribute. The possible values for this attribute are defined by the ST_TLOleChartBuildType simple type (§19.7.35).
grpId (Group ID)	This attribute ties effects persisted in the animation to the build information. The attribute is used by the editor when changes to the build information are made. GroupIDs are unique for a given shape. They are not guaranteed to be unique IDs across all shapes on a slide. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
spid (Shape ID)	This attribute specifies the shape to which the build applies. The possible values for this attribute are defined by the ST_DrawingElementId simple type (§20.1.10.21).
uiExpand (Expand UI)	This attribute describes the view option indicating if the build should be displayed expanded. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLOleBuildChart) is located in §A.3. *end note*]

19.5.16 bldP (Build Paragraph)

This element specifies how to build paragraph level properties.

[*Example*: Consider having animation applied only to 1st level paragraphs. The <bldP> element should be used as follows:

```
  <p:bldLst>
    <p:bldP spid="3" grpId="0" build="p"/>
  </p:bldLst>
```


end example]

Attributes	Description
advAuto (Auto Advance Time)	This attribute specifies time after which to automatically advance the build to the next step. The possible values for this attribute are defined by the ST_TLTime simple type (§19.7.38).
animBg (Animate Background)	This attribute indicates whether to animate the background of the shape associated with the text. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoUpdateAnimBg (Auto Update Animation Background)	This attribute indicates whether to automatically update the "animateBg" setting to true when the shape associated with the text has a fill or line. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
bldLvl (Build Level)	This attribute describes the build level for the paragraph. It is only supported in paragraph type builds i.e the build attribute shall also be set to "byParagraph" for this attribute to apply. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
build (Build Types)	This attribute describe the build types. The possible values for this attribute are defined by the ST_TLParaBuildType simple type (§19.7.36).
grpId (Group ID)	This attribute ties effects persisted in the animation to the build information. The attribute is used by the editor when changes to the build information are made. GroupIDs are unique for a given shape. They are not guaranteed to be unique IDs across all shapes on a slide. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
rev (Reverse)	This attribute is only supported in paragraph type builds. This specifies the direction of the build relative to the order of the elements in the container. When this is set to "true", the animations for the paragraphs are persisted in reverse order to the order of the paragraphs themselves such that the last paragraph animates first. Default value is "false". The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
spid (Shape ID)	This attribute specifies the shape to which the build applies.

Attributes	Description
	The possible values for this attribute are defined by the ST_DrawingElementId simple type (§20.1.10.21).
uiExpand (Expand UI)	This attribute describes the view option indicating if the build should be displayed expanded. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLBuildParagraph) is located in §A.3. *end note*]

19.5.17 bldSub (Build Sub Elements)

This element specifies the animation properties of a graphical object's sub-elements.

[*Example*: Consider applying animation to a graphical element consisting of a diagram. The <bldSub> element should be used as follows:

```
<p:bldLst>
  <p:bldGraphic spid="5" grpId="0">
    <p:bldSub>
      <a:bldDgm bld="one"/>
    </p:bldSub>
  </p:bldGraphic>
</p:bldLst>
```

end example]

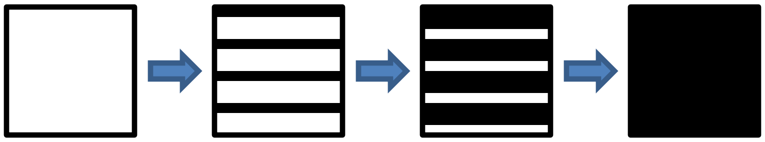
[*Note*: The W3C XML Schema definition of this element's content model (CT_AnimationGraphicalObjectBuildProperties) is located in §A.4.1. *end note*]

19.5.18 blinds (Blinds Slide Transition)

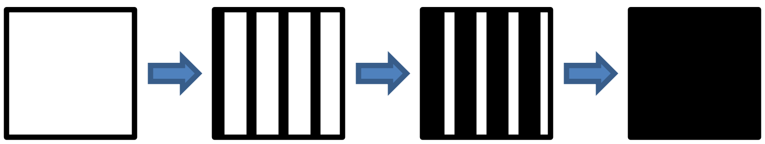
This element describes the blinds slide transition effect, which uses a set of horizontal or vertical bars and wipes them either left-to-right or top-to-bottom, respectively, until the new slide is fully shown. The rendering of this transition depends upon the attributes specified.

[*Example*: Consider the following cases in which the "blinds" slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
  <p:blinds dir="horz"/>
</p:transition>
```

```
<p:transition>
  <p:blinds dir="vert"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
dir (Transition Direction)	This attribute specifies a horizontal or vertical transition. The possible values for this attribute are defined by the ST_Direction simple type (§19.7.2).

[*Note*: The W3C XML Schema definition of this element's content model (CT_OrientationTransition) is located in §A.3. *end note*]

19.5.19 boolVal (Boolean Variant)

This element specifies a boolean value to be used for evaluation by a parent element. The exact meaning of the value contained within this element is not defined here but is dependent on the usage of this element in conjunction with one of the listed parent elements.

Attributes	Description
val (Value)	This attribute specifies the boolean value that this element contains and that should be used in evaluating this element. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimVariantBooleanVal) is located in §A.3. *end note*]

19.5.20 by (By)

This element describes the relative offset value for the color animation.

[*Example*: Consider a shape with a lightening emphasis animation applied to it. The <by> element should be used as follows:

```
<p:animClr clrSpc="hsl">
  <p:cBhvr>
    <p:cTn id="8" dur="500" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="4"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>stroke.color</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:by>
    <p:hsl h="0" s="0" l="0"/>
  </p:by>
</p:animClr>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLByAnimateColorTransform) is located in §A.3. *end note*]

19.5.21 by (By)

This element describes the relative offset value for the animation.

[*Example*: Consider a shape with an animation effect that scales the size of an object by 150%. The <by> element should be used as follows:

```
<p:animScale>
  <p:cBhvr>
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="4"/>
    </p:tgtEl>
  </p:cBhvr>
  <p:by x="150.000%" y="150.000%"/>
</p:animScale>
```

end example]

Attributes	Description
x (X coordinate)	This attribute describes the X coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§20.1.10.40).
y (Y coordinate)	This attribute describes the Y coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§20.1.10.40).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLPoint) is located in §A.3. *end note*]

19.5.22 cBhvr (Common Behavior)

This element describes the common behaviors of animations.

[*Example*: Consider trying to emphasize text within a shape by changing the size of its font. The <anim> element should be used as follows:

```
<p:anim to="1.5" calcmode="lin" valueType="num">
  <p:cBhvr override="childStyle">
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="3">
        <p:txEl>
          <p:charRg st="4294967295" end="4294967295"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>style.fontSize</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
</p:anim>
```

end example]

Attributes	Description
accumulate (Accumulate)	This attribute makes a repeating animation build with each iteration when set to "always." The possible values for this attribute are defined by the ST_TLBehaviorAccumulateType simple type (§19.7.27).

Attributes	Description
additive (Additive)	This attribute specifies how to apply the animation values to the original value for the property. The possible values for this attribute are defined by the ST_TLBehaviorAdditiveType simple type (§19.7.28).
by (By)	This attribute specifies a relative offset value for the animation.. The possible values for this attribute are defined by the W3C XML Schema string datatype.
from (From)	This attribute specifies the starting value of the animation. The possible values for this attribute are defined by the W3C XML Schema string datatype.
override (Override)	This attribute specifies how a behavior should override values of the attribute being animated on the target element. The "childStyle" clears the attributes on the children contained inside the target element. The possible values for this attribute are defined by the ST_TLBehaviorOverrideType simple type (§19.7.29).
rctx (Runtime Context)	This attribute describes the runtime context of the animation. The currently-understood values are “PPT” and “IE.” This is used to specify the behavior used when animating in the PPT slideshow vs. IE HTML runtime. An example can be seen with the transparency effect. In IE, the transparency is animated as a bitmap, where in PPT, the style.opacity property of a shape is used to animate the transparency. The possible values for this attribute are defined by the W3C XML Schema string datatype.
to (To)	This attribute specifies the ending value of the animation. The possible values for this attribute are defined by the W3C XML Schema string datatype.
xfrmType (Transform Type)	This attribute specifies the kind of transform to be used. The possible values for this attribute are defined by the ST_TLBehaviorTransformType simple type (§19.7.30).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_TLCommonBehaviorData) is located in §A.3. *end note*]

19.5.23 charRg (Character Range)

This element specifies animation on a character range defined by a start and end character position.

[*Example*: Consider animating the first word (characters 1 through 9) within a sentence. The <charRg> element should be used as follows:

```
<p:animMotion>
  <p:cBhvr>
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="3">
        <p:txEl>
          <p:charRg st="0" end="9"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>ppt_x</p:attrName>
      <p:attrName>ppt_y</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
</p:animMotion>
```

end example]

Attributes	Description
end (End)	This attribute defines the end of the index range. The possible values for this attribute are defined by the ST_Index simple type (§19.7.3).
st (Start)	This attribute defines the start of the index range. The possible values for this attribute are defined by the ST_Index simple type (§19.7.3).

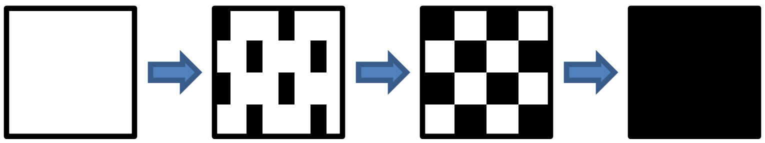
[*Note*: The W3C XML Schema definition of this element's content model (CT_IndexRange) is located in §A.3. *end note*]

19.5.24 checker (Checker Slide Transition)

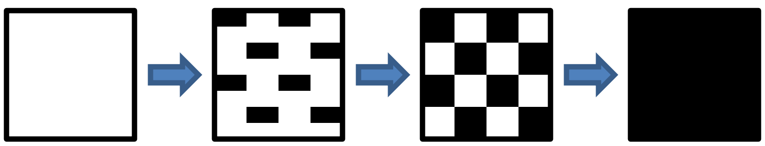
This element describes the checker slide transition effect, which uses a set of horizontal or vertical checkerboard squares and wipes them either left-to-right or top-to-bottom, respectively, until the new slide is fully shown. The rendering of this transition depends upon the attributes specified.

[*Example*: Consider the following cases in which the “checker” slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
  <p:checker dir="horz"/>
</p:transition>
```



```
<p:transition>
  <p:checker dir="vert"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
dir (Transition Direction)	This attribute specifies a horizontal or vertical transition. The possible values for this attribute are defined by the ST_Direction simple type (§19.7.2).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_OrientationTransition) is located in §A.3. *end note*]

19.5.25 childTnLst (Children Time Node List)

This element describes the list of time nodes that have a fixed location in the timing tree based on their parent time node. The children's start time is defined relative to their parent time node’s start.

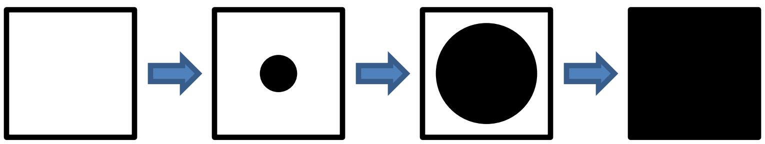
[*Note*: The W3C XML Schema definition of this element’s content model (CT_TimeNodeList) is located in §A.3. *end note*]

19.5.26 circle (Circle Slide Transition)

This element describes the circle slide transition effect, which uses a circle pattern centered on the slide that increases in size until the new slide is fully shown. The rendering of this transition has been shown below.

[*Example*: Consider the following case in which the “circle” slide transition is applied to a slide, along with a set of attributes. The proper usage and a sample rendering are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
  <p:circle/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Empty) is located in §A.3. *end note*]

19.5.27 clrVal (Color Value)

This element describes the color variant. This is used to specify a color that is to be used for animating the color property of an object.

[*Example*: Consider trying to emphasize text within a shape by changing the color its font.

```
<p:set>
  <p:cBhvr override="childStyle">
    …
  </p:cBhvr>
  <p:to>
    <p:clrVal>
      <a:schemeClr val="accent2"/>
    </p:clrVal>
  </p:to>
</p:set>
```

end example]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Color) is located in §A.4.1. *end note*]

19.5.28 cmd (Command)

This element describes the several non-durational commands that can be executed within a timeline. This can be used to send events, call functions on elements, and send verbs to embedded objects. For example "Object Action" effects for Embedded objects and Media commands for sounds/movies such as "PlayFrom(0.0)" and "togglePause".

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>cmd (Command)</td><td>This attribute defines the actual command to be issued. Depending on the command specified, the actual command can be made to invoke a wide range of actions on the linked or embedded object.

Reserved Values (when type = "call"):
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>play</td><td>play corresponding media</td></tr>
<tr><td>playFrom(s)</td><td>play corresponding media starting from s, where s is the number of seconds from the beginning of the clip</td></tr>
<tr><td>pause</td><td>pause corresponding media</td></tr>
<tr><td>resume</td><td>resume play of corresponding media</td></tr>
<tr><td>stop</td><td>stop play of corresponding media</td></tr>
<tr><td>togglePause</td><td>play corresponding media if media is already paused, pause corresponding media if media is already playing. If the corresponding media is not active, this command restarts the media and plays from its beginning.</td></tr>
</table>

Reserved Values (when type = "evt"):
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>onstopaudio</td><td>stop play of all audio</td></tr>
</table>

Reserved Values (when type = "verb"):
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>Open the object for editing</td></tr>
<tr><td>1</td><td>Open the object for viewing</td></tr>
</table>

The value of the cmd attribute shall be the string representation of an integer that represents the embedded object verb number. This verb number determines the action that the rendering application should take corresponding to this object when this point in the animation is reached.

[Example: Consider the following command</td></tr>
</table>

Attributes	Description
	`<p:cmd type="evt" cmd="onstopaudio">` `  <p:cBhvr>` `    <p:cTn display="0" masterRel="sameClick">` `      <p:stCondLst>` `        <p:cond evt="begin" delay="0">` `          <p:tn val="5"/>` `        </p:cond>` `      </p:stCondLst>` `    </p:cTn>` `    <p:tgtEl>` `      <p:sldTgt/>` `    </p:tgtEl>` `  </p:cBhvr>` `</p:cmd>` *end example*] In the above example, the event of onstopaudio stops all audio from playing once this particular animation is reached in the timeline. The possible values for this attribute are defined by the W3C XML Schema string datatype.
type (Command Type)	This attribute specifies the kind of command that is issued by the rendering application to the appropriate target application or object. There are three possible values, `call`, `evt`, and `verb`. A `call` command type is used to specify the class of commands that can then be issued. Call commands (type=”`call`”): This command type is used to call methods on the object specified (`play`(), `pause`(), etc.) Event Commands (type=”`evt`”): This command type is used to set an event for the object at this point in the timeline (`onstopaudio`, etc.) Verb Commands (type=”`verb`”): This command type is used to set verbs for the object to occur at this point in the timeline (0, 1, etc.) The possible values for this attribute are defined by the ST_TLCommandType simple type (§19.7.32).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_TLCommandBehavior) is located in §A.3. *end note*]

19.5.29 cMediaNode (Common Media Node Properties)

This element is used to describe behavior of media elements, such as sound or movies, in an animation.

[*Example*: Consider a shape with a sound effect attached to its animation. The <cMediaNode> element should be used as follows:

```
<p:audio>
  <p:cMediaNode mute="1">
    <p:cTn display="0" masterRel="sameClick">
      <p:stCondLst> … </p:stCondLst>
      <p:endCondLst> … </p:endCondLst>
    </p:cTn>
    <p:tgtEl> … </p:tgtEl>
  </p:cMediaNode>
</p:audio>
```

end example]

Attributes	Description
mute (Mute)	This attribute describes whether the media should be mute. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
numSld (Number of Slides)	This attribute describes the numbers of slides across which the media should play. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
showWhenStopped (Show When Stopped)	This attribute describes whether the media should be displayed when it is stopped. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
vol (Volume)	This attribute describes the volume of the media element. The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§20.1.10.44).

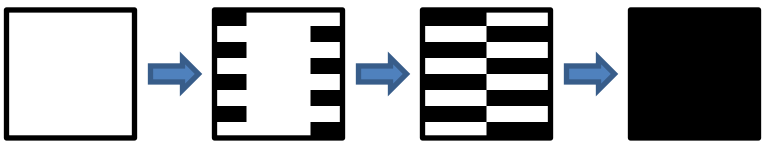
[*Note*: The W3C XML Schema definition of this element's content model (CT_TLCommonMediaNodeData) is located in §A.3. *end note*]

19.5.30 comb (Comb Slide Transition)

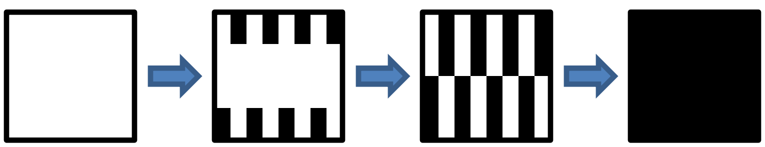
This element describes the comb slide transition effect, which uses a set of horizontal or vertical bars and wipes them from one end of the slide to the other until the new slide is fully shown. The rendering of this transition depends upon the attributes specified which have been shown below.

[*Example*: Consider the following cases in which the “comb” slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
  <p:comb dir="horz"/>
</p:transition>
```



```
<p:transition>
  <p:comb dir="vert"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
dir (Transition Direction)	This attribute specifies a horizontal or vertical transition. The possible values for this attribute are defined by the ST_Direction simple type (§19.7.2).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_OrientationTransition) is located in §A.3. *end note*]

19.5.31 cond (Condition)

This element specifies conditions on time nodes in a timeline. It is used within a list of start condition or list of end condition elements.

[*Example*: For example, suppose we have a shape with a two second delay after the animation is started.

```
<p:cTn>
  <p:stCondLst>
    <p:cond delay="2000"/>
  </p:stCondLst>
  <p:childTnLst>
    <p:set> … </p:set>
    <p:animEffect transition="in" filter="blinds(horizontal)">
      <p:cBhvr>
        <p:cTn id="7" dur="1000"/>
        <p:tgtEl>
          <p:spTgt spid="4"/>
        </p:tgtEl>
      </p:cBhvr>
    </p:animEffect>
  </p:childTnLst>
</p:cTn>
```

end example]

Attributes	Description
delay (Trigger Delay)	This attribute describes the delay after an animation is triggered. The possible values for this attribute are defined by the ST_TLTime simple type (§19.7.38).
evt (Trigger Event)	This attribute describes the event that triggers an animation. The possible values for this attribute are defined by the ST_TLTriggerEvent simple type (§19.7.48).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeCondition) is located in §A.3. *end note*]

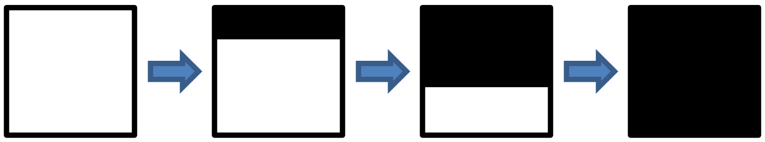
19.5.32 cover (Cover Slide Transition)

This element describes the cover slide transition effect, which moves the new slide in from an off-screen location, continually covering more of the previous slide until the new slide is fully shown. The rendering of this transition depends upon the attributes specified which have been shown below.

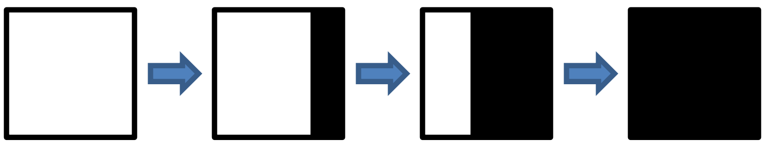
[*Example*: Consider the following cases in which the "cover" slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
```

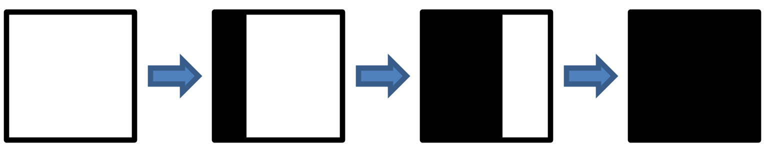
```
  <p:cover dir="d"/>
</p:transition>
```



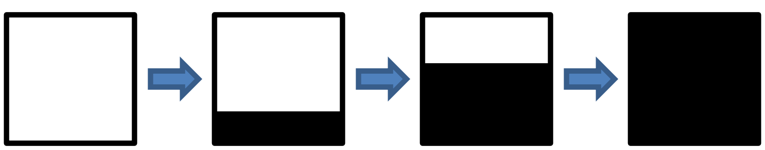
```
<p:transition>
  <p:cover dir="l"/>
</p:transition>
```



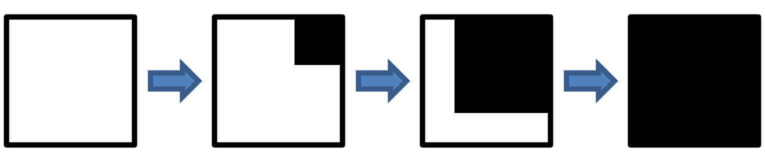
```
<p:transition>
  <p:cover dir="r"/>
</p:transition>
```



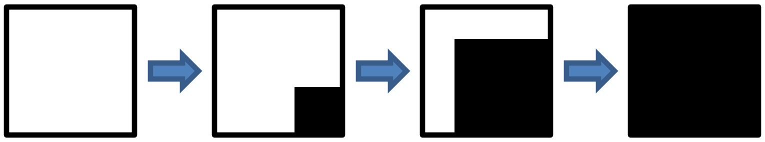
```
<p:transition>
  <p:cover dir="u"/>
</p:transition>
```



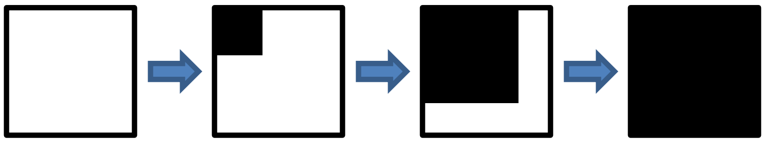
```
<p:transition>
  <p:cover dir="ld"/>
</p:transition>
```



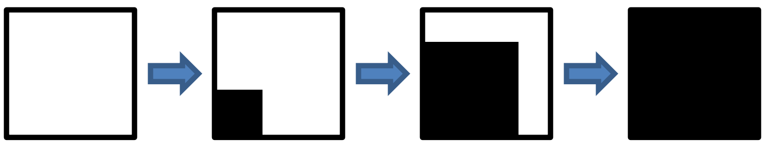
```
<p:transition>
  <p:cover dir="lu"/>
</p:transition>
```



```
<p:transition>
  <p:cover dir="rd"/>
</p:transition>
```



```
<p:transition>
  <p:cover dir="ru"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
dir (Direction)	This attribute specifies if the direction of the transition. The possible values for this attribute are defined by the ST_TransitionEightDirectionType simple type (§19.7.51).

[*Note*: The W3C XML Schema definition of this element's content model (CT_EightDirectionTransition) is located in §A.3. *end note*]

19.5.33 cTn (Common Time Node Properties)

This element describes the properties that are common for time nodes.

Attributes	Description
accel (Acceleration)	This attribute describes the percentage of specified duration over which the element's time takes to accelerate from 0 up to the "run rate." The possible values for this attribute are defined by the ST_PositiveFixedPercentage

Attributes	Description
	simple type (§20.1.10.44).
afterEffect (After Effect)	This attribute specifies whether there is an after effect applied to the time node. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
autoRev (Auto Reverse)	This attribute describes whether to automatically play the animation in reverse after playing it in the forward direction. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
bldLvl (Build level)	This attribute describes the build level of the animation. The possible values for this attribute are defined by the W3C XML Schema int datatype.
decel (Deceleration)	This attribute describes the percentage of specified duration over which the element's time takes to decelerate from the "run rate" down to 0. The possible values for this attribute are defined by the ST_PositiveFixedPercentage simple type (§20.1.10.44).
display (Display)	This attribute describes whether the state of the time node is visible or hidden. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
dur (Duration)	This attribute describes the duration of the time node, expressed as unit time. The possible values for this attribute are defined by the ST_TLTime simple type (§19.7.38).
evtFilter (Event Filter)	This attribute describes the event filter for this time node. The possible values for this attribute are defined by the W3C XML Schema string datatype.
fill (Fill)	This attribute describes the fill type for the time node. The possible values for this attribute are defined by the ST_TLTimeNodeFillType simple type (§19.7.41).
grpId (Group ID)	This attribute describes the Group ID of the time node. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
id (ID)	This attribute specifies the identifier for the timenode. The possible values for this attribute are defined by the ST_TLTimeNodeID simple type

Attributes	Description
	(§19.7.42).
masterRel (Master Relation)	This attribute specifies how the time node plays back relative to its master time node. The possible values for this attribute are defined by the ST_TLTimeNodeMasterRelation simple type (§19.7.43).
nodePh (Node Placeholder)	This attribute describes whether this node is a placeholder. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
nodeType (Node Type)	This attribute specifies the type of time node. The possible values for this attribute are defined by the ST_TLTimeNodeType simple type (§19.7.47).
presetClass (Preset Types)	This attribute descries the class of effect in which it belongs. The possible values for this attribute are defined by the ST_TLTimeNodePresetClassType simple type (§19.7.44).
presetID (Preset ID)	This attribute describes the preset identifier for the time node. The possible values for this attribute are defined by the W3C XML Schema int datatype.
presetSubtype (Preset SubType)	This attribute is a bitflag that specifies a direction or some other attribute of the effect. For example it can be set to specify a “From Bottom” for the Fly In effect, or “Bold” for the Change Font Style effect. The possible values for this attribute are defined by the W3C XML Schema int datatype.
repeatCount (Repeat Count)	This attribute describes the number of times the element should repeat, in units of thousandths. The possible values for this attribute are defined by the ST_TLTime simple type (§19.7.38).
repeatDur (Repeat Duration)	This attribute describes the amount of time over which the element should repeat. If absent, the attribute is taken to be the same as the specified duration. The possible values for this attribute are defined by the ST_TLTime simple type (§19.7.38).
restart (Restart)	This attribute specifies if a node is to restart when it completes its action. The possible values for this attribute are defined by the ST_TLTimeNodeRestartType simple type (§19.7.45).
spd (Speed)	This attribute specifies the percentage by which to speed up (or slow down) the timing. If negative, the timing is reversed. [*Example*: if speed is 200% and the specified duration is 10 seconds, the actual duration is 5 seconds. *end example*]

Attributes	Description
	The possible values for this attribute are defined by the ST_Percentage simple type (§20.1.10.40).
syncBehavior (Synchronization Behavior)	This attribute specifies how the time node synchronizes to its group. The possible values for this attribute are defined by the ST_TLTimeNodeSyncType simple type (§19.7.46).
tmFilter (Time Filter)	This attribute specifies the time filter for the time node. The possible values for this attribute are defined by the W3C XML Schema string datatype.

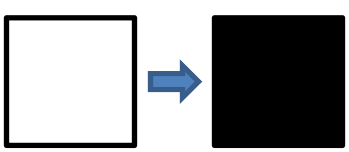
[*Note*: The W3C XML Schema definition of this element's content model (CT_TLCommonTimeNodeData) is located in §A.3. *end note*]

19.5.34 cut (Cut Slide Transition)

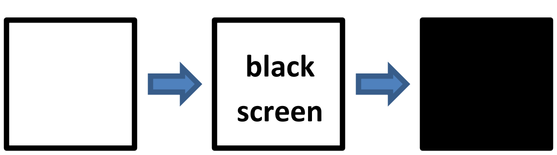
This element describes the cut slide transition effect, which simply replaces the previous slide with the new slide instantaneously. No animation is used, but an option exists to cut to a black screen before showing the new slide. The rendering of this transition depends upon the attributes specified which have been shown below.

[*Example*: Consider the following cases in which the "cut" slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
  <p:cut thruBlk="0"/>
</p:transition>
```



```
<p:transition>
  <p:cut thruBlk="1"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
thruBlk (Transition Through Black)	This attribute specifies if the transition starts from a black screen (and then transition the new slide over black). The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

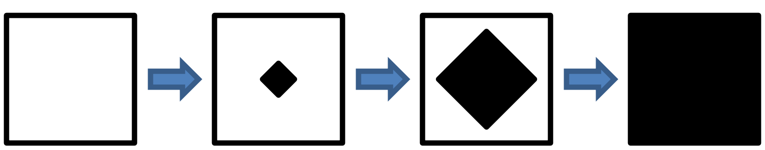
[*Note*: The W3C XML Schema definition of this element's content model (CT_OptionalBlackTransition) is located in §A.3. *end note*]

19.5.35 diamond (Diamond Slide Transition)

This element describes the diamond slide transition effect, which uses a diamond pattern centered on the slide that increases in size until the new slide is fully shown. The rendering of this transition has been shown below.

[*Example*: Consider the following case in which the "diamond" slide transition is applied to a slide, along with a set of attributes. The proper usage and a sample rendering are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
  <p:diamond/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

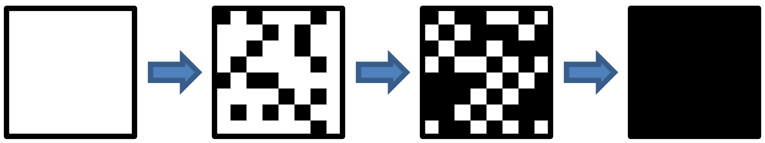
[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.3. *end note*]

19.5.36 dissolve (Dissolve Slide Transition)

This element describes the dissolve slide transition effect, which uses a set of randomly placed squares on the slide that continue to be added to until the new slide is fully shown. The rendering of this transition has been shown below.

[*Example*: Consider the following case in which the “dissolve” slide transition is applied to a slide, along with a set of attributes. The proper usage and a sample rendering are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
  <p:dissolve/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

[*Note*: The W3C XML Schema definition of this element’s content model (CT_Empty) is located in §A.3. *end note*]

19.5.37 endCondLst (End Conditions List)

This element describes a list of the end conditions that shall be met in order to stop the time node.

[*Example*: Consider a shape a shape with an audio attached to the animation. The <endCondList> element should be used as follows to specifies when the sound is done:

```
<p:audio>
  <p:cMediaNode>
    <p:cTn display="0" masterRel="sameClick">
      <p:stCondLst> … </p:stCondLst>
      <p:endCondLst>
        <p:cond evt="onStopAudio" delay="0">
          <p:tgtEl>
            <p:sldTgt/>
          </p:tgtEl>
        </p:cond>
      </p:endCondLst>
    </p:cTn>
    <p:tgtEl> … </p:tgtEl>
  </p:cMediaNode>
</p:audio>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeConditionList) is located in §A.3. *end note*]

19.5.38 endSnd (Stop Sound Action)

This element stops all previous sounds during a slide transition.

[*Example*: Consider a slide transition that stops all previous sounds. The<endSnd> element should be used as follows:

```
<p:transition>
    <p:sndAc>
        <p:endSnd/>
    </p:sndAc>
</p:transition>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.3. *end note*]

19.5.39 endSync (EndSync)

This element is used to synchronizes the stopping of parallel elements in the timing tree. It is used on interactive timeline sequences to specify that the interactive sequence's duration ends when all of the child timenodes have ended. It is also used to make interactive sequences restart-able (so that the entire interactive sequence can be repeated if the trigger object is clicked on repeatedly).

[*Example*: Consider a shape with a fill change animation. The <endSync> element should be used as follows:

```
<p:seq concurrent="1" nextAc="seek">
  <p:cTn>
    <p:stCondLst/>
    <p:endSync evt="end" delay="0">
      <p:rtn val="all"/>
    </p:endSync>
    <p:childTnLst/>
  </p:cTn>
  <p:nextCondLst/>
</p:seq>
```

end example]

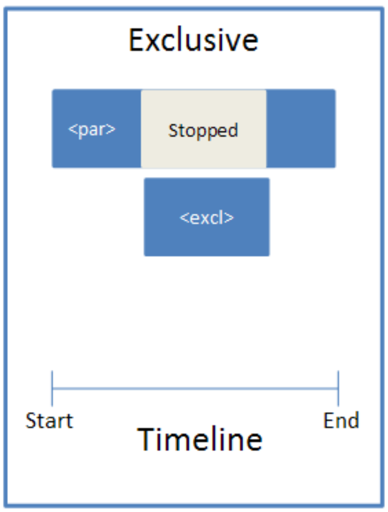
Attributes	Description
delay (Trigger Delay)	This attribute describes the delay after an animation is triggered. The possible values for this attribute are defined by the ST_TLTime simple type (§19.7.38).

Attributes	Description
evt (Trigger Event)	This attribute describes the event that triggers an animation. The possible values for this attribute are defined by the ST_TLTriggerEvent simple type (§19.7.48).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeCondition) is located in §A.3. *end note*]

19.5.40 excl (Exclusive)

This element describes the Exclusive time node. This time node is used to pause all other timelines when it is activated. Conceptually it can be though of as follows:



[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeNodeExclusive) is located in §A.3. *end note*]

19.5.41 fade (Fade Slide Transition)

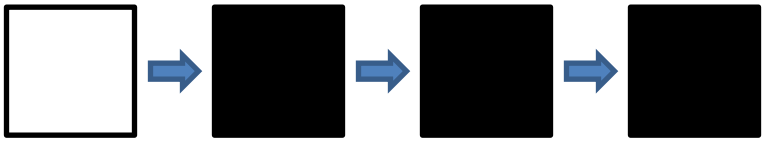
This element describes the fade slide transition effect, which smoothly fades the previous slide either directly to the new slide or first to a black screen and then to the new slide. The rendering of this transition depends upon the attributes specified which have been shown below.

[*Example*: Consider the following cases in which the "fade" slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

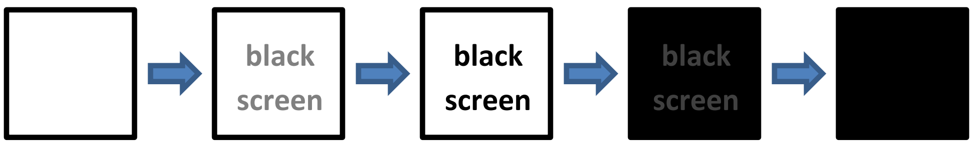
```
<p:transition>
  <p:fade thruBlk="0"/>
```



```
</p:transition>
```



```
<p:transition>
  <p:fade thruBlk="1"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
thruBlk (Transition Through Black)	This attribute specifies if the transition starts from a black screen (and then transition the new slide over black). The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_OptionalBlackTransition) is located in §A.3. *end note*]

19.5.42 fltVal (Float Value)

This element specifies a floating point value to be used for evaluation by a parent element. The exact meaning of the value contained within this element is not defined here but is dependent on the usage of this element in conjunction with one of the listed parent elements.

Attributes	Description
val (Value)	This attribute specifies the floating point value that this element contains and that should be used in evaluating this element. The possible values for this attribute are defined by the W3C XML Schema float datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimVariantFloatVal) is located in §A.3. *end note*]

19.5.43 from (From)

This element is used to specify the starting color of the target element.

[*Example*: Consider a shape with an animation fill change from one accent color to another. The <from> element should be used as follows:

```
<p:animClr clrSpc="rgb" dir="cw">
  <p:cBhvr>
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl> … </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>fillcolor</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:from>
    <a:schemeClr val="accent3"/>
  </p:from>
  <p:to>
    <a:schemeClr val="accent2"/>
  </p:to>
</p:animClr>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Color) is located in §A.4.1. *end note*]

19.5.44 from (From)

This element specifies an x/y co-ordinate to start the animation from.

[*Example*: Consider a shape with an animation sequence that needs to start at a certain coordinate. The <from> element should be used as follows:

```
<p:animScale>
  <p:cBhvr>
    <p:cTn> … </p:cTn>
    <p:tgtEl>
      <p:spTgt spid="4"/>
    </p:tgtEl>
  </p:cBhvr>
  <p:from x="100%" y="100%"/>
  <p:to x="80%" y="100%"/>
```



```
</p:animScale>
```

end example]

Attributes	Description
x (X coordinate)	This attribute describes the X coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§20.1.10.40).
y (Y coordinate)	This attribute describes the Y coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§20.1.10.40).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLPoint) is located in §A.3. *end note*]

19.5.45 graphicEl (Graphic Element)

This element specifies a graphical element which to animate.

[*Example*: Consider a diagram with an animation effect applied to it. The <graphicEl> element should be used as follows:

```
<p:set>
  <p:cBhvr>
    <p:cTn id="6" dur="1" fill="hold"> … </p:cTn>
    <p:tgtEl>
      <p:spTgt spid="4">
        <p:graphicEl>
          <a:dgm id="{87C2C707-C3F4-4E81-A967-A8B8AE13E575}"/>
        </p:graphicEl>
      </p:spTgt>
    </p:tgtEl>
    <p:attrNameLst> … </p:attrNameLst>
  </p:cBhvr>
  <p:to> … </p:to>
</p:set>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_AnimationElementChoice) is located in §A.4.1. *end note*]

19.5.46 hsl (HSL)

This element specifies an incremental HSL (Hue, Saturation, Lightness) value to add to a color animation.

[*Example*: Consider a shape with a lightening emphasis animation. The <hsl> element should be used as follows:

```
<p:animClr clrSpc="hsl">
  <p:cBhvr>
    <p:cTn id="8" dur="500" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="4"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>stroke.color</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:by>
    <p:hsl h="0" s="0" l="0"/>
  </p:by>
</p:animClr>
```

end example]

Attributes	Description
h (Hue)	Specifies hue as an angle. The values range from [0, 360] degrees. The possible values for this attribute are defined by the ST_Angle simple type (§20.1.10.3).
l (Lightness)	Specifies a lightness as a percentage. The values are in the range [-100%, 100%]. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§20.1.10.24).
s (Saturation)	Specifies a saturation as a percentage. The values are in the range [-100%, 100%]. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§20.1.10.24).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLByHslColorTransform) is located in §A.3. *end note*]

19.5.47 inkTgt (Ink Target)

This element specifies an animation target element that is represented by a sub-shape in a legacy graphical object.

[*Example*: Consider an ink diagram with an animation blinds transition effect applied to it. The <inkTgt> element should be used as follows:

```
<p:animEffect transition="in" filter="blinds(horizontal)">
  <p:cBhvr>
    <p:cTn id="7" dur="500"/>
    <p:tgtEl>
      <p:inkTgt spid="_x0000_s2057"/>
    </p:tgtEl>
  </p:cBhvr>
</p:animEffect>
```

end example]

Attributes	Description
spid (Shape ID)	This attribute specifies the shape identifier. The possible values for this attribute are defined by the ST_DrawingElementId simple type (§20.1.10.21).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLSubShapeId) is located in §A.3. *end note*]

19.5.48 intVal (Integer)

This element specifies an integer value to be used for evaluation by a parent element. The exact meaning of the value contained within this element is not defined here but is dependent on the usage of this element in conjunction with one of the listed parent elements.

Attributes	Description
val (Value)	This attribute specifies the integer value that this element contains and that should be used in evaluating this element. The possible values for this attribute are defined by the W3C XML Schema int datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimVariantIntegerVal) is located in §A.3. *end note*]

19.5.49 iterate (Iterate)

This element specifies how the animation should be successively applied to sub elements of the target element for a repeated effect. It can be applied to contained timing and animation structures over the letters, words, or shapes within a target element.

[*Example*: Consider a text animation where the words appear letter by letter. The <iterate> element should be used as follows:

```
<p:par>
  <p:cTn id="1" >
    <p:stCondLst> … </p:stCondLst>
    <p:iterate type="lt">
      <p:tmPct val="10000"/>
    </p:iterate>
    <p:childTnLst> … </p:childTnLst>
  </p:cTn>
</p:par>
```

end example]

Attributes	Description
backwards (Backwards)	This attribute specifies whether to go backwards in the timeline to the previous node. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
type (Iterate Type)	This attribute specifies the iteration behavior and applies it to each letter, word or shape within a container element. Values are by word, by letter, or by element. If there is no text or block elements such as shapes within the container or a single word, letter, or shape (depending on iterate type) then no iteration happens and the behavior is applied to the element itself instead. The possible values for this attribute are defined by the ST_IterateType simple type (§19.7.4).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLIterateData) is located in §A.3. *end note*]

19.5.50 newsflash (Newsflash Slide Transition)

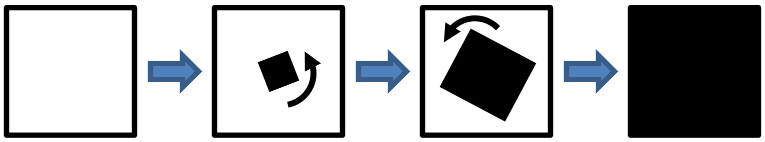
This element describes the newsflash slide transition effect, which grows and spins the new slide counter-clockwise into place over the previous slide. The rendering of this transition has been shown below.

[*Example*: Consider the following case in which the "newsflash" slide transition is applied to a slide, along with a set of attributes. The proper usage and a sample rendering are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
```



```
    <p:newsflash/>
  </p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.3. *end note*]

19.5.51 nextCondLst (Next Conditions List)

This element describes a list of conditions that shall be met to advance to the next animation sequence.

[*Example*: Consider a shape with a text emphasis changing the size of its font.

```
  <p:seq concurrent="1" nextAc="seek">
    <p:cTn id="2" dur="indefinite" nodeType="mainSeq"> … </p:cTn>
    <p:prevCondLst> … </p:prevCondLst>
    <p:nextCondLst>
      <p:cond evt="onNext" delay="0">
        <p:tgtEl>
          <p:sldTgt/>
        </p:tgtEl>
      </p:cond>
    </p:nextCondLst>
  </p:seq>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeConditionList) is located in §A.3. *end note*]

19.5.52 oleChartEl (Embedded Chart Element)

This element specifies the subelement of an embedded chart to animate.

[*Example*: Consider an embedded Chart with a entrance animation effect applied to each of the graph's categories. The <oldChartEl> element should be used as follows:

```
  <p:animEffect transition="in" filter="blinds(horizontal)">
    <p:cBhvr>
```



```
      <p:cTn id="12" dur="500"/>
      <p:tgtEl>
        <p:spTgt spid="19460">
          <p:oleChartEl type="category" lvl="1"/>
        </p:spTgt>
      </p:tgtEl>
    </p:cBhvr>
  </p:animEffect>
```

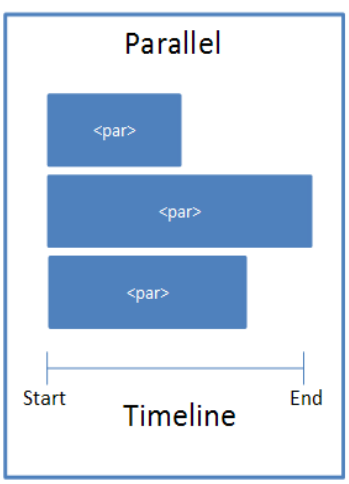
end example]

Attributes	Description
lvl (Level)	This attribute describes the element levels to animate. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.
type (Type)	This attribute specifies how to chart should be built during its animation. The possible values for this attribute are defined by the ST_TLChartSubelementType simple type (§19.7.31).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLOleChartTargetElement) is located in §A.3. *end note*]

19.5.53 par (Parallel Time Node)

This element describes the Parallel time node which can be activated along with other parallel time node containers. Conceptually it can be thought of as follows:



[*Example*: Consider a simple animation with a blind entrance. The <par> element should be used as follows:


```
<p:timing>
  <p:tnLst>
    <p:par>
      <p:cTn id="1" dur="indefinite" restart="never" nodeType="tmRoot">
        <p:childTnLst>
          <p:seq concurrent="1" nextAc="seek">
            …
          </p:seq>
        </p:childTnLst>
      </p:cTn>
    </p:par>
  </p:tnLst>
</p:timing>
```

end example]

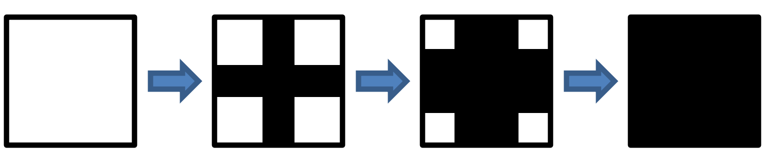
[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeNodeParallel) is located in §A.3. *end note*]

19.5.54 plus (Plus Slide Transition)

This element describes the plus slide transition effect, which uses a plus pattern centered on the slide that increases in size until the new slide is fully shown. The rendering of this transition has been shown below.

[*Example*: Consider the following case in which the "plus" slide transition is applied to a slide, along with a set of attributes. The proper usage and a sample rendering are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
  <p:plus/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.3. *end note*]

19.5.55 prevCondLst (Previous Conditions List)

This element describes a list of conditions that shall be met in order to go backwards in an animation sequence.

[*Example*: Consider trying to emphasize text within a shape by changing the size of its font.

```
<p:seq concurrent="1" nextAc="seek">
  <p:cTn id="2" dur="indefinite" nodeType="mainSeq">
  </p:cTn>
  <p:prevCondLst>
    <p:cond evt="onPrev" delay="0">
      <p:tgtEl>
        <p:sldTgt/>
      </p:tgtEl>
    </p:cond>
  </p:prevCondLst>
  <p:nextCondLst>
  </p:nextCondLst>
</p:seq>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeConditionList) is located in §A.3. *end note*]

19.5.56 pRg (Paragraph Text Range)

This element specifies a text range to animate based on starting and ending paragraph number.

[*Example*: Consider an animation entrance of the first 3 text paragraphs. The <pRg> element should be used as follows:

```
<p:animEffect transition="in" filter="checkerboard(across)">
  <p:cBhvr>
    <p:cTn id="12" dur="500"/>
    <p:tgtEl>
      <p:spTgt spid="3">
        <p:txEl>
          <p:pRg st="0" end="2"/>
        </p:txEl>
      </p:spTgt>
    </p:tgtEl>
  </p:cBhvr>
</p:animEffect>
```


end example]

Attributes	Description
end (End)	This attribute defines the end of the index range. The possible values for this attribute are defined by the ST_Index simple type (§19.7.3).
st (Start)	This attribute defines the start of the index range. The possible values for this attribute are defined by the ST_Index simple type (§19.7.3).

[*Note*: The W3C XML Schema definition of this element's content model (CT_IndexRange) is located in §A.3. *end note*]

19.5.57 progress (Progress)

This element defines the progression of an animation. The default for the way animation progress happens through an animEffect is a linear ramp from 0 to 1, starting at the effect's begin time & ending at the effect's end time. When you specify a value for the progress attribute, you are overriding this default behaviour. The value between 0 and 1 represents a percentage through the effect, where 0 is 0% and 1 is 100%.

Each animEffect is in fact an object-based transition. These transitions can be specified as "In" (where the object is not visible at 0% and becomes completely visible at 100%) or "Out" (where the object is visible at 0% and becomes completely invisible at 100%). You would set the progress attribute if you want to use the animEffect as a "static" effect, where the transition properties do not actually change over time. As an alternative to using the progress attribute, you can use the tmFilter (time filter), which is a base attribute of any effect/timenode, to specify the way that progress through an effect should be performed dynamically.

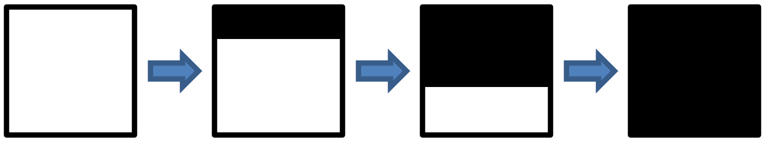
[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimVariant) is located in §A.3. *end note*]

19.5.58 pull (Pull Slide Transition)

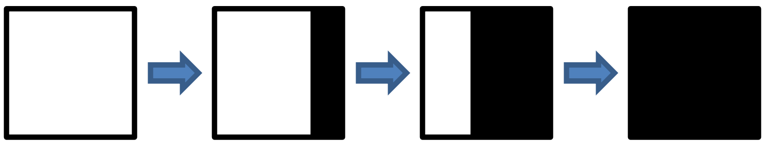
This element describes the pull slide transition effect, which moves the previous slide to an off-screen location, continually revealing more of the new slide until the new slide is fully shown. The rendering of this transition depends upon the attributes specified which have been shown below.

[*Example*: Consider the following cases in which the "pull" slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

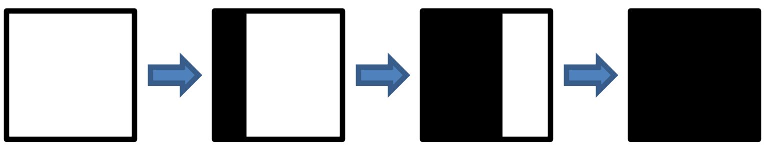
```
<p:transition>
  <p:pull dir="d"/>
</p:transition>
```

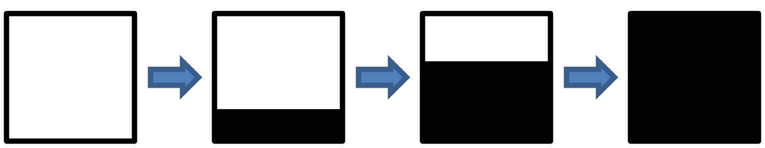
```
<p:transition>
  <p:pull dir="l"/>
</p:transition>
```



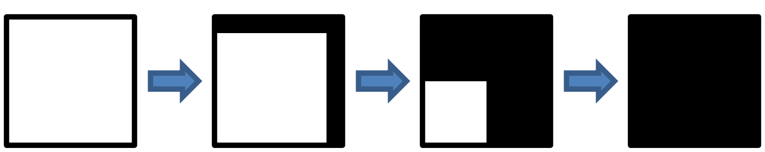
```
<p:transition>
  <p:pull dir="r"/>
</p:transition>
```



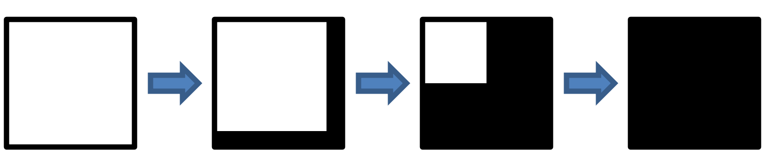
```
<p:transition>
  <p:pull dir="u"/>
</p:transition>
```



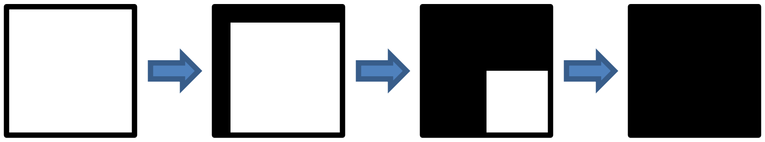
```
<p:transition>
  <p:pull dir="ld"/>
</p:transition>
```



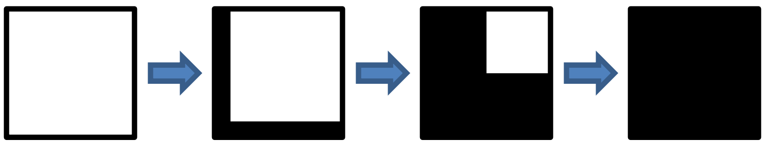
```
<p:transition>
  <p:pull dir="lu"/>
</p:transition>
```



```
<p:transition>
  <p:pull dir="rd"/>
</p:transition>
```



```
<p:transition>
  <p:pull dir="ru"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
dir (Direction)	This attribute specifies if the direction of the transition. The possible values for this attribute are defined by the ST_TransitionEightDirectionType simple type (§19.7.51).

[*Note*: The W3C XML Schema definition of this element's content model (CT_EightDirectionTransition) is located in §A.3. *end note*]

19.5.59 push (Push Slide Transition)

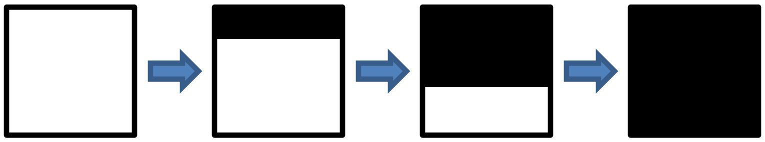
This element describes the push slide transition effect, which moves the new slide in from an off-screen location, continually pushing the previous slide to an opposite off-screen location until the new slide is fully shown. The rendering of this transition depends upon the attributes specified which have been shown below.

[*Example*: Consider the following cases in which the "push" slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

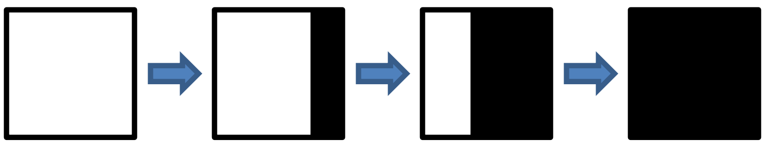
```
<p:transition>
  <p:push dir="d"/>
```



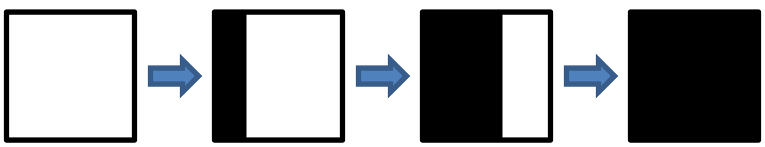
```
</p:transition>
```



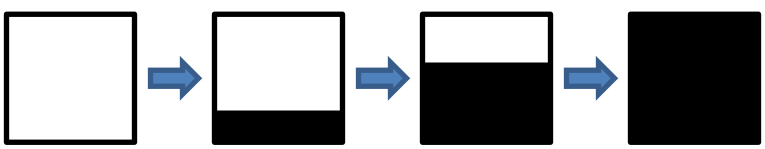
```
<p:transition>
  <p:push dir="l"/>
</p:transition>
```



```
<p:transition>
  <p:push dir="r"/>
</p:transition>
```



```
<p:transition>
  <p:push dir="u"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
dir (Direction)	This attribute specifies the direction of the slide transition. The possible values for this attribute are defined by the ST_TransitionSideDirectionType simple type (§19.7.53).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SideDirectionTransition) is located in §A.3. *end note*]

19.5.60 random (Random Slide Transition)

This element describes the random slide transition effect, which chooses a random transition from the set available in the rendering application. This transition thus can be different each time it is used.

[*Example*: Consider the following case in which the “random” slide transition is applied to a slide, along with a set of attributes. The proper usage is shown below:

```
<p:transition>
  <p:random/>
</p:transition>
```

end example]

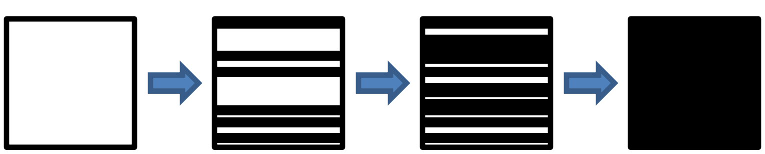
[*Note*: The W3C XML Schema definition of this element’s content model (CT_Empty) is located in §A.3. *end note*]

19.5.61 randomBar (Random Bar Slide Transition)

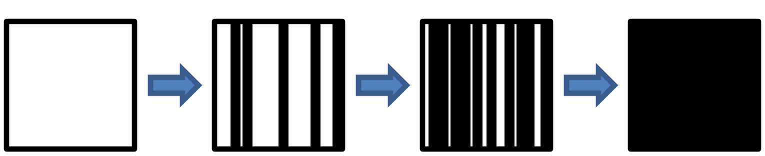
This element describes the randomBar slide transition effect, which uses a set of randomly placed horizontal or vertical bars on the slide that continue to be added to until the new slide is fully shown. The rendering of this transition depends upon the attributes specified which have been shown below.

[*Example*: Consider the following cases in which the “randomBar” slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
  <p:randomBar dir="horz"/>
</p:transition>
```



```
<p:transition>
  <p:randomBar dir="vert"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
dir (Transition Direction)	This attribute specifies a horizontal or vertical transition. The possible values for this attribute are defined by the ST_Direction simple type (§19.7.2).

[*Note*: The W3C XML Schema definition of this element's content model (CT_OrientationTransition) is located in §A.3. *end note*]

19.5.62 rCtr (Rotation Center)

This element describes the center of the rotation used to rotate a motion path by X angle.

[*Example*: For example, suppose we have a simple animation with a checkerbox text entrance.

```
<p:animMotion origin="layout" path="M 0 0  L 0.25 0.33333  E"
pathEditMode="relative" rAng="0" ptsTypes="">
  <p:cBhvr>
    <p:cTn id="6" dur="2000" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="3"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>ppt_x</p:attrName>
      <p:attrName>ppt_y</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:rCtr x="56.7%" y="83.4%"/>
</p:animMotion>
```

end example]

Attributes	Description
x (X coordinate)	This attribute describes the X coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§20.1.10.40).
y (Y coordinate)	This attribute describes the Y coordinate.

Attributes	Description
	The possible values for this attribute are defined by the ST_Percentage simple type (§20.1.10.40).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLPoint) is located in §A.3. *end note*]

19.5.63 rgb (RGB)

The element specifies an incremental RGB value to add to the color property.

[*Example*: Consider a shape with a color emphasis animation. The <rgb> element should be used as follows:

```
<p:animClr clrSpc="rgb">
  <p:cBhvr>
    <p:cTn id="8" dur="500" fill="hold"/>
    <p:tgtEl>
      <p:spTgt spid="4"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>stroke.color</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:by>
    <p:rgb r="10" g="20" b="30"/>
  </p:by>
</p:animClr>
```

end example]

Attributes	Description
b (Blue)	This attribute specifies a blue component luminance as a percentage. Values are in the range [-100%, 100%]. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§20.1.10.24).
g (Green)	This attribute specifies a green component luminance as a percentage. Values are in the range [-100%, 100%]. The possible values for this attribute are defined by the ST_FixedPercentage simple type (§20.1.10.24).
r (Red)	This attribute specifies a red component luminance as a percentage. Values are in the range [-100%, 100%].

Attributes	Description
	The possible values for this attribute are defined by the ST_FixedPercentage simple type (§20.1.10.24).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLByRgbColorTransform) is located in §A.3. *end note*]

19.5.64 rtn (Runtime Node Trigger Choice)

This element specifies the child time node that triggers a time condition. References a child time node or all child nodes. Order is based on the child's end time.

[*Example*: Consider an animation which ends the synchronization of all parallel time nodes when all the child nodes have ended their animation. The <rtn> element should be used as follows:

```
<p:cTn>
  <p:stCondLst> … </p:stCondLst>
  <p:endSync evt="end" delay="0">
    <p:rtn val="all"/>
  </p:endSync>
  <p:childTnLst> … </p:childTnLst>
</p:cTn>
```

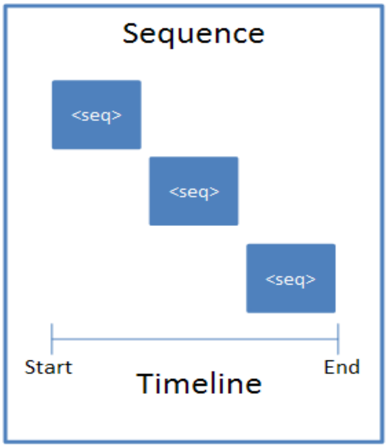
end example]

Attributes	Description
val (Value)	This attribute describes the value that triggers the runtime node. The possible values for this attribute are defined by the ST_TLTriggerRuntimeNode simple type (§19.7.49).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTriggerRuntimeNode) is located in §A.3. *end note*]

19.5.65 seq (Sequence Time Node)

This element describes the Sequence time node and it can only be activated when the one before it finishes. Conceptually it can be though of as follows:



[*Example*: For example, suppose we have a simple animation with a blind entrance.

```
<p:timing>
  <p:tnLst>
    <p:par>
      <p:cTn id="1" dur="indefinite" restart="never" nodeType="tmRoot">
        <p:childTnLst>
          <p:seq concurrent="1" nextAc="seek">
            …
          </p:seq>
        </p:childTnLst>
      </p:cTn>
    </p:par>
  </p:tnLst>
</p:timing>
```

end example]

Attributes	Description
concurrent (Concurrent)	This attribute specifies if concurrency is enabled or disabled. By default this attribute has a value of "disabled". When the value is set to "enabled", the previous element is left enabled when advancing to the next element in a sequence instead of being ended. This is only relevant for advancing via the next condition element being triggered. The only other way to advance to the next element would be to have the current element end, which implies it is no longer concurrent. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.
nextAc (Next Action)	This attribute specifies what to do when going forward in sequence. By default this attribute has a value of "none". When this is set to seek it seeks the element to a natural end time (not necessarily the actual end time).

Attributes	Description
	The natural end position is defined as the latest non-infinite end time of the children. If a child loops forever, the end of its first loop is used as its "end time" for the purposes of this calculation. Some container elements can have infinite durations due to an infinite-duration child element. The engine needs to recurse down through all infinite duration containers to calculate their natural duration in case a child might have non-infinite duration within it that needs to be taken into account. The possible values for this attribute are defined by the ST_TLNextActionType simple type (§19.7.34).
prevAc (Previous Action)	This attribute specifies what to do when going backwards in a sequence. By default it is set to "none" and nothing special is done. When the value is "skipTimed", the sequence continues to go backwards until it reaches a sequence element that was defined to begin only on the next condition element. The possible values for this attribute are defined by the ST_TLPreviousActionType simple type (§19.7.37).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeNodeSequence) is located in §A.3. *end note*]

19.5.66 set (Set Time Node Behavior)

This element allows the setting of a particular property value to a fixed value while the behavior is active and restores the value when the behavior is reset or turned off.

[*Example*: For example, suppose we want to set certain properties during an animation effect. The <set> element should be used as follows:

```
<p:childTnLst>
  <p:set>
    <p:cBhvr>
      <p:cTn id="6" dur="1" fill="hold"> … </p:cTn>
      <p:tgtEl>
        <p:spTgt spid="4"/>
      </p:tgtEl>
      <p:attrNameLst>
        <p:attrName>style.visibility</p:attrName>
      </p:attrNameLst>
    </p:cBhvr>
    <p:to>
      <p:strVal val="visible"/>
```



```
      </p:to>
    </p:set>
    <p:animEffect transition="in" filter="blinds(horizontal)">
      …
    </p:animEffect>
  </p:childTnLst>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLSetBehavior) is located in §A.3. *end note*]

19.5.67 sldTgt (Slide Target)

This element specifies the slide as the target element.

[*Example*: For example, suppose we have a simple animation with a blind entrance.

```
  <p:seq concurrent="1" nextAc="seek">
    <p:cTn id="2" dur="indefinite" nodeType="mainSeq"> … </p:cTn>
    <p:prevCondLst> … </p:prevCondLst>
    <p:nextCondLst>
      <p:cond evt="onNext" delay="0">
        <p:tgtEl>
          <p:sldTgt/>
        </p:tgtEl>
      </p:cond>
    </p:nextCondLst>
  </p:seq>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.3. *end note*]

19.5.68 snd (Sound)

This element specifies the audio information to play during a slide transition.

[*Example*: Consider a slide transition with an audio effect. The <snd> element should be used as follows:

```
      <p:transition>
          <p:sndAc>
              <p:stSnd>
                  <p:snd r:embed="rId2""/>
              </p:stSnd>
          </p:sndAc>
      </p:transition>
```

end example]

Attributes	Description
embed (Embedded Audio File Relationship ID) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the identification information for an embedded audio file. This attribute is used to specify the location of an object that resides locally within the file. [*Note*: A list of suggested audio types is provided in §15.2.2. *end note*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
name (Sound Name) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies the original name or given short name for the corresponding sound. This is used to distinguish this sound from others by providing a human readable name for the attached sound should the user need to identify the sound among others within the UI. The possible values for this attribute are defined by the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_EmbeddedWAVAudioFile) is located in §A.4.1. *end note*]

19.5.69 sndAc (Sound Action)

This element describes a sound action for slide transition. This element specifies that the start of the slide transition is accompanied by the playback of an audio file; the actual audio file used is specified by the snd element (§19.5.68).

[*Example*: Consider a slide transition with a sound effect. The <sndAc> element should be used as follows:

```
<p:transition>
    <p:sndAc>
        <p:stSnd>
            <p:snd r:embed="rId2"/>
        </p:stSnd>
    </p:sndAc>
</p:transition>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TransitionSoundAction) is located in §A.3. *end note*]

19.5.70 sndTgt (Sound Target)

This element describes the sound information for a target object.

[*Example*: Consider a shape with a sound effect animation. The <sndTgt> element should be used as follows:

```
<p:subTnLst>
  <p:audio>
    <p:cMediaNode>
      <p:cTn display="0" masterRel="sameClick"> … </p:cTn>
      <p:tgtEl>
        <p:sndTgt r:embed="rId2" r:link="rId3"/>
      </p:tgtEl>
    </p:cMediaNode>
  </p:audio>
</p:subTnLst>
```

end example]

Attributes	Description
embed (Embedded Audio File Relationship ID) Namespace: http://purl.oclc.org/ooxml/officeDocument/relationships	Specifies the identification information for an embedded audio file. This attribute is used to specify the location of an object that resides locally within the file. [*Note*: A list of suggested audio types is provided in §15.2.2. *end note*] The possible values for this attribute are defined by the ST_RelationshipId simple type (§22.8.2.1).
name (Sound Name) Namespace: http://purl.oclc.org/ooxml/drawingml/main	Specifies the original name or given short name for the corresponding sound. This is used to distinguish this sound from others by providing a human readable name for the attached sound should the user need to identify the sound among others within the UI. The possible values for this attribute are defined by the W3C XML Schema string datatype.

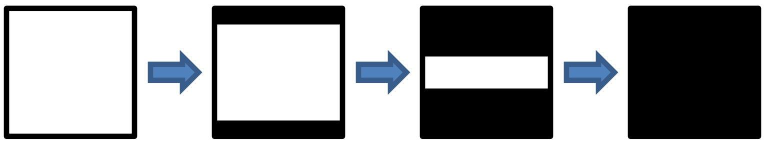
[*Note*: The W3C XML Schema definition of this element's content model (CT_EmbeddedWAVAudioFile) is located in §A.4.1. *end note*]

19.5.71 split (Split Slide Transition)

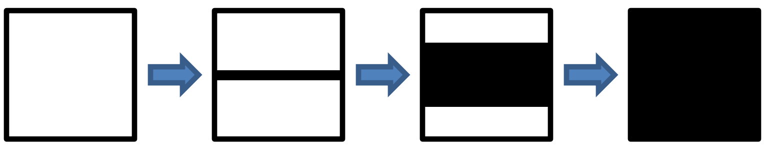
This element describes the split slide transition effect, which reveals the new slide directly on top of the previous one by wiping either horizontal or vertical from the outside in, or from the inside out, until the new slide is fully shown. The rendering of this transition depends upon the attributes specified which have been shown below.

[*Example*: Consider the following cases in which the “split” slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

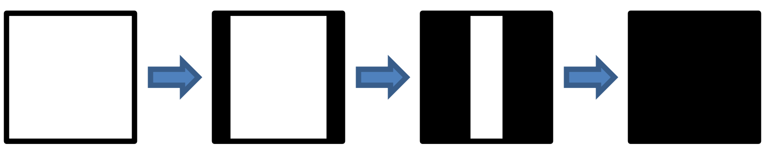
```
<p:transition>
  <p:split orient="horz" dir="in"/>
</p:transition>
```



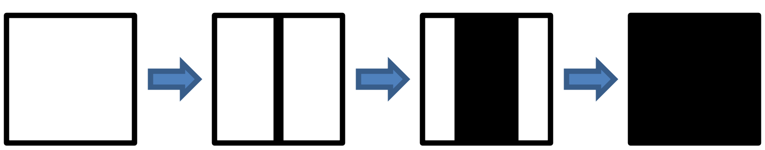
```
<p:transition>
  <p:split orient="horz" dir="out"/>
</p:transition>
```



```
<p:transition>
  <p:split orient="vert" dir="in"/>
</p:transition>
```



```
<p:transition>
  <p:split orient="vert" dir="out"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
dir (Direction)	This attribute specifies the direction of a "split" slide transition.

Attributes	Description
	The possible values for this attribute are defined by the ST_TransitionInOutDirectionType simple type (§19.7.52).
orient (Orientation)	This attribute specifies the orientation of a "split" slide transition. The possible values for this attribute are defined by the ST_Direction simple type (§19.7.2).

[*Note*: The W3C XML Schema definition of this element's content model (CT_SplitTransition) is located in §A.3. *end note*]

19.5.72 spTgt (Shape Target)

The element specifies the shape in which to apply a certain animation to.

[*Example*: Consider a shape whose id is 3 in which we want to apply a fade animation effect. The <spTgt> should be used as follows:

```
<p:animEffect transition="in" filter="fade">
  <p:cBhvr>
    <p:cTn id="7" dur="2000"/>
    <p:tgtEl>
      <p:spTgt spid="3"/>
    </p:tgtEl>
  </p:cBhvr>
</p:animEffect>
```

end example]

Attributes	Description
spid (Shape ID)	This attribute specifies the shape identifier. The possible values for this attribute are defined by the ST_DrawingElementId simple type (§20.1.10.21).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLShapeTargetElement) is located in §A.3. *end note*]

19.5.73 stCondLst (Start Conditions List)

This element contains a list conditions that shall be met for a time node to be activated.

[*Example*: For example, suppose we have a shape with an entrance appearance after 5 seconds. The <stCondLst>element should be used as follows:

```
<p:par>
  <p:cTn id="5" nodeType="clickEffect">
    <p:stCondLst>
      <p:cond delay="5000"/>
    </p:stCondLst>
    <p:childTnLst> … </p:childTnLst>
  </p:cTn>
</p:par>
```

end example]

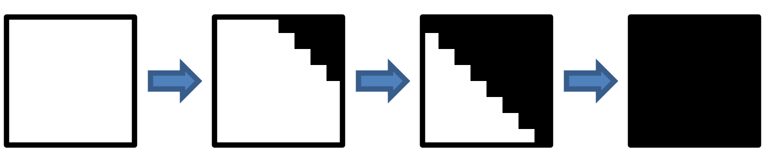
[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeConditionList) is located in §A.3. *end note*]

19.5.74 strips (Strips Slide Transition)

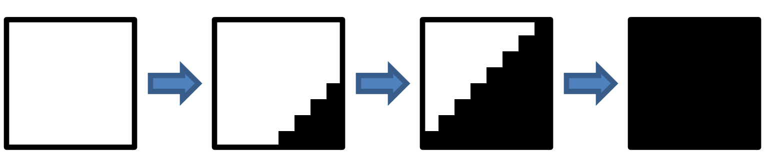
This element describes the strips slide transition effect, which uses a set of bars that are arranged in a staggered fashion and wipes them across the screen until the new slide is fully shown. The rendering of this transition depends upon the attributes specified which have been shown below.

[*Example*: Consider the following cases in which the "strips" slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

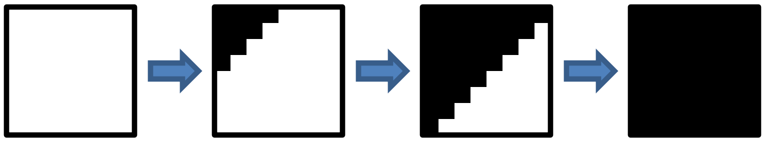
```
<p:transition>
  <p:strips dir="ld"/>
</p:transition>
```



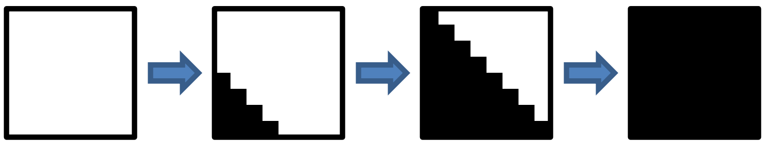
```
<p:transition>
  <p:strips dir="lu"/>
</p:transition>
```



```
<p:transition>
  <p:strips dir="rd"/>
</p:transition>
```

```
<p:transition>
  <p:strips dir="ru"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
dir (Direction)	This attribute specifies if the direction of the transition. The possible values for this attribute are defined by the ST_TransitionCornerDirectionType simple type (§19.7.50).

[*Note*: The W3C XML Schema definition of this element's content model (CT_CornerDirectionTransition) is located in §A.3. *end note*]

19.5.75 strVal (String Value)

This element specifies a string value to be used for evaluation by a parent element. The exact meaning of the value contained within this element is not defined here but is dependent on the usage of this element in conjunction with one of the listed parent elements.

Attributes	Description
val (Value)	This attribute specifies the string value that this element contains and that should be used in evaluating this element. The possible values for this attribute are defined by the W3C XML Schema string datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimVariantStringVal) is located in §A.3. *end note*]

19.5.76 stSnd (Start Sound Action)

This element describes the sound that starts playing during a slide transition.

[*Example*: Consider a slide transition that starts with a sound effect. The <stSnd> element should be used as follows:

```
<p:transition>
    <p:sndAc>
        <p:stSnd>
            <p:snd r:embed="rId2"/>
        </p:stSnd>
    </p:sndAc>
</p:transition>
```

end example]

Attributes	Description
loop (Loop Sound)	This attribute specifies if the sound loops until the next sound event occurs in slideshow. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TransitionStartSoundAction) is located in §A.3. *end note*]

19.5.77 subSp (Subshape)

This element specifies the subshape of a legacy graphical object to animate.

[*Example*: Consider adding animation to a legacy diagram. The <subSp> element should be used as follows:

```
<p:animEffect transition="in" filter="blinds(horizontal)">
  <p:cBhvr>
    <p:cTn id="7" dur="500"/>
    <p:tgtEl>
      <p:spTgt spid="2053">
        <p:subSp spid="_x0000_s70664"/>
      </p:spTgt>
    </p:tgtEl>
  </p:cBhvr>
</p:animEffect>
```

end example]

Attributes	Description
spid (Shape ID)	This attribute specifies the shape identifier. The possible values for this attribute are defined by the ST_DrawingElementId simple type (§20.1.10.21).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLSubShapeId) is located in §A.3. *end note*]

19.5.78 subTnLst (Sub-TimeNodes List)

This element describes time nodes that have a start time which is not based on the containing timenode. It is instead based on their master relationship (masterRel). At runtime, they are inserted dynamically into the timing tree as child timenodes for playback, based on the logic defined by the master relationship. These elements are used for animations such as "dim after" and "play sound effects"

[*Example*: Consider an animation with a "Fly In" effect on paragraphs so that each paragraph flies in on a separate click. Then the "Dim After" effect for paragraph 1 doe not happen until paragraph 2 flies in. The <subTnLst> element should be used as follows:

```
<p:par>
  <p:cTn id="5" grpId="0" nodeType="clickEffect">
    <p:stCondLst> … </p:stCondLst>
    <p:childTnLst> … </p:childTnLst>
    <p:subTnLst>
      <p:set>
        <p:cBhvr override="childStyle">
          <p:cTn fill="hold" masterRel="nextClick" afterEffect="1"/>
          <p:tgtEl> … </p:tgtEl>
          <p:attrNameLst> … </p:attrNameLst>
        </p:cBhvr>
        <p:to> … </p:to>
      </p:set>
    </p:subTnLst>
  </p:cTn>
</p:par>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TimeNodeList) is located in §A.3. *end note*]

19.5.79 tav (Time Animate Value)

This element defines a "keypoint" in animation interpolation.

[*Example*: Consider a shape with a "fly-in" animation. The <tav> element should be used as follows:

```
<p:anim calcmode="lin" valueType="num">
  <p:cBhvr additive="base"> … </p:cBhvr>
  <p:tavLst>
    <p:tav tm="0%">
      <p:val>
        <p:strVal val="1+#ppt_h/2"/>
      </p:val>
    </p:tav>
    <p:tav tm="100%">
      <p:val>
        <p:strVal val="#ppt_y"/>
      </p:val>
    </p:tav>
  </p:tavLst>
</p:anim>
```

end example]

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td>fmla (Formula)</td><td>This attribute allows for the specification of a formula to be used for describing a complex motion for an animated object. The formula manipulates the motion of the object by modifying a property of the object over a specified period of time. Each formula has zero or more inputs specified by the ($) symbol, zero or more variables specified by the (#) symbol pre-pended to the variable name and a target variable which is specified by the previously specified attrName element. The formula can contain one or more of any of the constants, operators or functions listed below. In addition to this, the formula can also contain floating point numbers and parentheses.

Mathematical operations have the following order of precedence, listed from lowest to highest. Operators listed on the same line have equal precedence.
• “+”, “-“
• “*”, “/”, “%”
• “^”
• Unary minus, Unary plus (e.g. -2, meaning 3*-2 is the same as 3*(-2))
• Variables, Constants (including numbers) and Functions (as listed previously)

Language Description:

<code>Digit   = '0' | '1' | ‘2’ | ‘3’ | ‘4’ | ‘5’ | ‘6’ | ‘7’</code>
<code> | ‘8’ | '9' ;</code></td></tr>
</table>

Attributes	Description
	`number   = digit , { digit } ;` `exponent = [ '-' ] , ( 'e' | 'E' ) , number ;` `value    = number , [ '.' number ] , [ exponent ] ;` `variable = '$' \| 'ppt_x' \| 'ppt_y' \| 'ppt_w' \| 'ppt_h' ;` `constant = value \| 'pi' \| 'e' ;` `ident    = 'abs' \| ‘acos’ \| ‘asin’ \| ‘atan’ \| ‘ceil’` `   \| ‘cos’ \| ‘cosh’ \| ‘deg’ \| ‘exp’ \| ‘floor’ \| ‘ln’` `   \| ‘max’ \| ‘min’ \| ‘rad’ \| ‘rand’ \| ‘sin’ \| ‘sinh’` `   \| ‘sqrt’ \| ‘tan’ \| 'tanh' ;` `function = ident , '(' , formula [ ',' , formula ] , ')' ;` `formula  = term , { [ '+' \| '-' ] , term } ;` `term     = power , { [ '*' \| '/' \| '%' ] , power } ;` `power    = unary [ '^' , unary ] ;` `unary    = [ '+' \| '-' ] , factor ;` `factor   = variable \| constant \| function \| parens ;` `parens   = '(' , formula , ')' ;` [*Note:* Formulas can only support a calcMode (Calculation Mode) of linear or discrete. If another calcMode is specified or no calcMode is specified then a calcMode of linear is assumed. *end note*] [*Note:* Any additional characters in the formula string that are not contained within the set described are considered invalid. *end note*] Variables: <table><tr><th>Name</th><th>Description</th></tr><tr><td>$</td><td>Formula input</td></tr><tr><td>ppt_x</td><td>Pre-animation x position of the object on the slide</td></tr><tr><td>ppt_y</td><td>Pre-animation y position of the object on the slide</td></tr><tr><td>ppt_w</td><td>Pre-animation width of the object</td></tr><tr><td>ppt_h</td><td>Pre-animation height of the object</td></tr></table> Constants: <table><tr><th>Name</th><th>Description</th></tr><tr><td>pi</td><td>Mathematical constant pi</td></tr><tr><td>e</td><td>Mathematical constant e</td></tr></table> Operators: <table><tr><th>Name</th><th>Description</th><th>Usage</th></tr><tr><td>+</td><td>Addition</td><td>“x+y”, adds x to the value y</td></tr><tr><td>-</td><td>Subtraction</td><td>“x-y”, subtracts y from the value x</td></tr><tr><td>*</td><td>Multiplication</td><td>“x*y”, multiplies x by the value y</td></tr></table>

Attributes	Description

/	Division	“x/y”, divides x by the value y
%	Modulus	“x%y”, the remainder of x/y
^	Power	“x^y”, x raised to the power y

Functions:

Name	Description	Usage
abs	Absolute value	“abs(x)”, absolute value of x
acos	Arc Cosine	“acos(x)”, arc cosine of the value x
asin	Arc Sine	“asin(x)”, arc sine of the value x
atan	Arc Tangent	“atan(x)”, arc tangent of the value x
ceil	Ceil value	“ceil(x)”, value of x rounded up
cos	Cosine	“cos(x)”, cosine of the value of x
cosh	Hyperbolic Cosine	“cosh(x)", hyperbolic cosine of the value x
deg	Radiant to Degree convert	“deg(x)”, the degree value of radiant value x
exp	Exponent	“exp(x)”, value of constant e raised to the power of x
floor	Floor value	“floor(x)”, value of x rounded down
ln	Natural logarithm	“ln(x)”, natural logarithm of x
max	Maximum of two values	“max(x,y)”, returns x if (x > y) or returns y if (y > x)
min	Minimum of two values	“min(x,y)", returns x if (x < y) or returns y if (y < x)
rad	Degree to Radiant convert	“rad(x)”, the radiant value of degree value x
rand	Random value	“rand(x)”, returns a random floating point value between 0 and x
sin	Sine	“sin(x)”, sine of the value x
sinh	Hyperbolic Sine	"sinh(x)”, hyperbolic sine of the value x
sqrt	Square root	“sqrt(x)”, square root of the value x
tan	Tangent	“tan(x)”, tangent of the value x
tanh	Hyperbolic Tangent	“tanh(x)", hyperbolic tangent of the value x

[*Example:* Consider the following animation path:

<table>
<tr><th>Attributes</th><th>Description</th></tr>
<tr><td></td><td>
<pre>
<p:animcalcmode="lin" valueType="num">
  <p:cBhvr>
    <p:cTn id="9" dur="664" tmFilter="0.0,0.0; 0.25,0.07;
0.50,0.2; 0.75,0.467; 1.0,1.0">
      <p:stCondLst>
        <p:cond delay="0"/>
      </p:stCondLst>
    </p:cTn>
    <p:tgtEl>
      <p:spTgtspid="4"/>
    </p:tgtEl>
    <p:attrNameLst>
      <p:attrName>ppt_y</p:attrName>
    </p:attrNameLst>
  </p:cBhvr>
  <p:tavLst>
    <p:tav tm="0%" fmla="#ppt_y-sin(pi*$)/3">
      <p:val>
        <p:fltValval="0.5"/>
      </p:val>
    </p:tav>
    <p:tav tm="100%">
      <p:val>
        <p:fltValval="1"/>
      </p:val>
    </p:tav>
  </p:tavLst>
</p:anim>
</pre>
The animation example above modifies the ppt_y variable of the object by subtracting sin(pi*$)/3 from the non-animated value of ppt_y. The start value is 0.5 and the end value is 1 specified in each of the val elements. The total time for this animation is specified within the dur attribute and the filtered time graph is specified by the tmFilter attribute. The end result is that the object moves from a point above its non-animated position back to its non-animated position. With the specification of the tmFilter it has a modified time graph such that it also appears to accelerate as it reaches its final position.
[Note: For this example, the non-animated value of ppt_y is the value of this variable if the object were to be statically rendered on the slide without animation properties. end note]
end example]
The possible values for this attribute are defined by the W3C XML Schema string datatype.
</td></tr>
<tr><td>tm (Time)</td><td>This attribute specifies the time at which the attribute being animated takes on the value.</td></tr>
</table>

Attributes	Description
	The possible values for this attribute are defined by the ST_TLTimeAnimateValueTime simple type (§19.7.39).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeAnimateValue) is located in §A.3. *end note*]

19.5.80 tavLst (Time Animated Value List)

This element specifies a list of time animated value elements.

[*Example*: Consider a shape with a "fly-in" animation. The <tav> element should be used as follows:

```
<p:anim calcmode="lin" valueType="num">
  <p:cBhvr additive="base"> … </p:cBhvr>
  <p:tavLst>
    <p:tav tm="0%">
      <p:val>
        <p:strVal val="1+#ppt_h/2"/>
      </p:val>
    </p:tav>
    <p:tav tm="100000">
      <p:val>
        <p:strVal val="#ppt_y"/>
      </p:val>
    </p:tav>
  </p:tavLst>
</p:anim>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeAnimateValueList) is located in §A.3. *end note*]

19.5.81 tgtEl (Target Element)

This element specifies the target children elements which have the animation effects applied to.

[*Example*: Consider a shape with ID 3 with a fade effect animation applied to it. The <tgtEl> element should be used as follows:

```
<p:animEffect transition="in" filter="fade">
  <p:cBhvr>
    <p:cTn id="7" dur="2000"/>
```



```
        <p:tgtEl>
          <p:spTgt spid="3"/>
        </p:tgtEl>
      </p:cBhvr>
    </p:animEffect>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTimeTargetElement) is located in §A.3. *end note*]

19.5.82 tmAbs (Time Absolute)

This element describes the duration of the iteration interval in absolute time.

[*Example*: Consider a text animation where the words appear letter by letter every 10 seconds. The <tmAbs> element should be used as follows:

```
<p:par>
  <p:cTn id="5" >
    <p:stCondLst> … </p:stCondLst>
    <p:iterate type="lt">
      <p:tmAbs val="10000"/>
    </p:iterate>
    <p:childTnLst> … </p:childTnLst>
  </p:cTn>
</p:par>
```

end example]

Attributes	Description
val (Time)	This attribute describes an amount of time, in milliseconds. The possible values for this attribute are defined by the ST_TLTime simple type (§19.7.38).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLIterateIntervalTime) is located in §A.3. *end note*]

19.5.83 tmPct (Time Percentage)

This element describes the duration of the iteration interval in a percentage of time.

[*Example*: Consider a text animation where the words appear letter by letter every 10th of the animation duration. The <tmPct> element should be used as follows:

```
<p:par>
  <p:cTn id="5" >
    <p:stCondLst> … </p:stCondLst>
    <p:iterate type="lt">
      <p:tmPct val="10%"/>
    </p:iterate>
    <p:childTnLst> … </p:childTnLst>
  </p:cTn>
</p:par>
```

end example]

Attributes	Description
val (Value)	This attribute specifies the time expressed as a percentage. The possible values for this attribute are defined by the ST_PositivePercentage simple type (§20.1.10.45).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLIterateIntervalPercentage) is located in §A.3. *end note*]

19.5.84 tmpl (Template Effects)

This element specifies the "template" effects that are used by the build element. Template effects are used in text builds on the master slide. They define the rules of what effect should be applied to the 1st level paragraph, 2nd level paragraph, etc.

[*Example*: Consider a template with a fade in effect applied to it. The <tmpl> element should be used as follows:

```
<p:timing>
  <p:tnLst> … </p:tnLst>
  <p:bldLst>
    <p:bldP spid="3" grpId="0" build="p">
      <p:tmplLst>
        <p:tmpl lvl="1">
        </p:tmpl>
      </p:tmplLst>
    </p:bldP>
  </p:bldLst>
</p:timing>
```

end example]

Attributes	Description
lvl (Level)	This attribute describes the paragraph indent level to which this template effect applies. The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTemplate) is located in §A.3. *end note*]

19.5.85 tmplLst (Template effects)

This element describes a list of template effects that describe what kind of effects should be applied to a paragraph level properties.

[*Example*: Consider a template with a fade in effect applied to it. The <tmpl> element should be used as follows:

```
<p:timing>
  <p:tnLst> … </p:tnLst>
  <p:bldLst>
    <p:bldP spid="3" grpId="0" build="p">
      <p:tmplLst>
        <p:tmpl lvl="1">
        </p:tmpl>
      </p:tmplLst>
    </p:bldP>
  </p:bldLst>
</p:timing>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTemplateList) is located in §A.3. *end note*]

19.5.86 tn (Time Node)

This element describes the time node trigger choice.

[*Example*: Consider a time node with an event condition. The <tn> element should be used as follows:

```
<p:par>
  <p:cTn id="5">
    <p:stCondLst>
      <p:cond delay="0"/>
    </p:stCondLst>
    <p:endCondLst>
      <p:cond evt="begin" delay="0">
```



```
        <p:tn val="5"/>
      </p:cond>
    </p:endCondLst>
    <p:childTnLst> … </p:childTnLst>
  </p:cTn>
</p:par>
```

end example]

Attributes	Description
val (Value)	This attribute specifies a time node identifier. The possible values for this attribute are defined by the ST_TLTimeNodeID simple type (§19.7.42).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTriggerTimeNodeID) is located in §A.3. *end note*]

19.5.87 tnLst (Time Node List)

This element specifies a list of time node elements used in an animation sequence.

[*Example*: Consider a simple animation sequence. The <tnLst> element should be used as follows:

```
<p:timing>
  <p:tnLst>
    <p:par> … </p:par>
  </p:tnLst>
</p:timing>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TimeNodeList) is located in §A.3. *end note*]

19.5.88 to (To)

This element specifies the target location for an animation motion or animation scale effect

[*Example*: Consider an animation with a "light speed" entrance effect.

```
<p:animScale>
  <p:cBhvr>
    <p:cTn id="9" dur="200" decel="10.5%" autoRev="1" fill="hold">
      <p:stCondLst>
        <p:cond delay="600"/>
```



```
        </p:stCondLst>
      </p:cTn>
      <p:tgtEl>
        <p:spTgt spid="4"/>
      </p:tgtEl>
    </p:cBhvr>
    <p:from x="100%" y="100%"/>
    <p:to x="80%" y="100%"/>
  </p:animScale>
```

end example]

Attributes	Description
x (X coordinate)	This attribute describes the X coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§20.1.10.40).
y (Y coordinate)	This attribute describes the Y coordinate. The possible values for this attribute are defined by the ST_Percentage simple type (§20.1.10.40).

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLPoint) is located in §A.3. *end note*]

19.5.89 to (To)

The element specifies the certain attribute of a time node after an animation effect.

[*Example*: Consider an animation effect that leaves a string value visible afterwards. The <to> element should be used as follows:

```
  <p:childTnLst>
    <p:set>
      <p:cBhvr> … </p:cBhvr>
      <p:to>
        <p:strVal val="visible"/>
      </p:to>
    </p:set>
    <p:anim calcmode="lin" valueType="num"> … </p:anim>
  …</p:childTnLst>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimVariant) is located in §A.3. *end note*]

19.5.90 to (To)

This element specifies the resulting color for the animation color change.

[*Example*: Consider emphasize a shape by changing its fill color from blue to red. The <to> element should be used as follows:

```
<p:childTnLst>
  <p:animClr clrSpc="rgb">
    <p:cBhvr> … </p:cBhvr>
    <p:to>
      <a:schemeClr val="accent2"/>
    </p:to>
  </p:animClr>
</p:childTnLst>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_Color) is located in §A.4.1. *end note*]

19.5.91 txEl (Text Element)

This element specifies a text element to animate.

[*Example*: Consider a shape containing text to be animated. The <txEl> should be used as follows:

```
<p:tgtEl>
  <p:spTgt spid="5">
    <p:txEl>
      <p:pRg st="1" end="1"/>
    </p:txEl>
  </p:spTgt>
</p:tgtEl>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLTextTargetElement) is located in §A.3. *end note*]

19.5.92 val (Value)

The element specifies a value for a time animate.

[*Example*: Consider a shape with a fade in animation effect. The <val> element should be used as follows:


```
<p:anim calcmode="lin" valueType="num">
  <p:cBhvr additive="base"> … </p:cBhvr>
  <p:tavLst>
    <p:tav tm="0%">
      <p:val>
        <p:strVal val="0-#ppt_w/2"/>
      </p:val>
    </p:tav>
    <p:tav tm="100%">
      <p:val>
        <p:strVal val="#ppt_x"/>
      </p:val>
    </p:tav>
  </p:tavLst>
</p:anim>
```

end example]

[*Note*: The W3C XML Schema definition of this element's content model (CT_TLAnimVariant) is located in §A.3. *end note*]

19.5.93 video (Video)

This element specifies video information in an animation sequence. This element specifies that this node within the animation tree triggers the playback of a video file; the actual video file used is specified by the videoFile element (§20.1.3.6).

[*Example*: Consider a slide with an animated video content. The <video> element is used as follows:

```
<p:cSld>
  <p:spTree>
    <p:pic>
      <p:nvPicPr>
        <p:cNvPr id="4"/>
        …
        <p:nvPr>
          <a:videoFile r:link="rId1" contentType="video/ogg"/>
        </p:nvPr>
      </p:nvPicPr>
      …
    </p:pic>
  </p:spTree>
</p:cSld>
…
<p:childTnLst>
```



```
    <p:seq concurrent="1" nextAc="seek">
     …
    </p:seq>
    <p:video>
      <p:cMediaNode>
        …
        <p:tgtEl>
          <p:spTgt spid="4"/>
        </p:tgtEl>
      </p:cMediaNode>
    </p:video>
  </p:childTnLst>
```

The video element specifies the location of the video playback within the animation sequence; its child spTgt element specifies that the shape which contains the video to be played has a shape ID of 4. If we look at the shape with that ID value, its child videoFile element references an external video file of content type `video/ogg` located at the target of the relationship with ID `rId1`.*end example*]

Attributes	Description
fullScrn (Full Screen)	This attribute specifies if the video is displayed in full-screen. The possible values for this attribute are defined by the W3C XML Schema boolean datatype.

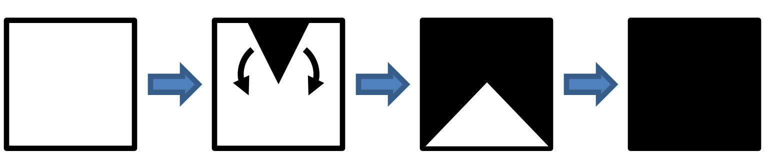
[*Note*: The W3C XML Schema definition of this element's content model (CT_TLMediaNodeVideo) is located in §A.3. *end note*]

19.5.94 wedge (Wedge Slide Transition)

This element describes the wedge slide transition effect, which uses two radial edges that wipe from top to bottom in opposite directions until the new slide is fully shown. The rendering of this transition has been shown below.

[*Example*: Consider the following case in which the "wedge" slide transition is applied to a slide, along with a set of attributes. The proper usage and a sample rendering are shown below, with the XML fragments preceding the corresponding rendering:

```
  <p:transition>
    <p:wedge/>
  </p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

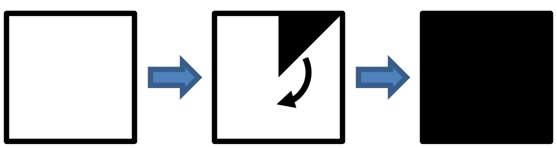
[*Note*: The W3C XML Schema definition of this element's content model (CT_Empty) is located in §A.3. *end note*]

19.5.95 wheel (Wheel Slide Transition)

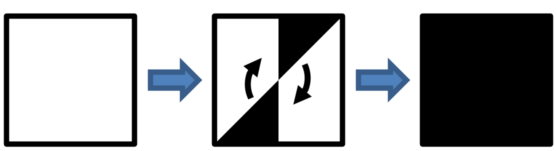
This element describes the wheel slide transition effect, which uses a set of radial edges and wipes them in the clockwise direction until the new slide is fully shown. The rendering of this transition depends upon the attributes specified which have been shown below.

[*Example*: Consider the following cases in which the “wheel” slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

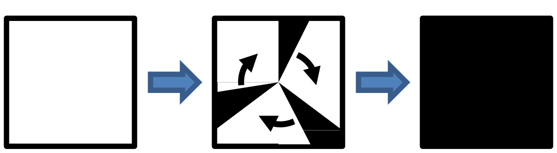
```
<p:transition>
  <p:wheel spokes="1"/>
</p:transition>
```



```
<p:transition>
  <p:wheel spokes="2"/>
</p:transition>
```



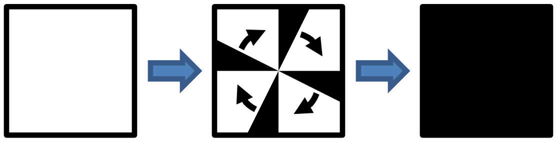
```
<p:transition>
  <p:wheel spokes="3"/>
</p:transition>
```



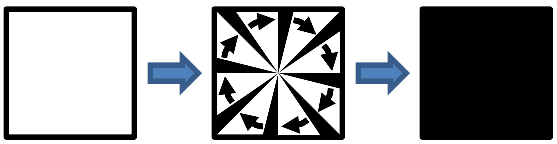
```
<p:transition>
```



```
    <p:wheel spokes="4"/>
  </p:transition>
```



```
  <p:transition>
    <p:wheel spokes="8"/>
  </p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
spokes (Spokes)	This attributes specifies the number of spokes ("pie pieces") in the wheel The possible values for this attribute are defined by the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_WheelTransition) is located in §A.3. *end note*]

19.5.96 wipe (Wipe Slide Transition)

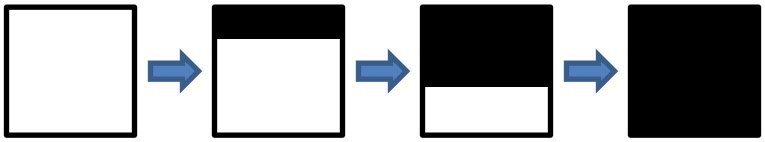
This element describes the wipe slide transition effect, which wipes the new slide over the previous slide from one edge of the screen to the opposite until the new slide is fully shown. The rendering of this transition depends upon the attributes specified which have been shown below.

[*Example*: Consider the following cases in which the "wipe" slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

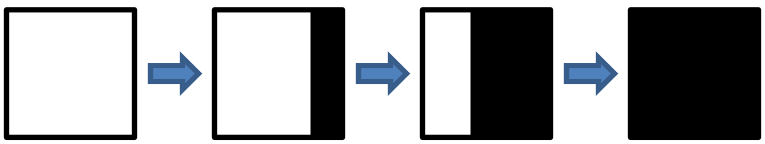
```
  <p:transition>
    <p:wipe dir="d"/>
```



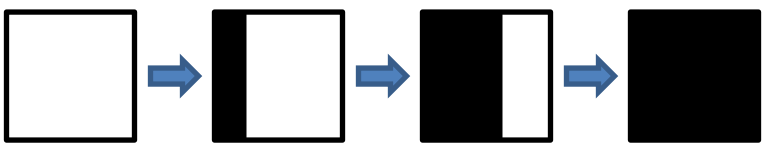
```
</p:transition>
```



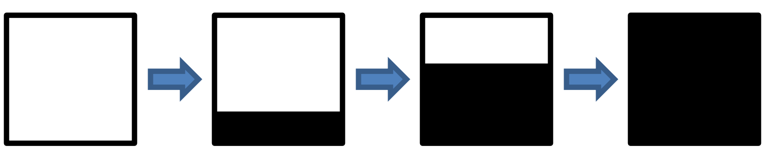
```
<p:transition>
  <p:wipe dir="l"/>
</p:transition>
```



```
<p:transition>
  <p:wipe dir="r"/>
</p:transition>
```



```
<p:transition>
  <p:wipe dir="u"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
dir (Direction)	This attribute specifies the direction of the slide transition. The possible values for this attribute are defined by the ST_TransitionSideDirectionType simple type (§19.7.53).

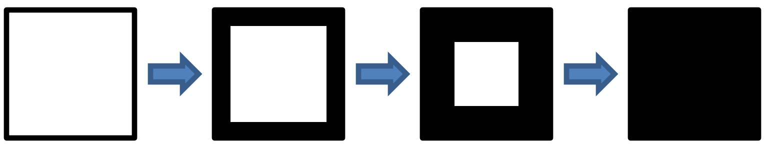
[*Note*: The W3C XML Schema definition of this element's content model (CT_SideDirectionTransition) is located in §A.3. *end note*]

19.5.97 zoom (Zoom Slide Transition)

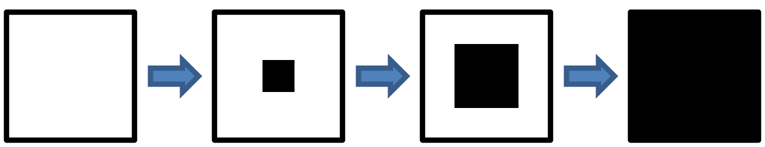
This element describes the zoom slide transition effect, which uses a box pattern centered on the slide that increases in size until the new slide is fully shown. The rendering of this transition depends upon the attributes specified which have been shown below.

[*Example*: Consider the following cases in which the “zoom” slide transition is applied to a slide, along with a set of attributes. The proper usage and sample renderings are shown below, with the XML fragments preceding the corresponding rendering:

```
<p:transition>
  <p:zoom dir="in"/>
</p:transition>
```



```
<p:transition>
  <p:zoom dir="out"/>
</p:transition>
```



end example]

[*Note*: Any rendering shown above is for example purposes only. Exact rendering of any transition is determined by the rendering application. As such, the same transition can have many variations depending on the implementation. *end note*]

Attributes	Description
dir (Direction)	This attribute specifies the direction of an "in/out" slide transition. The possible values for this attribute are defined by the ST_TransitionInOutDirectionType simple type (§19.7.52).

[*Note*: The W3C XML Schema definition of this element’s content model (CT_InOutTransition) is located in §A.3. *end note*]

19.6 Slide Synchronization Data

It is often the case that slides are repurposed from existing presentations to be used in other presentations. In such cases, it is often beneficial for there to be an association, or a pairing, between the original slide and all copied instances of it. In the presence of such a pairing, applications can enable a variety of time-saving features, including the automatic updates of copied slides when the original slide changes. The Slide Synchronization Data part is designed to enable such application-defined functionality.

This information is stored in the Slide Synchronization Data part, which is referenced via an implicit relationship from the associated Slide part.

19.6.1 sldSyncPr (Slide Synchronization Properties)

This element specifies the information needed to associate the original slide with all copied instances of it.

Attributes	Description
clientInsertedTime (Client Slide Insertion date/time)	The date and time that the original slide was last updated in the current presentation. The date/time is stored in ISO 8601 format. [*Note*: This value can be used to inform the user of when the last synchronization was, as well as to determine when to next check for an updated version. *end note*] The possible values for this attribute are defined by the W3C XML Schema dateTime datatype.
serverSldId (Server's Slide File ID)	A string that, when paired with the target of the Slide Synchronization Data part's external relationship, uniquely identifies the original slide. The possible values for this attribute are defined by the W3C XML Schema string datatype.
serverSldModifiedTime (Server's Slide File's modification date/time)	The date and time that the original slide was last modified in its location as defined by the target of the Slide Synchronization Data part's external relationship. The date and time are stored in ISO 8601 format. The possible values for this attribute are defined by the W3C XML Schema dateTime datatype.

[*Note*: The W3C XML Schema definition of this element's content model (CT_SlideSyncProperties) is located in §A.3. *end note*]

19.7 Simple Types

This is the complete list of simple types dedicated to PresentationML.

19.7.1 ST_BookmarkIdSeed (Bookmark ID Seed)

This simple type specifies constraints for value of the Bookmark ID seed.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 1.
- This simple type has a maximum value of less than 2147483648.

[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_BookmarkIdSeed) is located in §A.3. *end note*]

19.7.2 ST_Direction (Direction)

This simple type defines a direction of either horizontal or vertical.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
horz (Horizontal)	Defines a horizontal direction.
vert (Vertical)	Defines a vertical direction.

[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_Direction) is located in §A.3. *end note*]

19.7.3 ST_Index (Index)

This simple type defines the position of an object in an ordered list.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this simple type’s content model (ST_Index) is located in §A.3. *end note*]

19.7.4 ST_IterateType (Iterate Type)

This simple type specifies how the animation is applied over subelements of the target element.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
el (Element)	Iterate by element.

Enumeration Value	Description
lt (Letter)	Iterate by Letter.
wd (Word)	Iterate by Word.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_IterateType) is located in §A.3. *end note*]

19.7.5 ST_Name (Name string)

This simple type specifies a name, such as for a comment author or custom show.

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_Name) is located in §A.3. *end note*]

19.7.6 ST_OleObjectFollowColorScheme (Embedded object to Follow Color Scheme)

This simple type determines if the Embedded object is re-colored to reflect changes to the color schemes.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
full (Full)	Setting this enumeration causes the Embedded object to respond to all changes in the color scheme in the presentation.
none (None)	Setting this enumeration causes the Embedded object to not respond to changes in the color scheme in the presentation.
textAndBackground (Text and Background)	Setting this enumeration causes the Embedded object to respond only to changes in the text and background colors of the color scheme in the presentation.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_OleObjectFollowColorScheme) is located in §A.3. *end note*]

19.7.7 ST_PhotoAlbumFrameShape (Photo Album Shape for Photo Mask)

This simple type specifies the values for photo frame types within a photo album presentation.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
frameStyle1 (Rectangle Photo Frame)	
frameStyle2 (Rounded Rectangle Photo Frame)	
frameStyle3 (Simple White Photo Frame)	
frameStyle4 (Simple Black Photo Frame)	
frameStyle5 (Compound Black Photo Frame)	
frameStyle6 (Center Shadow Photo Frame)	
frameStyle7 (Soft Edge Photo Frame)	

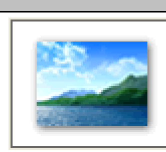
[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PhotoAlbumFrameShape) is located in §A.3. *end note*]

19.7.8 ST_PhotoAlbumLayout (Photo Album Layout Definition)

This simple type specifies the values for photo layouts within a photo album presentation.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
1pic (1 Photo per Slide)	

Enumeration Value	Description
	Specifies that photo album slides should have a single picture, centered horizontally and vertically, on the slide with no title.
1picTitle (1 Photo per Slide with Titles)	 Specifies that photo album slides should have a single picture and a single title text box, centered horizontally and vertically, on the slide.
2pic (2 Photos per Slide)	 Specifies that photo album slides should have two pictures of the same size, positioned side-by-side, centered horizontally and vertically, on the slide with no title.
2picTitle (2 Photos per Slide with Titles)	 Specifies that photo album slides should have two pictures of the same size, positioned side-by-side, with a single title text box centered over them, collectively centered horizontally and vertically, on the slide.
4pic (4 Photos per Slide)	 Specifies that photo album slides should have four pictures of the same size, positioned in a two-by-two matrix, centered horizontally and vertically, on the slide with no title.
4picTitle (4 Photos per Slide with Titles)	 Specifies that photo album slides should have four pictures of the same size, positioned in a two-by-two matrix, with a single title text box centered over the matrix, centered horizontally and vertically, on the slide.

Enumeration Value	Description
fitToSlide (Fit Photos to Slide)	 Specifies that photo album slides should have a single picture, stretched to fit the entire slide size, with no title.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PhotoAlbumLayout) is located in §A.3. *end note*]

19.7.9 ST_PlaceholderSize (Placeholder Size)

This simple type facilitates the storing of the size of the placeholder. This size is described relative to the body placeholder on the master.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
full (Full)	Specifies that the placeholder should take the full size of the body placeholder on the master.
half (Half)	Specifies that the placeholder should take the half size of the body placeholder on the master. Half size vertically or horizontally? Needs a picture.
quarter (Quarter)	Specifies that the placeholder should take a quarter of the size of the body placeholder on the master. Picture would be helpful

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PlaceholderSize) is located in §A.3. *end note*]

19.7.10 ST_PlaceholderType (Placeholder IDs)

This simple type facilitates the storing of the content type a placeholder should contain.

[*Note*: Some placeholder types are not allowed for all SlideBase types. *end note*]

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
body (Body)	Contains body text. Allowed for Slide, Slide Layout,

Enumeration Value	Description
	Slide Master, Notes, Notes Master. Can be horizontal or vertical on Slide and Slide Layout.
chart (Chart)	Contains a chart or graph. Special type. Allowed for Slide and Slide Layout.
clipArt (Clip Art)	Contains a single clip art image. Special type. Allowed for Slide and Slide Layout.
ctrTitle (Centered Title)	Contains a title intended to be centered on the slide. Allowed for Slide and Slide Layout.
dgm (Diagram)	Contains a diagram. Special type. Allowed for Slide and Slide Layout.
dt (Date and Time)	Contains the date and time. Allowed for Slide, Slide Layout, Slide Master, Notes, Notes Master, Handout Master
ftr (Footer)	Contains text to be used as a footer in the document. Allowed for Slide, Slide Layout, Slide Master, Notes, Notes Master, Handout Master
hdr (Header)	Contains text to be used as a header for the document. Allowed for Notes, Notes Master, Handout Master .
media (Media)	Contains multimedia content such as audio or a movie clip. Special type. Allowed for Slide and Slide Layout.
obj (Object)	Contains any content type. Special type. Allowed for Slide and Slide Layout.
pic (Picture)	Contains a picture. Special type. Allowed for Slide and Slide Layout.
sldImg (Slide Image)	Contains an image of the slide. Allowed for Notes and Notes Master.
sldNum (Slide Number)	Contains the number of a slide. Allowed for Slide, Slide Layout, Slide Master, Notes, Notes Master, Handout Master
subTitle (Subtitle)	Contains a subtitle. Allowed for Slide and Slide Layout.
tbl (Table)	Contains a table. Special type. Allowed for Slide and Slide Layout.
title (Title)	Contains a slide title. Allowed for Slide, Slide Layout and Slide Master. Can be horizontal or vertical on Slide and Slide Layout.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PlaceholderType) is located in §A.3. *end note*]

19.7.11 ST_PrintColorMode (Print Color Mode)

This simple type specifies the color mode that should be used when printing a presentation document.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
bw (Black and White Mode)	Print should be in Black and White only
clr (Color Mode)	Print should be in Full Color
gray (Grayscale Mode)	Print should be in Grayscale only

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PrintColorMode) is located in §A.3. *end note*]

19.7.12 ST_PrintWhat (Default print output)

This simple type specifies the default print layout that should be used when printing

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
handouts1 (1 Slide / Handout Page)	1 Slide and Handout Page layout should be used.
handouts2 (2 Slides / Handout Page)	2 Slides and Handout Page layout should be used.
handouts3 (3 Slides / Handout Page)	3 Slides and Handout Page layout should be used.
handouts4 (4 Slides / Handout Page)	4 Slides and Handout Page layout should be used.
handouts6 (6 Slides / Handout Page)	6 Slides and Handout Page layout should be used.
handouts9 (9 Slides / Handout Page)	9 Slides and Handout Page layout should be used.
notes (Notes)	Notes layout should be used.
outline (Outline)	Outline layout should be used.
slides (Slides)	Slides layout should be used.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_PrintWhat) is located in §A.3. *end note*]

19.7.13 ST_SlideId (Slide Identifier)

This simple type specifies the allowed numbering for the slide identifier.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

This simple type also specifies the following restrictions:

- This simple type has a maximum value of less than 2147483648.
- This simple type has a minimum value of greater than or equal to 256.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SlideId) is located in §A.3. *end note*]

19.7.14 ST_SlideLayoutId (Slide Layout ID)

This simple type sets the bounds for the slide layout id value. This layout id is used to identify the different slide layout designs.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 2147483648.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SlideLayoutId) is located in §A.3. *end note*]

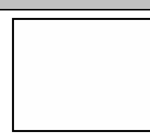
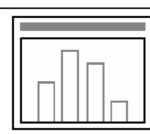
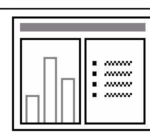
19.7.15 ST_SlideLayoutType (Slide Layout Type)

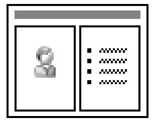
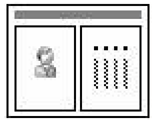
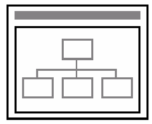
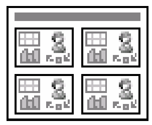
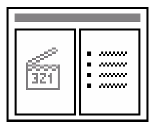
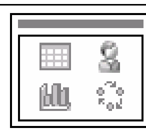
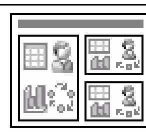
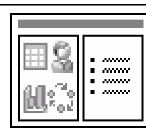
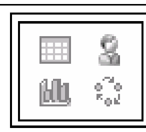
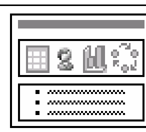
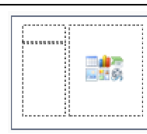
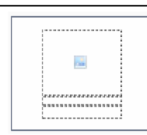
This simple type defines an arrangement of content on a slide. Each layout type is not tied to an exact positioning of placeholders, but rather provides a higher-level description of the content type and positioning of placeholders. This information can be used by the application to aid in mapping between different layouts. The application can choose which, if any, of these layouts to make available through its user interface.

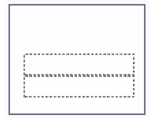
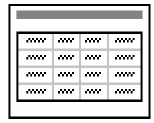
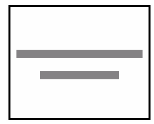
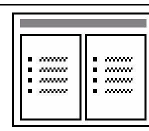
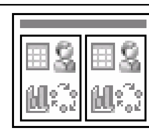
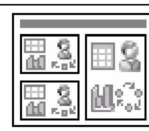
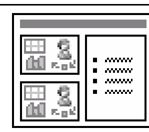
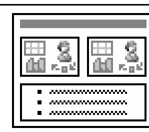
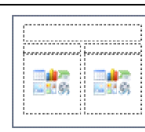
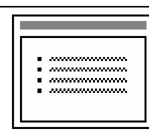
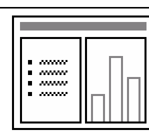
Each layout contains zero or more placeholders, each with a specific content type. An "object" placeholder can contain any kind of data. Media placeholders are intended to hold video or audio clips. The enumeration value descriptions include illustrations of sample layouts for each value of the simple type.

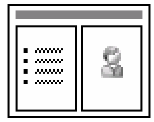
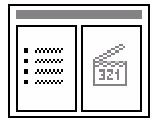
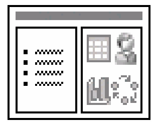
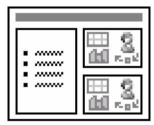
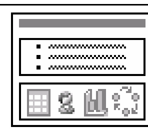
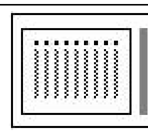
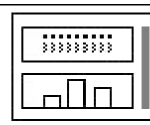
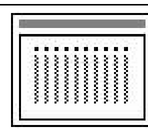
This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
blank (Slide Layout Type Enumeration (Blank))	 Blank
chart (Chart)	 Title and chart
chartAndTx (Slide Layout Type Enumeration (Chart and Text))	 Title, chart on left and text on right

Enumeration Value	Description
clipArtAndTx (Clip Art and Text)	 Title, clipart on left, text on right
clipArtAndVertTx (Clip Art and Vertical Text)	 Title, clip art on left, vertical text on right
cust (Slide Layout Type Enumeration (Custom))	Custom layout defined by user
dgm (Slide Layout Type Enumeration (Diagram))	 Title and diagram
fourObj (Four Objects)	 Title and four objects
mediaAndTx (Slide Layout Type Enumeration (Media and Text))	 Title, media on left, text on right
obj (Title and Object)	 Title and object
objAndTwoObj (Object and Two Object)	 Title, one object on left, two objects on right
objAndTx (Slide Layout Type Enumeration (Object and Text))	 Title, object on left, text on right
objOnly (Object)	 Object only
objOverTx (Slide Layout Type Enumeration (Object over Text))	 Title, object on top, text on bottom
objTx (Title, Object, and Caption)	 Title, object and caption text
picTx (Picture and Caption)	 Title, picture, and caption text
secHead (Section Header)	Section header title and subtitle text

Enumeration Value	Description
	
tbl (Slide Layout Type Enumeration (Table))	 Title and table
title (Slide Layout Type Enumeration (Title))	 Title layout with centered title and subtitle placeholders
titleOnly (Slide Layout Type Enumeration (Title Only))	 Title only
twoColTx (Slide Layout Type Enumeration (Two Column Text))	 Title, text on left, text on right
twoObj (Two Objects)	 Title, object on left, object on right
twoObjAndObj (Two Objects and Object)	 Title, two objects on left, one object on right
twoObjAndTx (Two Objects and Text)	 Title, two objects on left, text on right
twoObjOverTx (Two Objects over Text)	 Title, two objects on top, text on bottom
twoTxTwoObj (Two Text and Two Objects)	 Title, two objects each with text
tx (Slide Layout Type Enumeration (Text))	 Title and text
txAndChart (Slide Layout Type Enumeration (Text and Chart))	 Title, text on left and chart on right

Enumeration Value	Description
txAndClipArt (Text and Clip Art)	 Title, text on left, clip art on right
txAndMedia (Slide Layout Type Enumeration (Text and Media))	 Title, text on left, media on right
txAndObj (Slide Layout Type Enumeration (Text and Object))	 Title, text on left, object on right
txAndTwoObj (Text and Two Objects)	 Title, text on left, two objects on right
txOverObj (Slide Layout Type Enumeration (Text over Object))	 Title, text on top, object on bottom
vertTitleAndTx (Vertical Title and Text)	 Vertical title on right, vertical text on left
vertTitleAndTxOverChart (Vertical Title and Text Over Chart)	 Vertical title on right, vertical text on top, chart on bottom
vertTx (Vertical Text)	 Title and vertical text body

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SlideLayoutType) is located in §A.3. *end note*]

19.7.16 ST_SlideMasterId (Slide Master ID)

This simple type specifies the allowed numbering for the slide master identifier.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

This simple type also specifies the following restrictions:

- This simple type has a minimum value of greater than or equal to 2147483648.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SlideMasterId) is located in §A.3. *end note*]

19.7.17 ST_SlideSizeCoordinate (Slide Size Coordinate)

This simple type specifies the slide size coordinate in EMUs (English Metric Units).

This simple type's contents are a restriction of the ST_PositiveCoordinate32 datatype (§20.1.10.42).

This simple type also specifies the following restrictions:

- This simple type has a maximum value of less than or equal to 51206400.
- This simple type has a minimum value of greater than or equal to 914400.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SlideSizeCoordinate) is located in §A.3. *end note*]

19.7.18 ST_SlideSizeType (Slide Size Type)

This simple type specifies the kind of slide size that the slide should be optimized for.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
35mm (35mm Film)	Slide size should be optimized for 35mm film output
A3 (A3)	Slide size should be optimized for A3 output
A4 (A4)	Slide size should be optimized for A4 output
B4ISO (B4ISO)	Slide size should be optimized for B4ISO output
B4JIS (B4JIS)	Slide size should be optimized for B4JIS output
B5ISO (B5ISO)	Slide size should be optimized for B5ISO output
B5JIS (B5JIS)	Slide size should be optimized for B5JIS output
banner (Banner)	Slide size should be optimized for banner output
custom (Custom)	Slide size should be optimized for custom output
hagakiCard (Hagaki Card)	Slide size should be optimized for hagaki card output
ledger (Ledger)	Slide size should be optimized for ledger output
letter (Letter)	Slide size should be optimized for letter output
overhead (Overhead)	Slide size should be optimized for overhead output
screen16x10 (Screen 16x10)	Slide size should be optimized for 16x10 screen output
screen16x9 (Screen 16x9)	Slide size should be optimized for 16x9 screen output
screen4x3 (Screen 4x3)	Slide size should be optimized for 4x3 screen output

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SlideSizeType) is located in §A.3. *end note*]

19.7.19 ST_SplitterBarState (Splitter Bar State)

This simple type specifies the state that the splitter bar should be shown in. The splitter bar separates a primary and secondary region within a viewing area.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
maximized (Max)	The primary region occupies the greatest amount of the viewing area allowed by the application.
minimized (Min)	The primary region occupies the least amount of the viewing area allowed by the application.
restored (Restored)	The primary region has a specific intermediate size.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_SplitterBarState) is located in §A.3. *end note*]

19.7.20 ST_TLAnimateBehaviorCalcMode (Time List Animate Behavior Calculate Mode)

This simple type specifies how the animation flows from point to point.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
discrete (Calc Mode Enum (Discrete))	Descrete
fmla (Calc Mode Enum (Formula))	Formula
lin (Calc Mode Enum (Linear))	Linear

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLAnimateBehaviorCalcMode) is located in §A.3. *end note*]

19.7.21 ST_TLAnimateBehaviorValueType (Time List Animate Behavior Value Types)

This simple type specifies the type of property value.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
clr (Value Type Enum (Color))	Color
num (Value Type Enum (Number))	Number
str (Value Type Enum (String))	String

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLAnimateBehaviorValueType) is located in §A.3. *end note*]

19.7.22 ST_TLAnimateColorDirection (Time List Animate Color Direction)

This simple type specifies the direction in which to interpolate the animation (clockwise or counterclockwise).

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
ccw (Counter-Clockwise)	Counter-Clockwise
cw (Direction Enum (Clockwise))	Clockwise

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLAnimateColorDirection) is located in §A.3. *end note*]

19.7.23 ST_TLAnimateColorSpace (Time List Animate Color Space)

This simple type specifies the color space of the animation.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
hsl (Color Space Enum (HSL))	Hue, Saturation, Luminance
rgb (Color Space Enum (RGB))	Red, Green, Blue

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLAnimateColorSpace) is located in §A.3. *end note*]

19.7.24 ST_TLAnimateEffectTransition (Time List Animate Effect Transition)

This simple type specifies whether the effect is a transition in, transition out, or neither.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
in (Transition Enum (In))	In
none (Transition Enum (None))	None
out (Transition Enum (Out))	Out

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLAnimateEffectTransition) is located in §A.3. *end note*]

19.7.25 ST_TLAnimateMotionBehaviorOrigin (Time List Animate Motion Behavior Origin)

This simple type specifies what the origin of the motion path is relative to.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
layout (Origin Enum (Layout))	Layout
parent (Origin Enum (Parent))	Parent

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLAnimateMotionBehaviorOrigin) is located in §A.3. *end note*]

19.7.26 ST_TLAnimateMotionPathEditMode (Time List Animate Motion Path Edit Mode)

This simple type specifies how the motion path moves when the target element is moved.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
fixed (Path Edit Mode Enum (Fixed))	Fixed
relative (Path Edit Mode Enum (Relative))	Relative

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLAnimateMotionPathEditMode) is located in §A.3. *end note*]

19.7.27 ST_TLBehaviorAccumulateType (Behavior Accumulate Type)

This simple type makes a repeating animation build with each iteration when set to "always."

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
always (Accumulate Enum (Always))	Always
none (Accumulate Enum (None))	None

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLBehaviorAccumulateType) is located in §A.3. *end note*]

19.7.28 ST_TLBehaviorAdditiveType (Behavior Additive Type)

This simple type specifies how to apply the animation values to the original value for the property.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
base (Additive Enum (Base))	Base
mult (Additive Enum (Multiply))	Multiply
none (None)	None
repl (Additive Enum (Replace))	Replace
sum (Additive Enum (Sum))	Sum

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLBehaviorAdditiveType) is located in §A.3. *end note*]

19.7.29 ST_TLBehaviorOverrideType (Behavior Override Type)

This simple type specifies how a behavior should override values of the attribute being animated on the target element. The "childStyle" clears the attributes on the children contained inside the target element.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
childStyle (Override Enum (Child Style))	Child Style
normal (Override Enum (Normal))	Normal

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLBehaviorOverrideType) is located in §A.3. *end note*]

19.7.30 ST_TLBehaviorTransformType (Behavior Transform Type)

This simple type specifies how the behavior animates the target element.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
img (Image)	Image transform
pt (Point)	Point transform

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLBehaviorTransformType) is located in §A.3. *end note*]

19.7.31 ST_TLChartSubelementType (Chart Subelement Type)

This simple type defines an animation target element that is represented by a subelement of a chart.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
category (Category Axis)	Category
gridLegend (Grid Legend)	Background Element (Grid and Legend)
ptInCategory (Single Point in Category)	Category Element
ptInSeries (Single Point in Data Series)	Series Element
series (Data Series)	Series

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLChartSubelementType) is located in §A.3. *end note*]

19.7.32 ST_TLCommandType (Command Type)

This simple type specifies a command type.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
call (Command Type Enum (Call))	Call
evt (Command Type Enum (Event))	Event
verb (Command Type Enum (Verb))	Verb

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLCommandType) is located in §A.3. *end note*]

19.7.33 ST_TLDiagramBuildType (Diagram Build Types)

This simple type specifies the different diagram build types.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
allAtOnce (Diagram Build Type Enum (All At Once))	All At Once
breadthByLvl (Diagram Build Type Enum (Breadth By Level))	Breadth By Level
breadthByNode (Diagram Build Type Enum (Breadth By Node))	Breadth By Node
ccw (Diagram Build Type Enum (Counter-Clockwise))	Counter-Clockwise
ccwIn (Diagram Build Type Enum (Counter-Clockwise-In))	Counter-Clockwise-In
ccwOut (Diagram Build Type Enum (Counter-Clockwise-Out))	Counter-Clockwise-Out
cust (Diagram Build Type Enum (Custom))	Custom
cw (Diagram Build Type Enum (Clockwise))	Clockwise
cwIn (Diagram Build Type Enum (Clockwise-In))	Clockwise-In
cwOut (Diagram Build Type Enum (Clockwise-Out))	Clockwise-Out
depthByBranch (Diagram Build Type Enum (Depth By Branch))	Depth By Branch
depthByNode (Diagram Build Type Enum (Depth By Node))	Depth By Node
down (Diagram Build Type Enum (Down))	Down
inByRing (Diagram Build Type Enum (In-By-Ring))	In-By-Ring
outByRing (Diagram Build Type Enum (Out-By-Ring))	Out-By-Ring
up (Diagram Build Type Enum (Up))	Up
whole (Diagram Build Type Enum (Whole))	Whole

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLDiagramBuildType) is located in §A.3. *end note*]

19.7.34 ST_TLNextActionType (Next Action Type)

This simple type specifies what to do when going forward in a sequence. When the value is "seek," it seeks the current child element to its natural end time before advancing to the next element.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
none (Next Action Type Enum (None))	None
seek (Next Action Type Enum (Seek))	Seek

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLNextActionType) is located in §A.3. *end note*]

19.7.35 ST_TLOleChartBuildType (Embedded Chart Build Type)

This simple type describes how to build an embedded Chart.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
allAtOnce (Chart Build Type Enum (All At Once))	All At Once
category (Chart Build Type Enum (Category))	By Category
categoryEl (Chart Build Type Enum (Category Element))	By Category Element
series (Chart Build Type Enum (Series))	By Series
seriesEl (Chart Build Type Enum (Series Element))	By Series Element

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLOleChartBuildType) is located in §A.3. *end note*]

19.7.36 ST_TLParaBuildType (Paragraph Build Type)

This simple type describes how to build a paragraph.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
allAtOnce (All At Once)	Specifies to animate all paragraphs at once.

Enumeration Value	Description
cust (Custom)	Specifies the build has custom user settings.
p (Paragraph)	Specifies to animate paragraphs grouped by bullet level.
whole (Whole)	Specifies to animate the entire body of text as one block.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLParaBuildType) is located in §A.3. *end note*]

19.7.37 ST_TLPreviousActionType (Previous Action Type)

This simple type specifies what to do when going backwards in a sequence. When the value is "skipTimed," the sequence continues to go backwards until it reaches a sequence element that was defined to being only on a "next" event.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
none (Previous Action Type Enum (None))	None
skipTimed (Previous Action Type Enum (Skip Timed))	Skip Timed

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLPreviousActionType) is located in §A.3. *end note*]

19.7.38 ST_TLTime (Time)

This simple type specifies time after which to automatically advance the build to the next step. An amount of time, in milliseconds.

This simple type is a union of the following types:

- The ST_TLTimeIndefinite simple type (§19.7.40).
- The W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTime) is located in §A.3. *end note*]

19.7.39 ST_TLTimeAnimateValueTime (Animation Time)

This simple type specifies a percentage within the time span of the element. A value of `indefinite` means the attribute should be ignored.

This simple type is a union of the following types:

- The ST_PositiveFixedPercentage simple type (§20.1.10.44).
- The ST_TLTimeIndefinite simple type (§19.7.40).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTimeAnimateValueTime) is located in §A.3. *end note*]

19.7.40 ST_TLTimeIndefinite (Indefinite Time Declaration)

This simple type specifies a value that designates an "indefinite" amount time -- typically means this property is subordinate to other, defined properties.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
indefinite (Indefinite Type Enum)	Specifies Indefinite Time

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTimeIndefinite) is located in §A.3. *end note*]

19.7.41 ST_TLTimeNodeFillType (Time Node Fill Type)

This simple type specifies what modifications the effect leaves on the target element's properties when the effect ends.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
freeze (Freeze)	Freeze
hold (TimeNode Fill Type Enum (Hold))	Hold
remove (Remove)	Remove
transition (Transition)	Transition

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTimeNodeFillType) is located in §A.3. *end note*]

19.7.42 ST_TLTimeNodeID (Time Node ID)

This simple type represents a node or event on the timeline by its identifier.

This simple type's contents are a restriction of the W3C XML Schema unsignedInt datatype.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTimeNodeID) is located in §A.3. *end note*]

19.7.43 ST_TLTimeNodeMasterRelation (Time Node Master Relation)

This simple type specifies how the time node plays back relative to its master time node.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
lastClick (TimeNode Master Relation Enum (Last Click))	Last Click
nextClick (TimeNode Master Relation Enum (Next Click))	Next Click
sameClick (TimeNode Master Relation Enum (Same Click))	Same Click

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTimeNodeMasterRelation) is located in §A.3. *end note*]

19.7.44 ST_TLTimeNodePresetClassType (Time Node Preset Class Type)

This simple type specifies the class of effect in which this effect belongs.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
emph (Preset Type Enum (Emphasis))	Emphasis Preset
entr (Preset Type Enum (Entrance))	Entrance Preset
exit (Exit)	Exit Preset
mediacall (Preset Type Enum (Media Call))	Media Call Preset
path (Preset Type Enum (Path))	Path Preset
verb (Preset Type Enum (Verb))	Verb Preset

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTimeNodePresetClassType) is located in §A.3. *end note*]

19.7.45 ST_TLTimeNodeRestartType (Time Node Restart Type)

This simple type determines whether an effect can play more than once.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
always (Restart Enum (Always))	Always restart node
never (Restart Enum (Never))	Never restart node
whenNotActive (Restart Enum (When Not Active))	Restart when node is not active

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTimeNodeRestartType) is located in §A.3. *end note*]

19.7.46 ST_TLTimeNodeSyncType (Time Node Sync Type)

This simple type specifies how the time node synchronizes to its group.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
canSlip (TimeNode Sync Enum (Can Slip))	Can Slip
locked (TimeNode Sync Enum (Locked))	Locked

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTimeNodeSyncType) is located in §A.3. *end note*]

19.7.47 ST_TLTimeNodeType (Time Node Type)

This simple type specifies time node types.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
afterEffect (Node Type Enum (After Effect))	After Effect
afterGroup (Node Type Enum (After Group))	After Group
clickEffect (Node Type Enum (Click Effect))	Click Effect
clickPar (Node Type Enum (Click Paragraph))	Click Paragraph
interactiveSeq (Node Type Enum (Interactive Sequence))	Interactive Sequence
mainSeq (Node Type Enum (Main Sequence))	Main Sequence
tmRoot (Node Type Enum (Timing Root))	Timing Root

Enumeration Value	Description
withEffect (Node Type Enum (With Effect))	With Effect
withGroup (Node Type Enum (With Group))	With Group

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTimeNodeType) is located in §A.3. *end note*]

19.7.48 ST_TLTriggerEvent (Trigger Event)

This simple type specifies a particular event that causes the time condition to be true.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
begin (Trigger Event Enum (Begin))	Fire trigger at the beginning
end (Trigger Event Enum (End))	Fire trigger at the end
onBegin (Trigger Event Enum (On Begin))	Fire trigger at the beginning
onClick (Trigger Event Enum (On Click))	Fire trigger on a mouse click
onDblClick (Trigger Event Enum (On Double Click))	Fire trigger on double-mouse click
onEnd (Trigger Event Enum (On End))	Fire trigger at the end
onMouseOut (Trigger Event Enum (On Mouse Out))	Fire trigger on mouse out
onMouseOver (Trigger Event Enum (On Mouse Over))	Fire trigger on mouse over
onNext (Trigger Event Enum (On Next))	Fire trigger on next node
onPrev (Trigger Event Enum (On Previous))	Fire trigger on previous node
onStopAudio (Trigger Event Enum (On Stop Audio))	Fire trigger on stop audio

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTriggerEvent) is located in §A.3. *end note*]

19.7.49 ST_TLTriggerRuntimeNode (Trigger RunTime Node)

This simple type specifies the child time node that triggers a time condition. References a child TimeNode or all child nodes. Order is based on the child's end time.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
all (Trigger RunTime Node Enum (All))	All
first (Trigger RunTime Node (First))	First
last (Trigger RunTime Node (Last))	Last

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TLTriggerRuntimeNode) is located in §A.3. *end note*]

19.7.50 ST_TransitionCornerDirectionType (Transition Corner Direction Type)

This simple type specifies diagonal directions for slide transitions.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
ld (Transition Corner Direction Enum (Left-Down))	Specifies the slide transition direction of left-down
lu (Transition Corner Direction Enum (Left-Up))	Specifies the slide transition direction of left-up
rd (Transition Corner Direction Enum (Right-Down))	Specifies the slide transition direction of right-down
ru (Transition Corner Direction Enum (Right-Up))	Specifies the slide transition direction of right-up

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TransitionCornerDirectionType) is located in §A.3. *end note*]

19.7.51 ST_TransitionEightDirectionType (Transition Eight Direction)

This simple type specifies the direction of an animation.

This simple type is a union of the following types:

- The ST_TransitionCornerDirectionType simple type (§19.7.50).
- The ST_TransitionSideDirectionType simple type (§19.7.53).

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TransitionEightDirectionType) is located in §A.3. *end note*]

19.7.52 ST_TransitionInOutDirectionType (Transition In/Out Direction Type)

This simple type specifies if a slide transition should go in or out.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
in (Transition In/Out Direction Enum (In))	Specifies the slide transition should go in
out (Transition In/Out Direction Enum (Out))	Specifies the slide transition should go out

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TransitionInOutDirectionType) is located in §A.3. *end note*]

19.7.53 ST_TransitionSideDirectionType (Transition Side Direction Type)

This simple type defines a set of slide transition directions.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
d (Transition Side Direction Enum (Down))	Specifies that the transition direction is down
l (Transition Side Direction Enum (Left))	Specifies that the transition direction is left
r (Transition Side Direction (Right))	Specifies that the transition direction is right
u (Transition Side Direction Enum (Up))	Specifies that the transition direction is up

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TransitionSideDirectionType) is located in §A.3. *end note*]

19.7.54 ST_TransitionSpeed (Transition Speed)

This simple type defines the allowed transition speeds for transitioning from the current slide to the next.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
fast (Fast)	Fast slide transition.
med (Medium)	Medium slide transition.
slow (low)	Slow slide transition.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_TransitionSpeed) is located in §A.3. *end note*]

19.7.55 ST_ViewType (List of View Types)

This simple type specifies the kind of view that should be used when displaying the presentation document to the user.

This simple type's contents are a restriction of the W3C XML Schema token datatype.

This simple type is restricted to the values listed in the following table:

Enumeration Value	Description
handoutView (Handout View)	Handout View mode should be used.
notesMasterView (Notes Master View)	Notes Master View mode should be used.
notesView (Notes View)	Notes View mode should be used.
outlineView (Outline View)	Outline View mode should be used.
sldMasterView (Slide Master View)	Slide Master View mode should be used.
sldSorterView (Slide Sorter View)	Slide Sorter View mode should be used.
sldThumbnailView (Slide Thumbnail View)	Slide Thumbnail View mode should be used.
sldView (Normal Slide View)	Normal Slide View mode should be used.

[*Note*: The W3C XML Schema definition of this simple type's content model (ST_ViewType) is located in §A.3. *end note*]

