[MS-VGSFF]:

Visio Graphics Service (.vdw) File Format

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights**. This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets**. Microsoft does not claim any trade secret rights in this documentation.
- **Patents**. Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **License Programs**. To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.
- **Trademarks**. The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names**. The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

Reservation of Rights. All other rights are reserved, and this notice does not grant any rights other than as specifically described above, whether by implication, estoppel, or otherwise.

Tools. The Open Specifications documentation does not require the use of Microsoft programming tools or programming environments in order for you to develop an implementation. If you have access to Microsoft programming tools and environments, you are free to take advantage of them. Certain Open Specifications documents are intended for use in conjunction with publicly available standards specifications and network programming art and, as such, assume that the reader either is familiar with the aforementioned material or has immediate access to it.

Support. For questions and support, please contact dochelp@microsoft.com.

Revision Summary

Date	Revision History	Revision Class	Comments
7/13/2009	0.1	Major	Initial Availability
8/28/2009	0.2	Editorial	Revised and edited the technical content
11/6/2009	0.3	Editorial	Revised and edited the technical content
2/19/2010	1.0	Major	Updated and revised the technical content
3/31/2010	1.01	Editorial	Revised and edited the technical content
4/30/2010	1.02	Editorial	Revised and edited the technical content
6/7/2010	1.03	Editorial	Revised and edited the technical content
6/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
7/23/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	1.04	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	1.04	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	2.0	Major	Significantly changed the technical content.
4/11/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
9/12/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	3.0	Major	Significantly changed the technical content.
11/18/2013	3.0	None	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	3.0	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	3.0	None	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
7/31/2014	3.0	None	No changes to the meaning, language, or formatting of the technical content.
10/30/2014	3.0	None	No changes to the meaning, language, or formatting of the technical content.
3/16/2015	4.0	Major	Significantly changed the technical content.
9/4/2015	4.0	None	No changes to the meaning, language, or formatting of the technical content.
7/15/2016	4.0	None	No changes to the meaning, language, or formatting of the technical content.
9/14/2016	4.0	None	No changes to the meaning, language, or formatting of the technical content.
9/19/2017	4.1	Minor	Clarified the meaning of the technical content.
7/24/2018	5.0	Major	Significantly changed the technical content.
10/1/2018	6.0	Major	Significantly changed the technical content.
12/11/2018	6.0	None	No changes to the meaning, language, or formatting of the technical content.
4/22/2021	7.0	Major	Significantly changed the technical content.
8/17/2021	8.0	Major	Significantly changed the technical content.
4/16/2024	9.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The Visio Graphics Service (.vdw) File Format describes a Web drawing, which is a collection of drawing pages, shapes, fonts, images, data connections, and diagram update information that can be rendered as a vector or raster drawing.

Sections 1.7 and 2 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

A1: A **reference style** in which each column is identified sequentially from left-to-right with a letter or series of letters in alphabetical order. Column headings are ordered A-Z, then AA-AZ, BA-BZ... ZA-ZZ, AAA-AAZ, and so forth. Each row is numbered sequentially from the top down.

ActiveX Data Objects (ADO): A data access interface that connects to, retrieves, manipulates, and updates data in Object Linking and Embedding (OLE) database-compliant data sources.

add-in: Supplemental functionality that is provided by an external application or macro to extend the capabilities of an application.

American National Standards Institute (ANSI) character set: A character set defined by a code page approved by the American National Standards Institute (ANSI). The term "ANSI" as used to signify Windows code pages is a historical reference and a misnomer that persists in the Windows community. The source of this misnomer stems from the fact that the Windows code page 1252 was originally based on an ANSI draft, which became International Organization for Standardization (ISO) Standard 8859-1 [ISO/IEC-8859-1]. In Windows, the ANSI character set can be any of the following code pages: 1252, 1250, 1251, 1253, 1254, 1255, 1256, 1257, 1258, 874, 932, 936, 949, or 950. For example, "ANSI application" is usually a reference to a non-**Unicode** or code-page-based application. Therefore, "ANSI character set" is often misused to refer to one of the character sets defined by a Windows code page that can be used as an active system code page; for example, character sets defined by code page 1252 or character sets defined by code page 950. Windows is now based on **Unicode**, so the use of ANSI character sets is strongly discouraged unless they are used to interoperate with legacy applications or legacy data.

assembly name: The name of a collection of one or more files that is versioned and deployed as a unit. See also assembly.

Boolean: An operation or expression that can be evaluated only as either true or false.

bounding rectangle: A frame that encompasses an object. A bounding rectangle is not rotated and, therefore, always aligns along the x and y axes.

class name: The name that is used to refer to a class module that provides an implementation of a behavior.

color space: A system that describes color numerically by mapping color components to a multidimensional coordinate system. The number of dimensions is typically two, three, or four. For example, if colors are expressed as a combination of the three components red, green, and blue, a three-dimensional space can describe all possible colors. Grayscale colors can be mapped to a two-dimensional color space. If transparency is considered a component, four dimensions are appropriate. Also referred to as color model.

connection string: A series of arguments, delimited by a semicolon, that defines the location of a database and how to connect to it.

culture name: A part of a language identification tagging system, as described in [RFC1766]. Culture names adhere to the format "<languagecode2>-<country/regioncode2>." If a two-letter language code is not available, a three-letter code that is derived from [ISO-639] is used.

data provider: A known data source that is specific to a target type and that provides data to a collector type.

data source: A database, web service, disk, file, or other collection of information from which data is queried or submitted. Supported data sources vary based on application and data provider.

data type: A property of a field that defines the kind of data that is stored in the field, or defines the kind of data returned by an expression when the expression is evaluated.

drawing: A collection of drawing objects, such as shapes, curves, or WordArt, that are viewed together as a single image.

embedded image: An image that is stored within a document rather than being linked to a source file that is outside the document.

Extensible Application Markup Language (XAML): A declarative XML-based language that is used to represent a tree of objects. XAML markup is stored in .xaml files or, for workflow schemas, .xoml files.

field: An element or attribute in a data source that can contain data.

floating-point number: A number that is represented by a mantissa and an exponent according to a given base. The mantissa is typically a value between "0" and "1". To find the value of a floating-point number, the base is raised to the power of the exponent, and the mantissa is multiplied by the result.

font: An object that defines the graphic design, or formatting, of a collection of numbers, symbols, and letters. A font specifies the style (such as bold and strikeout), size, family (a typeface such as Times New Roman), and other qualities to describe how the collection is drawn.

font family: A set of fonts that all have common stroke width and serif characteristics. For example, Times Roman and Times Roman Italic are members of the same font family.

globally unique identifier (GUID): A term used interchangeably with universally unique identifier (UUID) in Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the value. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] must be used for generating the **GUID**. See also universally unique identifier (UUID).

header row: A row in a table, typically the first row, that contains labels for columns in the table.

hue-saturation-luminance (HSL): A color model that defines a color by using three dimensions: hue, the color itself; saturation, the purity of the color; and luminance, the amount of light that is either reflected or absorbed by the color. See also color scheme and **color space**.

language code identifier (LCID): A 32-bit number that identifies the user interface human language dialect or variation that is supported by an application or a client computer.

list: A container within a SharePoint site that stores list items. A list has a customizable schema that is composed of one or more fields.

Office data connection (ODC) file: A file that stores information about a connection to a data source, such as an Access database, worksheet, or text file. This file facilitates data source administration.

OLE compound file: A form of structured storage, as described in [MS-CFB]. A compound file allows independent storages and streams to exist within a single file.

OLE DB: A set of interfaces that are based on the Component Object Model (COM) programming model and expose data from a variety of sources. These interfaces support the amount of Database Management System (DBMS) functionality that is appropriate for a data store and they enable a data store to share data.

Open Database Connectivity (ODBC): A standard software API method for accessing data that is stored in a variety of proprietary personal computer, minicomputer, and mainframe databases. It is an implementation of [ISO/IEC9075-3:2008] and provides extensions to that standard.

pixel: A discrete unit of display on a computer display device.

Portable Network Graphics (PNG): A bitmap graphics file format that uses lossless data compression and supports variable transparency of images (alpha channels) and control of image brightness on different computers (gamma correction). PNG-format files have a .png file name extension.

primary key: A field or set of fields that uniquely identifies each record in a table. A primary key cannot contain a null value.

query: A formalized instruction to a data source to either extract data or perform a specified action. A query can be in the form of a query expression, a method-based query, or a combination of the two. The data source can be in different forms, such as a relational database, XML document, or in-memory object. See also search query.

range: An addressable region that is in a workbook. A range typically consists of zero or more cells and represents a single, contiguous rectangle of cells on a single sheet.

red-green-blue (RGB): A color model that describes color information in terms of the red (R), green (G), and blue (B) intensities in a color.

reference style: A system that is used in formulas to specify cells or ranges of cells. A reference style specifies a cell in a two-dimensional table by identifying the row and column that contain that cell or range of cells.

refresh: A process that retrieves values from a data source and populates a workbook with those values.

row: A collection of columns that contains property values that describe a single item in a set of items that match the restriction specified in a query.

rule: A condition or action, or a set of conditions or actions, that performs tasks automatically based on events and values.

sheet: A part of an Excel workbook. There are four types of sheets: worksheet, macro sheet, dialog sheet, and chart sheet. Multiple sheets are stored together within a workbook.

storage: An element of a compound file that is a unit of containment for one or more storages and streams, analogous to directories in a file system, as described in [MS-CFB].

stream: An element of a compound file, as described in [MS-CFB]. A stream contains a sequence of bytes that can be read from or written to by an application, and they can exist only in storages.

Structured Query Language (SQL): A database query and programming language that is widely used for accessing, querying, updating, and managing data in relational database systems.

Unicode: A character encoding standard developed by the Unicode Consortium that represents almost all of the written languages of the world. The **Unicode** standard [UNICODE5.0.0/2007] provides three forms (UTF-8, UTF-16, and UTF-32) and seven schemes (UTF-8, UTF-16, UTF-16 BE, UTF-16 LE, UTF-32, UTF-32 LE, and UTF-32 BE).

Unicode code point: Any value in the Unicode codespace, which is a range of integers from "0" to "10FFFF16". Each code point is a unique positive integer that maps to a specific character.

Uniform Resource Identifier (URI): A string that identifies a resource. The URI is an addressing mechanism defined in Internet Engineering Task Force (IETF) Uniform Resource Identifier (URI): Generic Syntax [RFC3986].

Uniform Resource Locator (URL): A string of characters in a standardized format that identifies a document or resource on the World Wide Web. The format is as specified in [RFC1738].

view: See form view (Microsoft InfoPath), list view (SharePoint Products and Technologies), or **View** (Microsoft Business Connectivity Services).

whitespace: A character that can be found between words, including a space (" "), a carriage return in combination with a line feed (newline), and a tab character.

workbook: A container for a collection of **sheets**.

zero-based index: An index in which the first item has an index of "0" (zero).

zoom level: The degree to which a portion of an image, document, or other screen object is made to appear closer or farther away relative to its default appearance. This value is usually expressed as a percentage of the default appearance.

MAY, SHOULD, MUST, SHOULD NOT, MUST NOT: These terms (in all caps) are used as defined in [RFC2119]. All statements of optional behavior use either MAY, SHOULD, or SHOULD NOT.

1.2 References

Links to a document in the Microsoft Open Specifications library point to the correct section in the most recently published version of the referenced document. However, because individual documents in the library are not updated at the same time, the section numbers in the documents may not match. You can confirm the correct section numbering by checking the Errata.

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information.

[IEEE754] IEEE, "IEEE Standard for Binary Floating-Point Arithmetic", IEEE 754-1985, October 1985, http://ieeexplore.ieee.org/servlet/opac?punumber=2355

[ISO/IEC-14496-22] International Organization for Standardization, "Information technology -- Coding of audio-visual objects -- Part 22: Open Font Format", 2007, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=43466

[ISO/IEC29500-1:2011] ISO/IEC, "Information Technology -- Document description and processing languages -- Office Open XML File Formats -- Part 1: Fundamentals and Markup Language Reference", ISO/IEC 29500-1:2011, 2011, http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=59575

[ISO/IEC29500-2:2011] ISO/IEC, "Information technology -- Document description and processing languages -- Office Open XML File Formats -- Part 2: Open Packaging Conventions", ISO/IEC 29500-

2:2011, 2011,
http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=59576

[MS-CFB] Microsoft Corporation, "Compound File Binary File Format".

[MS-OAUT] Microsoft Corporation, "OLE Automation Protocol".

[MS-ODBCSTR] Microsoft Corporation, "ODBC Connection String Structure".

[MS-SLXV] Microsoft Corporation, "Silverlight XAML Vocabulary Specification 2008".

[MS-WPFXV] Microsoft Corporation, "WPF XAML Vocabulary Specification 2006".

[MS-XAML] Microsoft Corporation, "XAML Object Mapping Specification 2006".

[RFC2083] Boutell, T., et al., "PNG (Portable Network Graphics) Specification Version 1.0", RFC 2083, March 1997, https://www.rfc-editor.org/info/rfc2083

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, https://www.rfc-editor.org/info/rfc2119

[RFC4234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005, https://www.rfc-editor.org/info/rfc4234

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006, https://www.rfc-editor.org/info/rfc4627

[XMLSCHEMA1] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, https://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, https://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

1.2.2 Informative References

[HTML] World Wide Web Consortium, "HTML 4.01 Specification", W3C Recommendation, December 1999, http://www.w3.org/TR/html4/

[MSDN-CompareOptions] Microsoft Corporation, "CompareOptions Enum", https://learn.microsoft.com/en-us/dotnet/api/system.globalization.compareoptions

[MSDN-ENCLOC] Microsoft Corporation, "Encoding and Localization", .NET Framework Developer's Guide, http://msdn.microsoft.com/en-us/library/h6270d0z.aspx

[MSDN-FormattingTypes] Microsoft Corporation, "Formatting Types", .NET Developer's Framework, http://msdn.microsoft.com/en-us/library/fbxft59x.aspx

[MSDN-OLEDB] Microsoft Corporation, "Microsoft OLE DB", https://learn.microsoft.com/en-us/previous-versions/windows/desktop/ms722784(v=vs.85)

[MSDN-ToDouble] Microsoft Corporation, "Convert.ToDouble Method", .NET Framework Class Library, http://msdn.microsoft.com/en-us/library/system.convert.todouble.aspx

1.3 Overview

This document describes data contained in a compound file. There are several streams and **storages** within the File Structure, each using a different persistence format for its data. A **stream** named

"VisioServerData" contains a ZIP archive that stores all the information needed to describe a Web drawing.

A ShapeGraphic XML Part in the ZIP archive describes the graphical elements displayed in the Web drawing. These graphical elements are expressed in Extensible Application Markup Language (XAML). Graphical elements can be static or dynamic. Dynamic graphical elements, also called data graphics, have visual properties that are bound to data in a **data source**, and the appearance of these elements changes as data in the data source refreshes.

A collection of XML Parts in the ZIP archive describes the Web drawing's data connections, information about its shape and the diagram update logic its dynamic graphical elements. These parts are expressed as XML. One of these parts, the DataGraphic XML Part, describes recalculated properties that are to be evaluated after the data in the Web drawing has been refreshed from data sources. In particular, a grammar for formula evaluation exists to describe how changes in the data are translated into changes in properties of graphical elements in the Web drawing. See Formula Evaluation and Shape Property Recalculation for a detailed explanation of this grammar.

Additional items in the ZIP archive contain information for the **fonts** and images used in the Web drawing. See Fonts and Images for more information.

1.4 Relationship to Protocols and Other Structures

This file format specifies the streams and **storages** in an **OLE compound file** that are required to render a Web drawing. For details about OLE compound files see [MS-CFB].

1.5 Applicability Statement

This document specifies a persistence format for Web drawing content, which can include drawing pages, shapes, **fonts**, images, data connections, and recalculation information, as described in Section 2.2.1. The persistence format is applicable when the document content is graphical in nature.

This persistence format is applicable for use as a stand-alone document.

This persistence format provides interoperability with applications that create or read documents conforming to this structure.

1.6 Versioning and Localization

This document covers versioning and localization issues in the following areas:

- Document version: App XML Part, section 2.4.2
- Default document language: App XML Part, section 2.4.2
- Local overrides to document language are specified in attributes, properties and function arguments in individual sections of this document.

1.7 Vendor-Extensible Fields

This persistence format can be extended by storing information in streams and **storages** that are not specified in Section 2. Implementations are not required to preserve or remove additional **streams** or storages when modifying an existing document.

2 Structures

This section specifies the overall structure of a file that conforms to this specification.

2.1 File Structure Overview

A file of the type specified by this specification MUST be a compound file.

A compound file contains **storages** and streams. A stream that contains a ZIP package is used to persist the information necessary to represent fully a Web drawing. This package contains a collection of parts that are used to persist data in XML or standard binary formats and specify various aspects of the Web drawing as well as the structure of the package.

The following figure shows a possible implementation of a file specified by this specification.

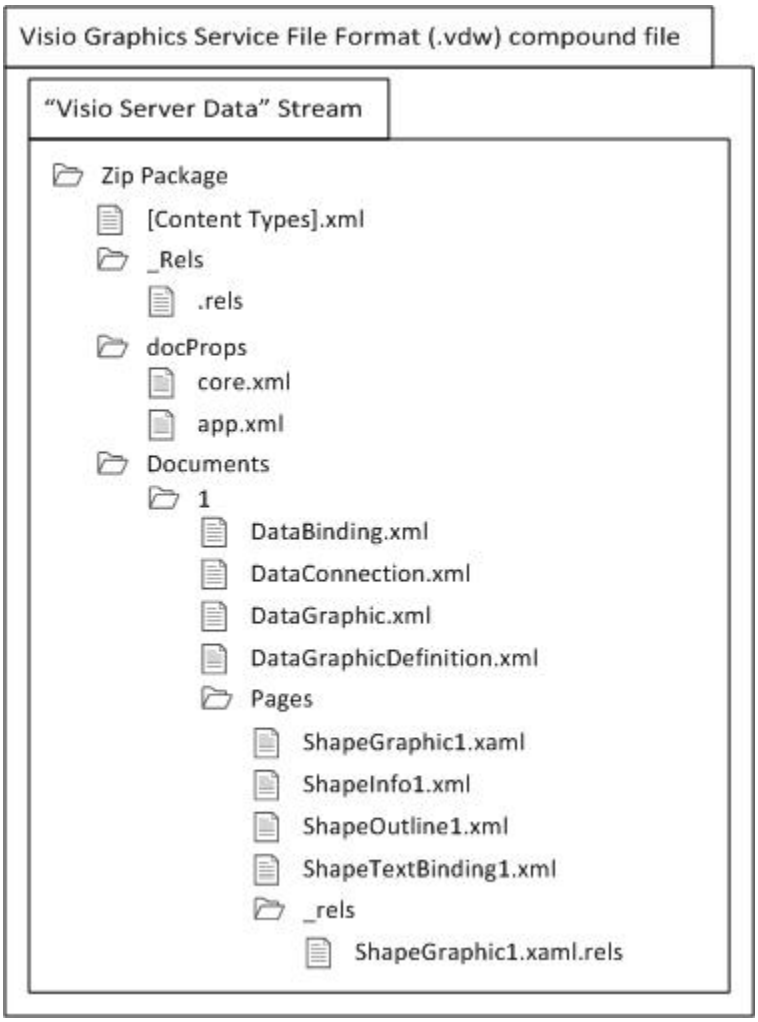

Figure 1: Possible Implementation of a File

2.1.1 Compound File

A file of the type specified by this document MUST be an **OLE compound file** as specified in [MS-CFB].

2.1.2 Streams

A file of the type specified by this document consists of **streams** as specified in [MS-CFB].

A file of the type specified by this document MUST contain exactly one stream with the name "VisioServerData". This stream completely specifies the information necessary to represent a Web drawing. Other streams that exist in the file are unused and MUST be ignored when rendering a Web drawing.

2.1.3 Package

The "VisioServerData" stream MUST contain exactly one package that is a ZIP archive that conforms to the Open Packaging Conventions as specified in [ISO/IEC29500-2:2011], the further packaging restrictions specified in [ISO/IEC29500-1:2011] Section 9, and this specification.

2.1.4 Parts

A package is composed of multiple parts as specified in [ISO/IEC29500-2:2011] Section 9.1. Each part has an associated content type that specifies the format it is persisted in. Each part can also be the target or the source of a connection between two parts called a relationship, as specified in [ISO/IEC29500-2:2011] Section 9.3.

The valid parts, content types, and required and optional relationships between all parts in this package are specified in Part Enumeration.

2.1.5 Relationships

A relationship specifies a connection between a source and a target resource as specified in [ISO/IEC29500-1:2011] Section 8.3. Relationship identifiers are used in binary and XML part content to reference unique relationship elements in relationship parts that in turn target other resources. There are several different types of relationships:

- A package relationship is a relationship where the target is a part and the source is the package as a whole.
- A part-to-part relationship is a relationship where the target is a part and the source is a part in the package.
- An explicit relationship is a relationship where a resource is referenced from the contents of a source part by referencing the ID attribute value of a relationship element.
- An implicit relationship is a relationship that is not explicit.
- An internal relationship is a relationship where the target is a part in the package.
- An external relationship is a relationship where the target is an external resource not in the package.

2.1.6 Part Enumeration

The "VisioServerData" stream contains the following ZIP package parts and relationships:

Part Name	Relationship target of...
Core	package
App	package

Part Name	Relationship target of...
Fonts	package
Images	ShapeGraphic
DataBinding	package
DataConnection	package
DataGraphic	package
DataGraphicDefinition	package
ShapeInfo	ShapeGraphic
ShapeOutline	ShapeGraphic
ShapeTextBinding	ShapeGraphic
Rels	package
ShapeGraphic	package
Content Type	package

2.1.6.1 ShapeGraphic

Content Type:	application/xaml+xml
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties

An instance of this part specifies a vector description of a drawing page and all the shapes contained within it. The syntax of the structures contained in this part is specified in the ShapeGraphic XML Part.

One ShapeGraphic part MUST exist in the package for each unique value of the **ID** attribute across all the CT_PageMetaData elements contained in the App XML Part. The name of each part of this type MUST be of the form "ShapeGraphic*N*.xaml", where N is the value of the associated **ID** attribute.

Each ShapeGraphic part MUST be the target of a package relationship.

A ShapeGraphic part is permitted to have implicit relationships to the following parts:

- ShapeTextBinding
- ShapeOutline
- ShapeInfo
- Images

2.1.6.2 Fonts

Content Type:	application/vnd.openxmlformats-officedocument.obfuscatedFont
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/font

An instance of this part type specifies a **font** resource used in rendering a Web drawing. Each part of this type is a binary font file that conforms to the Open Type Font Format as specified in [ISO/IEC-14496-22]. The part MUST be modified from the original Open Type Font Format in the following way:

The part name MUST of the form "$B_{03}B_{02}B_{01}B_{00}$-$B_{11}B_{10}$-$B_{21}B_{20}$-$B_{30}B_{31}$-$B_{32}B_{33}B_{34}B_{35}B_{36}B_{37}$.odttf" where each "$B_{xy}$" is a two digits hexadecimal representation of a byte taken from a **GUID**.

The part data MUST satisfy the following requirements:

- The first 16 bytes of the part data MUST be the result of a binary XOR operation of the corresponding original Open Type Font Format byte with B_{37}, B_{36}, B_{35}, B_{34}, B_{33}, B_{32}, B_{31}, B_{30}, B_{20}, B_{21}, B_{10}, B_{11}, B_{00}, B_{01}, B_{02}, and B_{03}.
- The next 16 bytes of the part data MUST be the result of a binary XOR operation of the corresponding original Open Type Font Format byte with the same sequence of bytes extracted from the part name.

One Fonts part MUST exist in the package for each unique value of the **FontURI** attribute across all **Glyph** elements contained within all ShapeGraphic parts, as specified in [MS-SLXV] section 1.22.31.1.4, that does not contain the string "CommonFonts/". The name of the Fonts part MUST be equal to the value of **FontURI**.

All Fonts parts MUST be the target of a package relationship. A Fonts part MUST NOT have implicit or explicit relationships to any other part specified in this specification.

2.1.6.3 Images

Content Type:	**image/png**
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/image

An instance of this part type specifies an image resource used in rendering a Web drawing. Each part of this type is an image file that MUST support the **Portable Network Graphics (PNG)** format specified in [RFC2083].

One Images part MUST exist in the package for each unique value of the following attributes of elements contained within all ShapeGraphic parts:

- All **Source** attributes of the Image elements as specified in [MS-SLXV] section 1.22.31.1.5. The name of the Images part MUST be equal to the value of **Source**.
- All **ImageSource** attributes of the ImageBrush elements as specified in [MS-SLXV] section 1.22.2.3.1. The name of the Images part MUST be equal to the value of **ImageSource**.

All Images parts MUST be the target of an implicit relationship from the related ShapeGraphic part. An Images part MUST NOT have implicit or explicit relationships to any other part specified in this specification.

2.1.6.4 App

This part is specified in [ISO/IEC29500-1:2011] Section 15.2.12.3. The syntax of the structures contained in this part is specified in the App XML Part.

2.1.6.5 Core

This part is specified in [ISO/IEC29500-1:2011] Section 15.2.12.1. The syntax of the structures contained in this part is specified in the Core XML Part.

2.1.6.6 DataBinding

Content Type:	application/xml
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties

An instance of this part type specifies the information that is used to create data bindings between a recordsets and shapes in drawing pages in a Web drawing. The syntax of the structures contained in this part is specified in the DataBinding XML Part.

A package MUST contain exactly one DataBinding part. The DataBinding part MUST have the name "DataBinding.xml". The DataBinding part MUST be the target of a package relationship.

The DataBinding part MUST NOT have implicit or explicit relationships to any other part specified in this specification.

2.1.6.7 DataConnection

Content Type:	application/xml
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties

An instance of this part type specifies the data connection information needed to **query data sources** and refresh the recordsets referenced by a Web drawing. The syntax of the structures contained in this part is specified in the DataConnection XML Part.

A package MUST contain exactly one DataConnection part. The DataConnection part MUST have the name "DataConnection.xml". The DataConnection part MUST be the target of a package relationship.

The DataConnection part MUST NOT have implicit or explicit relationships to any other part specified in this specification.

2.1.6.8 DataGraphicDefinition

Content Type:	application/xml
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties

An instance of this part type specifies information used to update the image element data graphics in a Web drawing. The syntax of the structures contained in this part is specified in the DataGraphicDefinition XML Part.

A package MUST contain exactly one DataGraphicDefinition part. The DataGraphicDefinition part MUST have the name "DataGraphicDefinition.xml". The DataGraphicDefinition part MUST be the target of a package relationship.

The DataGraphicDefinition part MUST NOT have implicit or explicit relationships to any other part specified in this specification.

2.1.6.9 DataGraphic

Content Type:	application/xml
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties

An instance of this part type specifies information used to update all data graphics in a Web drawing. The syntax of the structures contained in this part is specified in the DataGraphic XML Part.

A package MUST contain exactly one DataGraphic part. The DataGraphic part MUST have the name "DataGraphic.xml". The DataGraphic part MUST be the target of a package relationship.

The DataGraphic part MUST NOT have implicit or explicit relationships to any other part specified in this specification.

2.1.6.10 ShapeInfo

Content Type:	application/xml
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties

An instance of this part type specifies information about the shapes on a drawing page in a Web drawing. The syntax of the structures contained in this part is specified in the ShapeInfo XML Part.

One ShapeInfo part MUST exist for each ShapeGraphic part in the package. The ShapeInfo part MUST be the target of an implicit relationship from the related ShapeGraphic part. The name of each ShapeInfo part MUST be of the form "ShapeInfo*N*.xml", where N is the value of the **ID** attribute of the CT_PageMetaData element contained in the App XML Part that also corresponds to the ShapeGraphic part.

The ShapeInfo part MUST NOT have implicit or explicit relationships to any other part specified in this specification.

2.1.6.11 ShapeTextBinding

Content Type:	application/xml
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties

An instance of this part type specifies information used to update the text element data graphics in a Web drawing. The syntax of the structures contained in this part is specified in the ShapeTextBinding XML Part.

One ShapeTextBinding part MUST exist for each ShapeGraphic part in the package. The ShapeTextBinding part MUST be the target of an implicit relationship from the related ShapeGraphic part. The name of each ShapeTextBinding part MUST be of the form "ShapeTextBinding*N*.xml", where N is the value of the **ID** attribute of the CT_PageMetaData element contained in the App XML Part that also corresponds to the ShapeGraphic part.

The ShapeTextBinding part MUST NOT have implicit or explicit relationships to any other part specified in this specification.

2.1.6.12 ShapeOutline

Content Type:	application/xml
Source Relationship:	http://schemas.openxmlformats.org/officeDocument/2006/relationships/custom-properties

An instance of this part type specifies the geometric outline information of the shapes in a drawing page in a Web drawing. The syntax of the structures contained in this part is specified in the ShapeOutline XML Part.

One ShapeOutline part MUST exist for each ShapeGraphic part in the package. The ShapeOutline part MUST be the target of an implicit relationship from the related ShapeGraphic part. The name of each ShapeOutline part MUST be of the form "ShapeOutline*N*.xml", where N is the value of the **ID** attribute

of the CT_PageMetaData element contained in the App XML Part that also corresponds to the ShapeGraphic part.

The ShapeOutline part MUST NOT have implicit or explicit relationships to any other part specified by this specification.

2.1.6.13 Rels

This part and its syntax are specified in [ISO/IEC29500-2:2011] Section 9.3.

2.1.6.14 ContentType

This part and its syntax are specified in [ISO/IEC29500-2:2011] Section 10.1.2.

2.2 Conceptual Overview

This section specifies how higher-level features of the file format are represented by combinations of parts.

2.2.1 Web Drawing

A Web drawing is a collection of drawing pages, shapes, **fonts**, images, data connections, and recalculation information required to render the **drawing** in a web browser.

A Web drawing is specified by the "VisioServerData" stream within the file format as specified in the File Structure Overview.

Properties of the Web drawing such as the name of the application that created it, the list of drawing pages it contains, and its description are specified by the child elements of the CT_Properties element in the App XML Part and by the CT_coreProperties element in the Core XML Part.

For examples of the contents of the "VisioServerData" stream for various Web drawings, see Structure Examples.

2.2.2 Drawing Page

A drawing page is a collection of shapes that are viewed together.

The graphical information necessary to render a drawing page is specified by the ShapeGraphic XML Part. Additional information about shapes in the drawing page and about the properties of the drawing page itself is specified by a ShapeInfo XML Part.

A drawing page is uniquely identified by **ID** and **Name** attributes in a CT_PageMetaData element in the App XML Part. The following elements in other parts of the document have attributes that are equal to **ID** or **Name** and specify supplementary information about the drawing page:

- A CT_Page element in the ShapeInfo XML Part has a **Name** attribute that is equal to the **Name** attribute of CT_PageMetaData and specifies information about the shapes in the drawing page and the default **view** for the drawing page.
- A CT_PageInfo element in the ShapeInfo XML Part has a **Name** attribute that is equal to the **Name** attribute of CT_PageMetaData.
- A CT_DataRecordset element in the DataConnection XML Part has a **Pages** attribute that specifies a comma-separated list of indices of drawing pages. The index of each drawing page in this list is equal to the **ID** attribute of one of the **PageMetaData** child elements of CT_PagesMetaData in the

ShapeInfo XML Part. A CT_DataRecordset element specifies a recordset linked to shapes in these drawing pages.

- A CT_DataGraphics_Page element in the DataGraphic XML Part has a **PageName** attribute that is equal to the **Name** attribute of CT_PageMetaData and specifies information used to update the data graphics in the drawing page.

A drawing page has display characteristics such as **zoom level** and offset position of the point on the drawing page that is centered in the view are specified by the attributes of the CT_Page element in the ShapeInfo XML Part.

A drawing page can contain text, which can depend on various **fonts**. Each font used in rendering text in a drawing page is specified by a Fonts part and is referenced as specified in Fonts.

A drawing page can contain **embedded images**. Each image used in a drawing page is specified by an Images part and is referenced as specified in Images.

2.2.3 Shape

A shape is a collection of geometry, text, images, hyperlinks, and shape data in a drawing page.

The set of shapes in a drawing page is specified by the CT_ShapeInfo child elements of the CT_Page element in the ShapeInfo XML Part.

The geometry, text, formatting, positioning and image information for a shape is specified by the ShapeGraphic XML Part. Hyperlinks, shape data, and other shape properties are specified by the CT_ShapeInfo child element of the CT_Page element in the ShapeInfo XML Part. The following sections specify additional information about shapes.

2.2.3.1 Shape Identification

A shape is uniquely identified by the **Name** attribute in a CT_ShapeInfo element in the ShapeInfoXML Part. The following elements in other parts of the document have attributes that are equal to **Name** and specify supplementary information about this shape:

- A **Canvas** graphical element in the ShapeGraphic XML Part has a **Name** attribute that is equal to the **Name** attribute of CT_ShapeInfo and specifies the visual information about the shape.
- A CT_DataGraphics_Shape element in the DataGraphic XML Part has a **ShapeName** attribute that is equal to the **Name** attribute of CT_ShapeInfo and specifies information used to update the data graphics associated with the shape.

2.2.3.2 Shape Visualization

The graphical information necessary to render a shape is specified by the ShapeGraphic XML Part using Silverlight **XAML** and MUST conform to the restrictions specified in XAML Shapes.

Shapes that have no graphical or textual attributes bound to **data sources** are called static shapes. The display characteristics of static shapes will never change during the lifetime of the drawing page.

Shapes that have graphical attributes bound to data sources are called recalculated shapes. When the data in the data source changes, the graphical attributes of these shapes are recalculated and the visual appearance of the shapes change. Additional restrictions on the XAML that specifies recalculated shapes are specified in XAML Recalculated Shapes.

For more information about binding data to shapes, see Data Binding to Web Drawing Elements.

2.2.3.3 Shape Selection

Interactivity with the web drawing, including shape selection, is enabled with the use of geometric outlines of shapes. The geometric outline of a shape is specified by the **Canvas** element in the ShapeGraphic XML Part that corresponds to the shape and is also specified by a CT_Shape element in the ShapeOutline XML Part. The value of the **Name** attribute of the CT_Shape element MUST be equal to the value of the **Name** attribute of the CT_ShapeInfo element that corresponds to the shape.

Additional geometric information is specified to facilitate the highlighting of a selected shape. The selected shape is visually indicated by highlighting its **bounding rectangle**. The bounding rectangle of a shape is specified by the **Layout** attribute of the corresponding CT_ShapeInfo element in the ShapeInfo XML Part.

2.2.3.4 Shape Hyperlinks

A shape can have one or more hyperlinks associated with it. Hyperlinks can point to destinations outside the Web drawing or to drawing pages and shapes within the Web drawing. The set of hyperlinks associated with a shape is specified by the CT_Hyperlinks child element of the corresponding CT_ShapeInfo element in the ShapeInfo XML Part.

The properties of each hyperlink, such as its name, **Uniform Resource Identifier (URI)**, and description, are specified by the attributes of the CT_Hyperlink child element of the CT_Hyperlinks element in the ShapeInfo XML Part.

2.2.3.5 Shape Data

A shape can have data associated with it which provides information about its meaning. A shape's data is stored as a set of shape data items. The set of shape data items associated with a shape is specified by the CT_ShapeDataItems child element of the corresponding CT_ShapeInfo element in the ShapeInfo XML Part.

The properties of each shape data item, such as its label, value, and **data type**, are specified by the attributes of the CT_ShapeData child element of the CT_ShapeDataItems element in the ShapeInfo XML Part. The **BindingID** attribute of this element is an identifier that specifies the **row** of data in a recordset that is bound to this shape data item, as specified in the CT_Binding element of the DataBinding XML Part. If the **BindingID** attribute is not present, then the shape data item is not bound to data in a **data source**.

2.2.4 Data Connectivity and Refresh

This section describes how **data sources** can be referenced, queried and connected to from within a Web drawing.

2.2.4.1 Data Connections

A Web drawing can be linked to data from databases and other **data sources** which can affect various attributes of the Web drawing including its visual appearance. A relationship to such data sources is called a data connection.

A data connection contains properties that specify how the application connects to and **queries** a data source, including the type of **data provider** (for example, **OLE DB** or **ODBC**) required to access a data source, the name of the server on which the data source is hosted, security information to access the data source, and a query to execute on the server.

Data connections are stored either completely inside or partly outside of the document. If the **FileName** attribute of a CT_DataConnection element in the DataConnection XML Part is not present, a data connection is completely specified by the following pair of elements:

- The containing CT_DataConnection.
- The CT_DataRecordset with a **connectionID** attribute that matches the **ID** attribute of the containing CT_DataConnection.

If the **FileName** attribute is present, then a data connection is specified by the triplet of:

- The containing CT_DataConnection.
- The CT_DataRecordset with a **connectionID** attribute that matches the **ID** attribute of the containing CT_DataConnection.
- The information contained in the **Office data connection (ODC) file** found at the path described by the value of the **FileName** attribute.

A Web drawing can contain zero or more data connections, each uniquely identified by the **ID** attribute of the CT_DataRecordset element. Data connections can be established for the types of data sources listed in the **ConnectionString** attribute of the CT_DataConnection element.

2.2.4.2 Recordset

A recordset is the data that is returned from a **data source**, organized into sets of **rows** and **fields**. The properties of this recordset, such as its identifier, the underlying data connection, the number and type of fields each row contains, the row addressing method, and the set of drawing pages on which the data is used, are specified in a CT_DataRecordset element in a DataConnection XML Part.

2.2.4.3 Recordset Refresh

The operation of replacing the contents of a recordset with data queried from an underlying **data source**, using the associated data connection, is called refreshing the recordset.

2.2.4.4 Recordset Row Addressing

There are two mutually exclusive **row** addressing methods that specify how to reference a particular row across **refresh** operations: the row order method and the primary key method.

2.2.4.4.1 Row Order Method

In the **row** order method, recordset rows are addressed by their position in the recordset, regardless of their contents. This addressing method is specified by the **RowOrder** attribute of the CT_DataRecordset element.

2.2.4.4.2 Primary Key Method

In the **primary key** method, recordset **rows** are addressed by value of their **primary key**. This is addressing method is specified by the **PrimaryKey** element of the CT_DataRecordset element.

2.2.5 Data Binding to Web Drawing Elements

This section describes how a shape data item can be bound to a recordset cell.

2.2.5.1 Data Binding

The association between a particular cell in a recordset, specified by a **row** and a **field**, and a shape data item is called a data binding.

A cell in a recordset can be bound to zero or more shape data items. A cell can be bound to only one shape data item per shape. A shape data item can be bound to only one cell.

The cell bound to a shape data item is specified by the **BindingID** and **Name** attributes of the corresponding CT_ShapeData element in the ShapeInfo XML Part. The **BindingID** attribute references a CT_Binding element which specifies the row of the recordset. The **Name** attribute references a CT_DataColumn element which specifies the field of the recordset.

The CT_BindingConnection element that contains this CT_Binding element referenced by **BindingID** specifies the recordset that contains the cell bound to the shape data item.

When the data in a cell in a recordset changes, the shape data item that it is bound to is updated in the following manner:

- The **Value** attribute of the CT_ShapeData is updated with the value of the data in the cell. This could involve a **data type** conversion from the data type of the cell as specified by the **Type** attribute of its corresponding CT_DataColumn, to the data type of the shape data item as specified by the **Type** attribute of the bound CT_ShapeData element.
- The **FormattedValue** attribute of the CT_ShapeData is updated with a string representation of the value of the data in the cell formatted for display. The formatting of the value into a string is done using the following attributes of the CT_ShapeData element: **Format, LangID, UnitLabel, CalendarID, CurrencyID, ServerAction, Unit and DisplayUnit.**

2.2.5.2 Diagram Update

Updating the vector description of the Web drawing from its current state to an updated state based on change in an underlying recordset is called a diagram update.

Formula expressions in these Web drawing elements that are dependent on data bound shape data items use the PtgDataBinding token to refer to the shape data items. The data bindings referred to by these shape data items enable the formula expressions to be recalculated from updated values obtained from a recordset.

2.2.5.3 Data Graphics

A set of **XAML** graphical elements that can be updated by formula expressions is called a data graphic. The following table describes the types of XAML elements that can be updated and the elements within the various XML parts that determine how the XAML elements are updated.

XAML Elements	XML Elements
Sheet Elements, Formatting Elements and Geometry Elements	CT_DataGraphics_Shape in the DataGraphic XML Part
Text Elements	CT_ShapeText in the ShapeTextBinding XML part
Image Elements	CT_DataGraphics_Shape in the DataGraphic XML Part

XAML graphical elements are made dependent on data bound shape data items when the formula expressions specified within the related XML elements contain PtgDataBinding tokens. The data bindings referred to by these shape data items enable the formula expressions to be recalculated from updated values obtained from a recordset.

2.2.6 Recalculating Shape Properties

This section describes concepts used when recalculating shape properties.

2.2.6.1 Color Table

The color table specifies a zero-based array of **red-green-blue (RGB)** color values for the document.

The color table is specified by the **ColorTable** attribute of the CT_DataGraphics element in the DataGraphic XML Part of the document.

Valid indexes into the color table MUST be greater than or equal to 0 and less than the size of the color table.

2.2.6.2 Formulas

The following sections describe the concepts and elements of a formula.

2.2.6.2.1 Formula Expression

A formula expression is a sequence of values and functions that, when evaluated, produce a new value.

A formula expression contains a sequence of parse tokens, each of which is an operand token, a control token, or a function token.

Formula expressions are stored as strings using Reverse-Polish notation. Reverse-Polish notation is a logical system for the specification of mathematical formulas in which operands are followed by operators.

2.2.6.2.2 Formula Evaluation

Evaluation of a formula specified in Reverse-Polish notation is usually based around an evaluation stack. Tokens are added to the evaluation stack as they are encountered in the Reverse-Polish notation and are removed as they are used in the evaluation. When expression evaluation is finished, there will be exactly one parse token left on the evaluation stack. This token is the result of the evaluation.

2.2.6.2.3 Parse Tokens

All tokens are stored as Parse Things (ptgs). The format of these tokens is TokenValue:TokenType. The syntax of TokenValue is described for each token in the Parse Token Definitions section. TokenType specifies the type of the token and MUST be an entry under the ID column in the Parse Token Table in the Parse Token Table section.

2.2.6.2.3.1 Control Tokens

Control tokens are used at runtime to control the evaluation stack. The control tokens are PtgJmp, PtgJmpF, PtgJmpT, PtgJmpLabel, PtgPop and PtgPushTop. The tokens PtgJmp, PtgJmpF, PtgJmpT, and PtgJmpLabel do not push values onto the evaluation stack but can pop values from the evaluation stack. The tokens PtgJmp, PtgJmpF, and PtgJmpT specify a jump location which MUST be present in a succeeding PtgJmpLabel token. When a PtgJmp, PtgJmpF, and PtgJmpT token is encountered, the succeeding tokens up to the matching PtgJmpLabel token in the formula expression are not pushed onto the evaluation stack and are not evaluated. Evaluation resumes at the token succeeding that PtgJmpLabel token.

2.2.6.2.3.2 Function Tokens

Function tokens represent variable or fixed argument functions. The TokenValue MUST be the name of the function as specified for each function by the ABNF func-name column in the Function Token Table in the Function Token Table section. The TokenType specifies the type of the token and MUST be

either PtgFunc or PtgFuncVar. For variable argument functions, the number of arguments is specified by the ABNF.

2.2.6.2.3.3 Operand Tokens

Operand tokens represent values that are used by functions. An operand specifies a single value that is persisted in the file as one of the tokens specified in the token group vAny, or as a PtgMissArg, PtgDataBinding or PtgRecalcRef. An operand token represents, and can be converted into, one of the following types of values:

- A string value
- A numeric value
- A **Boolean** value
- A currency value
- A color value
- A date value
- An error value

These conversions translate many different source operand tokens into tokens representing different classes of inputs, such as strings and numeric values that are required by functions. Functions typically operate on the converted tokens but can also refer to elements of the source token. See the Custom Input Types for details on common token conversions used by functions.

2.2.6.2.3.3.1 String Values

A string value is used to represent textual information and is persisted in the file as a PtgStr1. Other tokens can also represent a string value according to the conversion specified in the custom input type, vString.

2.2.6.2.3.3.2 Numeric Values

A numeric value represents a number with or without units.

A numeric value is persisted in the file as one of the tokens in the token group vNum (except PtgDate) or as a PtgUnsWord. Other token types can also represent numeric values as specified in the following custom input types: vDouble, vFloat, vSignedInt, vSignedLong, vUnsignedInt, vUnsignedLong.

Numeric values with units represent measured numbers. The units that are supported include length, angle, duration and typographic units.

All numeric values, with or without units, are expressed and persisted in the file in internal units as described in Custom Internal Unit Types. The internal unit depends on the type of measurement the numeric value represents.

If the numeric value represents a length or distance measurement, the value that is persisted in the file is expressed as a lengthInternalUnitNumber. The operand tokens that represent length or distance measurements are persisted in the file as one of these tokens: PtgNumCM, PtgNumDft, PtgNumF, PtgNumFI, PtgNumI, PtgNumKM, PtgNumM, PtgNumMI, PtgNumMM, PtgNumNM, PtgNumYards.

If the numeric value represents an angle measurement, the value that is persisted in the file is expressed as an angleInternalUnitNumber. The operand tokens that represent angles are specified in the vAngle custom token grouping.

If the numeric value represents a duration measurement, the value that is persisted in the file is expressed as a durationInternalUnitNumber. The operand tokens that represent durations are PtgEDay, PtgEHour, PtgEMin, PtgESec, PtgEWeek, and PtgTDurDft.

If the numeric value represents a length measurement used in typography, the value that is persisted in the file is expressed as a typographicInternalUnitNumber. The operand tokens that represent typographic measurements are PtgTypCD, PtgTypCi, PtgTypDft, PtgTypDi, PtgTypPi, PtgTypPP, and PtgTypPt.

A numeric value that represents a percentage value is persisted in the file as a PtgNumPct token.

A numeric value with units persisted in the file as a PtgPageDft token indicates that the internal units are determined by the default values of the drawing page, as specified by PtgPageDft.

If the numeric value has no units, it represents a number and is persisted in the file as a PtgNum or PtgUnsWord.

All numeric values can have a dimension. One-dimensional units are used to represent length, angle, duration and typographic measurements, and are represented as numeric values as described earlier. Two-dimensional units are used to represent area measurements and three-dimensional units are used to represent volume measurements. A numeric value that has a dimension greater than one is called a multidimensional number and is persisted in the file as a PtgNumMultiDim.

2.2.6.2.3.3.3 Boolean Values

A value that represents a **Boolean** value is persisted in the file as a PtgBool. Other tokens can also represent a Boolean value according to the conversion specified in the custom input type, vBoolean.

2.2.6.2.3.3.4 Currency Values

A value that represents a currency is persisted in the file as a PtgCy. No other token type can represent a currency value. The only custom input type that preserves both a currency and its associated currency ID is vDoubleEx.

2.2.6.2.3.3.5 Color Values

A value that represents a color value is persisted in the file as a PtgColorRGB. Other tokens can also represent a color value according to the conversion specified in the custom input type, vColor.

2.2.6.2.3.3.6 Date Values

A value that represents a date is persisted in the file as a PtgDate. Other token types can represent a date according to the conversion specified in the DateTime function.

2.2.6.2.3.3.7 Error Values

An error code that is returned as a result of a formula evaluation is persisted in the file as a PtgErr. When a function encounters an error value as one of its arguments, it returns the same error value. The exceptions are the functions IsErr, IsErrNA, IsError, and IsErrValue which are specifically designed to detect certain error values.

2.2.6.2.4 Evaluation Stack

The evaluation stack is used to keep track of intermediate stages and precedence of operations when evaluating a formula expression. The expression is evaluated from beginning to end, and operands are pushed onto the stack as they are encountered. When function tokens are encountered, the required number of operands is popped from the stack and the result of the operation is pushed back onto the stack. Control tokens are not pushed onto the stack. Evaluation begins with an empty stack, and when

the evaluation is finished, there is exactly one token left on the stack. The remaining token is the result of the evaluation.

2.2.6.3 Unit Number

A Unit Number is a numeric value with a unit of measure. The numeric value is a Custom Internal Unit Type or a value specified by PtgDate. The unit of measure determines how the numeric value is formatted and displayed in the user interface.

2.3 ShapeGraphic XML Part

This part specifies the graphical elements that define the visual contents of a drawing page in the document before diagram update is performed on it. These graphical elements MUST be expressed in Silverlight **XAML** as specified in [MS-SLXV] and MUST conform to the constraints defined in this section.

2.3.1 XAML Terminology

The following terms are used to specify constraints on the ShapeGraphic part in this document:

Element of type *T*, where *T* is **Canvas**, **Ellipse**, **Rectangle**, **Path**, **Image**, **TextBlock**, **Glyphs**, **MatrixTransform**, **SolidColorBrush**, **ImageBrush**, or **Run**: Refers to an XML element that can be processed into an Object Node as described in section 6.6.2 of [MS-XAML], the XamlType being *T* from the Silverlight Xaml Schema Information Set described in [MS-SLXV].

Element of type *T.M*, where *T* is **Canvas**, **Ellipse**, **Rectangle**, **Path, Glyphs**, or **Run** and *M* is **RenderTransform**, **Fill, Stroke**, **FontWeight**, or **FontStyle**: Refers to an XML element that can be processed into Member Nodes as described in section 6.6.5 of [MS-XAML], the XamlType being *T* and the XamlMember being *M* from the Silverlight Xaml Schema Information Set described in [MS-SLXV].

M property, where *M* is **Source**, **ImageSource**, **Fill**, **Visibility**, **Name**: Refers to either an XML child element that can be processed into a Member Node as described in section 6.6.5 of [MS-XAML], or an XML attribute that can be processed into a Member Node as described in section 6.6.3 of [MS-XAML], the XamlMember being *M* from the Silverlight Xaml Schema Information Set described in [MS-SLXV].

Path primitive : Refers to single-letter commands that are part of the **Data** attribute of an element of type **Path**, according to the parsing of that attribute specified in the **GeometrySyntax** Text Syntax Information Set in [MS-WPFXV].

2.3.2 XAML Resources

In the ShapeGraphic part, elements of type **Glyphs** refer to external **font** resources, and elements of type **Image** and **ImageBrush** refer to external image resources.

2.3.2.1 Fonts

All elements of type **Glyphs** in the ShapeGraphic part MUST have a **FontUri** attribute defined. If the **FontUri** attribute does not have a prefix of "CommonFonts/", then there MUST be a Fonts part whose name is equal to the value of this attribute. When the **FontUri** attribute does have a prefix of "CommonFonts/", the value of this attribute after that prefix, which MUST be a value from the following table, specifies the **font family** used to render the glyphs, as well as whether the glyphs use bold or italic formatting.

Value	Font family	Bold	Italic
96124B06-B06C-4D87-9DC8-D4BBF5D7B681.odttf	Arial	False	False

Value	Font family	Bold	Italic
C358B472-DE75-4EC3-B0D4-CD16217840DF.odttf	Arial	True	False
4FB139D2-73CE-4404-8857-257D1164B8E2.odttf	Arial	False	True
A1A273E7-72DC-40C0-95F0-8C9DD99A69A4.odttf	Arial	True	True
15BE407E-E656-49FB-BD0D-6956801B8457.odttf	Arial Black	False	False
80AAA2AC-A1E3-4D23-A23A-629E77B9D6E3.odttf	Arial Narrow	False	False
5B6F85F0-B9CA-4234-85D5-712569646CF5.odttf	Arial Narrow	True	False
222735B6-9799-46F7-8179-CDFA36EB9772.odttf	Arial Narrow	False	True
0F4CEC39-6EF4-494B-95C4-1C5393B64D7F.odttf	Arial Narrow	True	True
E94BD929-2AB4-4DCE-8733-ADA37D797930.odttf	Book Antiqua	False	False
FD94B59A-84D1-4A30-9DA0-609D670006A4.odttf	Book Antiqua	True	False
3EAB2CEC-411A-4FBB-B52F-C4DAA4564A22.odttf	Book Antiqua	False	True
DCE069D4-DCE4-4DCF-B653-E5C3062890C5.odttf	Book Antiqua	True	True
808543AF-7936-4CA8-B1B4-2A71EABA6571.odttf	Bookman Old Style	False	False
81AD0CBD-900F-4EF6-8905-865617916412.odttf	Bookman Old Style	True	False
E3117E24-05C9-4CDA-92E8-7EA58D9B1180.odttf	Bookman Old Style	False	True
DF0D362F-ECF6-41CC-B34E-A222E2F48765.odttf	Bookman Old Style	True	True
4C1AABE5-BA71-4F2C-8263-66731F995C4B.odttf	Calibri	False	False
2FCC3378-38EC-463F-BEC2-438FD506EF47.odttf	Calibri	True	False
EEA6899E-27A6-4A9A-BA90-C83F21189897.odttf	Calibri	False	True
C4D86DCF-C433-4630-B4F8-BC8EFB8A51E9.odttf	Calibri	True	True
6966055E-CCDF-4307-913C-BDFF248FD081.odttf	Cambria	False	False
43223052-5DE8-484B-8EAA-48F50A2CDFBC.odttf	Cambria	True	False
82B04200-E877-4DF3-9841-8091979F5F37.odttf	Cambria	False	True
F79EADC2-EC72-4D6D-8440-F3973033C747.odttf	Cambria	True	True
A3821346-4A5B-4182-8832-B5B2EF80A41B.odttf	Candara	False	False
09EDCD55-8604-4B5E-9618-352681491E13.odttf	Candara	True	False
E73B8D74-2F83-414F-A9B6-1CA67B3BD3E8.odttf	Candara	False	True
597038BC-C5F4-449E-8EFF-4F9460C0AC27.odttf	Candara	True	True
67BF9C01-7EE5-448D-B713-CCBAA6B227D7.odttf	Century	False	False
DCFBF993-0CAA-4C71-A58E-1CDD969BD4B8.odttf	Comic Sans MS	False	False
C67B43C3-A5C3-41B6-9C47-1D54FDD4FD6D.odttf	Comic Sans MS	True	False
865B4E77-E76B-4460-9ECB-E88E367B9E51.odttf	Consolas	False	False
CB370F8E-26AA-45D3-B0C1-E8AE76EE39D7.odttf	Consolas	True	False

Value	Font family	Bold	Italic
A86B1F02-B649-4AC1-AED6-F06B39A2427F.odttf	Consolas	False	True
BF5AEE82-A2A1-4AA1-A9B7-7DB1732D4AB3.odttf	Consolas	True	True
DA5A5D60-F7F2-4520-AEA3-442AFAC8B032.odttf	Corbel	False	False
24D25870-0E64-4226-A752-E0F9E41A580D.odttf	Corbel	True	False
EC4CF21B-B4A1-4D31-9CCD-EDB33D819FB8.odttf	Corbel	False	True
AAE9787A-E07D-49FB-9CF1-5B6EEC4AF38E.odttf	Corbel	True	True
D39AF8ED-AB87-4B59-B8C9-91CCE4C08A9D.odttf	Courier New	False	False
6B2A0BA7-1A7A-4751-9FB8-D06A2E0CDF15.odttf	Courier New	True	False
7B691A34-3EAA-4756-AC6F-615659B2C177.odttf	Courier New	False	True
00E33E22-B4B7-4B8A-A1A7-EA05026E9926.odttf	Courier New	True	True
B6243CDB-E343-4FF9-86F2-7FE6B48C19FC.odttf	Franklin Gothic Book	False	False
DEF36608-0031-4DF9-81DD-E63A93642DB7.odttf	Franklin Gothic Book	False	True
11B53828-77A2-4F10-BD06-56576BD2B2DD.odttf	Franklin Gothic Demi	False	False
13C1D5A6-C56C-4D8E-AD71-3838DC49FB26.odttf	Franklin Gothic Demi	False	True
AA42910A-F8C6-4293-8116-A4D6154E262C.odttf	Franklin Gothic Demi Cond	False	False
61025B6F-F0E9-407F-9524-B3AF18AD2577.odttf	Franklin Gothic Heavy	False	False
C436D513-1D30-4CE3-8A2A-E7A3F83DAAF1.odttf	Franklin Gothic Heavy	False	True
F369B7AE-EE88-4B34-ADEC-CB15AF8A6B36.odttf	Franklin Gothic Medium	False	False
F5138333-541D-4CC9-8E00-D341395C3278.odttf	Franklin Gothic Medium	False	True
61855DFC-5F03-498A-90C5-02D4DFCE359D.odttf	Franklin Gothic Medium Cond	False	False
3E3519FA-71B1-4F7B-8161-7AD77198FB2C.odttf	Garamond	False	False
8A7CF9AD-5C0B-423F-AD30-8CF1F6B29719.odttf	Garamond	True	False
F5109707-DE9D-416E-8535-C99238F13E8A.odttf	Garamond	False	True
C7E23E6B-DD21-4F21-A7D5-8ABBA7692910.odttf	Georgia	False	False
9F0D1069-4284-46EA-B336-7A8649D6BF55.odttf	Georgia	True	False
9F56D674-6626-4E6B-9223-D8C7AEC2D901.odttf	Georgia	False	True
9AFCD208-D49C-4CAE-9BE5-C256BFC488B0.odttf	Georgia	True	True
2F23B780-62D4-4384-928D-5A5733F6011D.odttf	Segoe UI	False	False
39943FD9-9366-4596-BE1C-2FFA67FFC4D8.odttf	Segoe UI	True	False
297C663E-9645-485B-B560-996CF4D7E544.odttf	Segoe UI	False	True
C0D43036-F10C-4CAE-8A01-C26DFEB3D46B.odttf	Segoe UI	True	True
47E24325-C88C-450B-B178-769A8BE88241.odttf	Symbol	False	False
8D1E89DD-FC5F-4230-8652-F172DA76E95E.odttf	Tahoma	False	False

Value	Font family	Bold	Italic
C1D99FC0-2154-43D4-9E21-CADA3D102904.odttf	Tahoma	True	False
D56DC39A-1D0F-4F8A-9F87-493EBD134E45.odttf	Times New Roman	False	False
CE46F0D4-CDA3-41DA-98F7-7AE756189318.odttf	Times New Roman	True	False
12FEEB78-5B96-44C3-9AF6-3237613F26C9.odttf	Times New Roman	False	True
EB2B113F-1CC3-462D-88E8-ACE2599DA9B4.odttf	Times New Roman	True	True
DF71F07D-0181-4119-B75E-F96878BD1D63.odttf	Trebuchet MS	False	False
4D44B836-2782-4D27-8AE0-DBEBAC456B46.odttf	Trebuchet MS	True	False
1BFDA688-1B13-4263-9F1F-FCFDD57A9F28.odttf	Trebuchet MS	False	True
54EA0ACE-735A-45C0-A667-55462535C685.odttf	Trebuchet MS	True	True
0536AE94-BE0F-4463-A07B-D5FE77A2D4B5.odttf	Verdana	False	False
6E52AB96-E48C-4AB7-9B8D-0E953F032BD4.odttf	Verdana	True	False
BA79D24C-3CB7-4AA7-BBAD-7A9040F8EF56.odttf	Verdana	False	True
8F4AA1FC-1A6E-48D9-AAC0-7C5FB1D78982.odttf	Verdana	True	True
22323AAC-91D5-4C5B-92BC-851531B77A51.odttf	Wingdings	False	False

Every **Glyphs** element whose **FontUri** attribute has a prefix of "CommonFont/" MUST have a **UnicodeString** attribute defined. For each value in the left-hand column of the following table, every **Glyphs** element whose **FontUri** is equal to that value MUST have a **UnicodeString** attribute composed of characters whose **Unicode code points** are in the corresponding cell in the right-hand column.

FontUri	Unicode code points MUST be
CommonFont/22323AAC-91D5-4C5B-92BC-851531B77A51.odttf	32, 74, 76, 108, 110, 113, 118, 158, 167, 178, 216, 223, 224, 232, 233, 234 or 252
CommonFont/47E24325-C88C-450B-B178-769A8BE88241.odttf	32, 43, 45, 61, 97, 98, 109, 152, 176, 180, 183, 190, 210, 211, 212 or 226
CommonFont/*V*, where *V* is another value	Greater than or equal to 32 and less than 127, or one of the following values: 163, 167, 169, 174, 189, 224, 225, 228, 232, 233, 243, 246, 252, 8211, 8212, 8216, 8217, 8220, 8221, 8226, 8230, 8482, and 8722

2.3.2.2 Images

All elements of type **Image** in the ShapeGraphic part MUST have a **Source** property defined, the value of which MUST be the name of an Images part.

All elements of type **ImageBrush** in the ShapeGraphic part MUST have an **ImageSource** property defined, the value of which MUST be the name of an Images part.

2.3.3 XAML Shapes

All graphical elements that correspond to shapes in the drawing page are specified as **XAML UIElement** objects and MUST conform to the Silverlight XAML syntax as specified in [MS-SLXV] section 1.22.31.

For each CT_ShapeInfo element in the ShapeInfo XML Part, there MUST be an element of type **Canvas** whose **Name** attribute is equal to the **Name** attribute of the CT_ShapeInfo. That **Canvas** element MUST have a **Tag** attribute defined with a non-empty value. The child elements of that **Canvas** element specify the appearance of the shape that corresponds to the CT_ShapeInfo.

2.3.4 XAML Recalculated Shapes

A **XAML** shape that that can be updated by formula expressions is called data graphic. There are five types of data graphics:

- Sheet element data graphic
- Formatting element data graphic
- Geometry element data graphic
- Text element data graphic
- Image element data graphic

The following sections specify the constraints that each type of data graphics MUST conform to so that it can be updated during diagram update.

2.3.4.1 Sheet Elements

A **sheet** element data graphic is a **XAML** element of type **Canvas** whose geometric positioning information can be updated by formula expressions. The **Name** attribute of this **Canvas** MUST be equal to the **Name** attribute of a CT_Sheet element in the DataGraphic XML Part.

2.3.4.1.1 Shape Transform

For each CT_Sheet element in the DataGraphic XML Part that includes **Angle**, **FlipX**, **FlipY**, **PinX**, **PinY**, **LocPinX** or **LocPinY** among its defined attributes, the corresponding sheet element data graphic MUST have a child **Canvas.RenderTransform** element that has a child element of type **MatrixTransform**. This **Canvas.RenderTransform** element is called a Shape Transform and specifies an affine transformation of the geometry elements included in the **sheet** element data graphic.

The measurement units for the **PinX, PinY, LocPinX, LocPinY** attributes are specified as length internal units. The measurement units for the **Angle** attribute are specified as angle internal units. The **FlipX** and **FlipY** attributes do not have measurement units.

The **MatrixTransform** used to apply these transforms to the **sheet** element data graphic is constructed by applying the transforms in the following order:

1. Translate by (-**LocPinX**, -**LocPinY**)
2. Mirror about the X axis if **FlipX** is equal to 1 and about the Y axis if **FlipY** is equal to 1
3. Rotate around origin by **Angle**
4. Translate by (**PinX**, **PinY**)

2.3.4.2 Formatting Elements

A formatting element data graphic is a sheet element data graphic with supplementary constraints that allow the visual formatting properties of the **Canvas** to be updated by formula expressions. Each CT_Sheet element associated with a formatting element data graphic MUST specify the **FillBackground**, **FillBackgroundTrans**, **FillForeground**, **FillForegroundTrans**, **FillPattern**, **LineColor**, **LineColorTrans** or **LineWeight** attributes or MUST have one or more child elements of type CT_Geometry.

A formatting element data graphic MUST contain child elements of type **Path**, **Ellipse**, **Rectangle**, **Canvas**, **TextBlock** or **Canvas.RenderTransform**.

If a formatting element data graphic has a child element of type **Path**, then it MUST NOT have a child element of type **Rectangle**, and it MUST NOT have a child element of type **Ellipse** that is not a Model Ellipse.

If a formatting element data graphic has a child element of type **Rectangle**, then it MUST NOT have a child element of type **Path** or **Ellipse**.

2.3.4.2.1 Shadow Canvas

A child **Canvas** element of a formatting element data graphic with a **Name** value equal to the string "Shdw" is a Shadow Canvas. It MUST contain only child elements of types **Path**, **Ellipse**, **Rectangle** or **Canvas.RenderTransform**.

The **Path**, **Rectangle** or **Ellipse** elements that are child elements of a Shadow Canvas specify the appearance and the relative position of the shadow of the formatting element data graphic. A Shadow Canvas is recalculated only according to the formulas in the **LineWeight** attribute and the **Geometry** child elements of a CT_Sheet.

All **Path**, **Rectangle** or **Ellipse** elements that are part of a formatting element data graphic MUST be either child elements of the formatting element data graphic, or child elements of the corresponding Shadow Canvas.

2.3.4.2.2 Fill Attributes

For each CT_Sheet element in the DataGraphic XML Part that includes **FillBackground**, **FillBackgroundTrans**, **FillForeground**, **FillForegroundTrans**, or **FillPattern** among its defined attributes, all child **Path**, **Ellipse** or **Rectangle** elements of the corresponding formatting element data graphic MUST NOT have a **Fill** attribute. Any of these elements that have a **Fill** property defined MUST have it defined as a child **Path.Fill**, **Ellipse.Fill** or **Rectangle.Fill** element.

2.3.4.2.3 Stroke Attributes

For each CT_Sheet element in the DataGraphic XML Part that has a **LineColor** or **LineColorTrans** attribute, all child **Path**, **Ellipse** or **Rectangle** elements of the corresponding formatting element data graphic MUST NOT have a **Stroke** attribute. Any of these elements that have a **Stroke** property defined MUST have it defined as a child **Path.Stroke**, **Ellipse.Stroke** or **Rectangle.Stroke** element that has a child **SolidColorBrush** element.

2.3.4.3 Geometry Elements

A geometry element data graphic is a child **Canvas** of a sheet element data graphic whose geometry can be updated by formula expressions. The **Name** attribute of this child **Canvas** MUST contain the string "_G*n*_", where *n* MUST be decimal number and where *n* MUST equal the value of one of the **Index** attributes of a CT_Geometry element contained within the CT_Sheet associated with the sheet element data graphic.

For each CT_Geometry element in the DataGraphic XML Part, the corresponding sheet element data graphic MUST contain at least one corresponding geometry element data graphic.

2.3.4.3.1 Geometry Path Specifications

For a CT_Geometry element in the DataGraphic XML Part with a **BoundPts** attribute defined, all the corresponding geometry element data graphics that are either Model Elements, or such that there is no Model Element associated to the same CT_Geometry, MUST be of type **Path** or of type **Ellipse**. When they are of type **Path**, they MUST have a **Data** attribute that specifies the path of the geometry. The value of any of these **Data** attributes MUST contain a number of path primitives that is greater than or equal to the **pointindex1** of the corresponding CT_Geometry **BoundPts** attribute. These path primitives are indexed in the order of their appearance in the **Data** attribute, the first one having the index 1.

For each CT_Geometry element in the DataGraphic XML Part whose **BoundPts** attribute has two indices, the corresponding path primitive in each geometry element data graphic MUST be of type **Point** as specified in [MS-SLXV] section 1.62 or **LineSegment** as specified in [MS-SLXV] section 1.22.19.3.

For each CT_Geometry element in the DataGraphic XML Part whose **BoundPts** attribute has three indices, the corresponding path primitive in each geometry element data graphic MUST be of type **ArcSegment** as specified in [MS-SLXV] section 1.22.19.1**.**

2.3.4.4 Model Elements

Model Elements are invisible **XAML** elements that are intended for informational purposes.

2.3.4.4.1 Model Paths

A **Path** element that is a child of a sheet element data graphic and whose **Name** attribute contains the string "_model" is a Model Path.

A Model Path specifies a geometry element data graphic, without any corner rounding applied.

A Model Path is rendered after a corner rounding radius is applied to each corner when diagram update is performed.

A Model Path MUST have a **Visibility** property defined as "Collapsed".

A Model Path MUST have a **Tag** attribute with an xsd:double ([XMLSCHEMA2] section 3.2.5) value, which specifies the corner rounding radius to be applied to the path before diagram update.

For a CT_Sheet element in the DataGraphic XML Part that has a **Rounding** attribute whose value is different from "0:65" or "0:98", the corresponding sheet element data graphic MUST contain a Model Path that is specific to each child CT_Geometry element of that CT_Sheet element.

2.3.4.4.2 Model Ellipses

An **Ellipse** element that is a child of a sheet element data graphic and whose **Name** attribute contains the string "_model" is a Model Ellipse.

A Model Ellipse MUST have a **Visibility** attribute set to "Collapsed".

A Model Ellipse specifies an ellipse that is rendered after the position of its center and its major and minor axis are recalculated according to the formulas in the corresponding CT_Geometry elements of the DataGraphic XML Part.

2.3.4.5 Text Elements

A text element data graphic is a **XAML** element of type **Canvas** whose text can be updated by formula expressions. The **Name** attribute of this **Canvas** MUST equal the value of a **Name** attribute of a CT_ShapeText element in a ShapeTextBinding XML Part.

2.3.4.5.1 Glyph Canvas

A child **Canvas** element of a text element data graphic with no Name property defined is the Glyph Canvas corresponding to the text.

Each child of a Glyph Canvas MUST be of type **Glyphs** or **Canvas.RenderTransform**, and specify the appearance, position and color of the text before diagram update.

A child **Glyphs** element of a Glyph Canvas MUST have a child **Glyphs.Fill** element. That element MUST have a child **SolidColorBrush** element.

A Glyph Canvas can have transforms applied to it as specified by the **TextAngle**, **TextPinX**, **TextPinY**, **TextLocPinX** or **TextLocPinY** attributes of the corresponding CT_Sheet element in the DataGraphic XML Part.

The measurement units for the **TextPinX, TextPinY, TextLocPinX, TextLocPinY** attributes are specified as length internal units. The measurement units for the **TextAngle** attribute are specified as angle internal units.

The **RenderTransform** matrix is constructed by applying the transforms in the following order:

1. Translate by (-**TextLocPinX**, -**TextLocPinY**)
2. Rotate around origin by **TextAngle**
3. Translate by (**TextPinX**, **TextPinY**)

2.3.4.5.2 Text Model

A text element data graphic MUST have a child **TextBlock** element, and that element MUST have exactly one child **Run** element. That **TextBlock** element is the Text Model.

The Text Model specifies the value and typographical attributes of the text before diagram update is performed.

A Text Model MUST have a **Visibility** property set to the value "Collapsed".

A Text Model MUST have the following attributes defined: **Width**, **Height** and **Tag**.

The value of the **Tag** attribute of a Text Model specifies additional positioning information for the text within the text element data graphic. It MUST conform to the following syntax:

Syntax:

```
horizontal_alignment;vertical_alignment;rtl;line_spacing;margin_left;margin_right;margin_top;margin_bottom
```

horizontal_alignment specifies how the text is horizontally located within the Glyph Canvas and MUST be a value from the following table:

Value	Meaning
Left	The text lines are flushed to the left of the text block.
Center	Each text line is centered in the text block.

Value	Meaning
Right	The text lines are flushed to the right of the text block.

vertical_alignment specifies how the text is vertically located within the Glyph Canvas and MUST be a value from the following table:

Value	Meaning
Top	The text lines are flushed to the top of the text block.
Center	The text lines are centered in the text block.
Bottom	The text lines are flushed to the bottom of the text block.

rtl is unused and MUST be ignored.

line_spacing is an xsd:double ([XMLSCHEMA2] section 3.2.5) that specifies the distance between lines, as a factor of the value of the **FontSize** attribute in the child **Run** element.

margin_left is an xsd:double ([XMLSCHEMA2] section 3.2.5) that specifies the distance in inches between the left edge of the glyph canvas and the left edge of the space occupied by the text glyphs.

margin_right is an xsd:double ([XMLSCHEMA2] section 3.2.5) that specifies the distance in inches between the right edge of the glyph canvas and the right edge of the space occupied by the text glyphs.

margin_top is an xsd:double ([XMLSCHEMA2] section 3.2.5) that specifies the distance in inches between the top edge of the glyph canvas and the top edge of the space occupied by the text glyphs.

margin_bottom is an xsd:double ([XMLSCHEMA2] section 3.2.5) that specifies the distance in inches between the bottom edge of the glyph canvas and the bottom edge of the space occupied by the text glyphs.

2.3.4.5.3 Text Run

A child **Run** element of a Text Model is called a Text Run.

The **Name** attribute of each Text Run MUST be equal to the **Name** attribute of the CT_TextRun child of the corresponding CT_ShapeText element.

Each Text Run MUST have the attributes **FontFamily** and **FontSize** defined.

Each Text Run MUST NOT have a **Run.FontWeight** or **Run.FontStyle** child element. If the **FontWeight** or **FontStyle** properties are defined, they MUST be defined as attributes.

2.3.4.6 Image Elements

An image element data graphic is a **XAML** element of type **Canvas** that contains one or more **Image** elements whose source image can be updated by formula expressions. The **Name** attribute of this **Canvas** MUST equal the value of a **DefShapeID** attribute of a CT_DataGraphicDef element in the DataGraphic XML part.

Each image element data graphic MUST have at least one child element of type **Image**.

2.4 XML Parts

2.4.1 Introduction

The following sections specify the structure of the XML parts that are in the ZIP archive of the "VisioServerData" stream.

2.4.2 App XML Part

This part specifies the properties of the document. App Schema, see section 5.1.

2.4.3 Core XML Part

An optional part that specifies additional properties of the document.

2.4.4 DataBinding XML Part

This part specifies the data binding information that is used to link external data to shapes in drawing pages of the document.

2.4.4.1 Global Elements

2.4.4.1.1 BindingConnections

A CT_BindingConnections element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="BindingConnections" type="CT_BindingConnections"/>
```

2.4.4.2 Complex Types

2.4.4.2.1 CT_PrimaryKeyValue

Referenced by: CT_PrimaryKeyValues

The **CT_PrimaryKeyValue** complex type specifies the value of one component of the **primary key** in a recordset.

Attributes:

Value : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the value of a **field** that is a component of the primary key.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_PrimaryKeyValue">
  <xsd:attribute name="Value" type="xsd:string"/>
</xsd:complexType>
```

2.4.4.2.2 CT_PrimaryKeyValues

Referenced by: CT_Binding

The **CT_PrimaryKeyValues** complex type specifies the values of all the components of the **primary key** of the **row** that is bound to a shape.

Child Elements:

PrimaryKey : A list of CT_PrimaryKeyValue elements. The count and the order of the CT_PrimaryKeyValue elements in this list MUST match the **PrimaryKeys.PrimaryKey** child elements of its containing CT_DataRecordset, whose **ID** attribute is equal to the **RecordsetID** attribute of this element's containing CT_BindingConnection.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_PrimaryKeyValues">
  <xsd:sequence>
    <xsd:element name="PrimaryKey" minOccurs="1" maxOccurs="unbounded"
type="CT_PrimaryKeyValue"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.4.2.3 CT_Binding

Referenced by: CT_Bindings

The **CT_Binding** complex type specifies a recordset **row** that is bound to a shape.

Child Elements:

PrimaryKeys : An optional CT_PrimaryKeyValues element which specifies the recordset row that is bound to a shape addressed using the primary key method. If the **Row** attribute exists, this element MUST NOT exist. If the **Row** attribute does not exist, this element MUST exist.

Attributes:

BindingID : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the identifier for this binding. It MUST be unique amongst all **BindingID** attributes in the document.

Row : An optional xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the one-based row number used to address a row using the row-order method. If the **PrimaryKeys** element exists, this attribute MUST NOT exist. If the **PrimaryKeys** element does not exist, this attribute MUST exist.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_Binding">
  <xsd:sequence>
    <xsd:element name="PrimaryKeys" minOccurs="0" maxOccurs="1" type="CT_PrimaryKeyValues"/>
  </xsd:sequence>
  <xsd:attribute name="BindingID" type="xsd:integer"/>
  <xsd:attribute name="Row" type="xsd:integer" use="optional"/>
</xsd:complexType>
```

2.4.4.2.4 CT_Bindings

Referenced by: CT_BindingConnection

The **CT_Bindings** complex type specifies all the **rows** in a recordset that are bound to shapes.

Child Elements:

Binding : A list of CT_Binding elements.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_Bindings">
  <xsd:sequence>
    <xsd:element name="Binding" minOccurs="1" maxOccurs="unbounded" type="CT Binding"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.4.2.5 CT_BindingConnection

Referenced by: CT_BindingConnections

The **CT_BindingConnection** complex type specifies a recordset that has **rows** of data that are bound to shapes.

Child Elements:

Bindings : A CT_Bindings element.

Attributes:

RecordsetID : An xsd:unsignedLong ([XMLSCHEMA2] section 3.3.21) attribute that specifies the identifier of the recordset. It MUST match an **ID** attribute of a CT_DataRecordset element in the DataConnection XML part.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_BindingConnection">
  <xsd:sequence>
    <xsd:element name="Bindings" minOccurs="1" maxOccurs="1" type="CT_Bindings"/>
  </xsd:sequence>
  <xsd:attribute name="RecordsetID" type="xsd:unsignedLong"/>
</xsd:complexType>
```

2.4.4.2.6 CT_BindingConnections

Referenced by: BindingConnections

The **CT_BindingConnections** complex type specifies all the recordsets in the document that have **rows** of data that are bound to shapes.

Child Elements:

BindingConnection : An optional list of CT_BindingConnection elements. CT_BindingConnection elements are not present if the shapes are not bound to data from recordsets.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT BindingConnections">
  <xsd:sequence>
    <xsd:element name="BindingConnection" minOccurs="0" maxOccurs="unbounded"
type="CT_BindingConnection"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.5 DataConnection XML Part

This part specifies the data connection information needed to **query data sources** and refresh the recordsets referenced by a Web drawing.

2.4.5.1 Global Elements

2.4.5.1.1 Connections

A CT_Connections element.

```
<xsd:element name="Connections" type="CT_Connections"/>
```

2.4.5.2 Complex Types

2.4.5.2.1 CT_DataConnection

Referenced by: CT_DataConnections

The **CT_DataConnection** complex type specifies a data connection.

Attributes:

ID : An xsd:unsignedLong ([XMLSCHEMA2] section 3.3.21) attribute that specifies the identifier of a **CT_DataConnection**. It MUST be unique amongst all the **DataConnection** child elements of the containing CT_DataConnections.

ConnectionString : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the **connection string** to a **data source**. It MUST be a valid connection string as specified in [MS-ODBCSTR]. The type of the data source specified in the connection string MUST be from the following table.

Type of data source	Syntax
OLE DB	MUST be a valid connection string as specified in [MS-ODBCSTR]
ODBC	MUST be a valid connection string as specified in [MS-ODBCSTR]
List	MUST be in the following format: "PROVIDER=WSS;DATABASE=*list URL*;LIST={*list GUID*};*viewparam*" ▪ *list URL* is the **URL** of a list ▪ *list GUID* is the **GUID** of the list ▪ *viewparam* MUST be in the format "VIEW={*view GUID*};" if the data source is a **view** of a list, where *view GUID* is the GUID of the view; otherwise, *viewparam MUST* be an empty string
Workbook	MUST be in the following format: "DataModule=Microsoft.Office.Visio.Server.EcsDataHandler,Microsoft.Office.Visio.Server; Data Source=*workbook URL*;Extended Properties='HDR=*hdrvalue*;';"

	▪ *hdrvalue* equals YES if the first **row** of data is the **header row**, and NO otherwise ▪ *workbook URL* is the URL of a workbook
Custom	MUST be in the following format: "DataModule=*class name,assembly name*;Add-in key=*value pairs*;" ▪ *class name* is a **class name** ▪ *assembly name* is an **assembly name** ▪ *value pairs* is a semicolon-separated set of key-value pairs specific to the **add-in**

Filename : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the full path of an **ODC file**. If the **ConnectionString** attribute exists, this attribute MUST NOT exist. If the **ConnectionString** attribute does not exist, this attribute MUST exist.

```
<xsd:complexType name="CT_DataConnection">
  <xsd:attribute name="ID" type="xsd:unsignedLong" use="required"/>
  <xsd:attribute name="ConnectionString" type="xsd:string" use="optional"/>
  <xsd:attribute name="Filename" type="xsd:string" use="optional"/>
</xsd:complexType>
```

2.4.5.2.2 CT_DataConnections

Referenced by: CT_Connections

The **CT_DataConnections** complex type specifies all the data connections of the document.

Child Elements:

DataConnection : An optional list of CT_DataConnection elements. CT_DataConnection elements are not present if the document does not have data connections.

```
<xsd:complexType name="CT_DataConnections">
  <xsd:sequence>
    <xsd:element name="DataConnection" minOccurs="0" maxOccurs="unbounded"
type="CT_DataConnection"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.5.2.3 CT_PrimaryKey

Referenced by: CT_PrimaryKeys

The **CT_PrimaryKey** complex type specifies a component of a **primary key**.

Attributes:

Key : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the name of a **field** that is a component of a primary key. It MUST be a value from the **Name** attribute of a **DataColumn** child element of the CT_DataColumns of a CT_DataRecordset whose primary key is being specified.

```
<xsd:complexType name="CT_PrimaryKey">
  <xsd:attribute name="Key" type="xsd:string" use="required"/>
```

```
</xsd:complexType>
```

2.4.5.2.4 CT_PrimaryKeys

Referenced by: CT_DataRecordset

The **CT_PrimaryKeys** complex type specifies all the components of the **primary key** of a recordset.

Child Elements:

PrimaryKey : A list of CT_PrimaryKey elements. The number of elements in this list MUST NOT exceed the number of **DataColumn** child elements of the CT_DataColumns of a CT_DataRecordset whose primary key is being specified.

```
<xsd:complexType name="CT_PrimaryKeys">
  <xsd:sequence>
    <xsd:element name="PrimaryKey" minOccurs="1" maxOccurs="unbounded" type="CT_PrimaryKey"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.5.2.5 CT_DataColumn

Referenced by: CT_DataColumns

The **CT_DataColumn** complex type specifies a **field** in a recordset.

Attributes:

Name : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the identifier of the field.

DisplayName : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the display name of the field.

Type : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the **ADO data type** of the field. It MUST be a value from the standard ADO data types as listed in following table. For more information about these data types, see [MSDN-OLEDB].

Value	Meaning
0	Empty
2	Small Integer
3	Integer
4	Single Precision Floating Point
5	Double Precision Floating Point
6	Currency
7	Date

8	BSTR
9	IDispatch
10	Error
11	**Boolean**
12	Variant
13	IUnknown
14	Decimal
16	Tiny Integer
17	Unsigned Tiny Integer
18	Unsigned Small Integer
19	Unsigned Integer
20	Big Integer
21	Unsigned Big Integer
64	File Time
72	**GUID**
128	Binary
129	Character
130	Wide Character
131	Numeric
132	User Defined Type
133	Database Date
134	Database Time

135	Database Time Stamp
136	Chapter
138	Property Variant
139	Variable Number
200	Variable Character
201	Long Variable Character
202	Wide Variable Character
203	Long Wide Variable Character
204	Variable Binary Blob
205	Long Variable Binary Blob
8192	Array

```
<xsd:complexType name="CT_DataColumn">
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="DisplayName" type="xsd:string" use="required"/>
  <xsd:attribute name="Type" type="xsd:integer" use="required"/>
</xsd:complexType>
```

2.4.5.2.6 CT_DataColumns

Referenced by: CT_DataRecordset

The **CT_DataColumns** complex type specifies the **fields** of a recordset.

Child Elements:

DataColumn : A list of CT_DataColumn elements. The **DisplayName** attribute of each **DataColumn** in this list MUST be unique. The **Name** attribute of each **DataColumn** in this list MUST be unique.

```
<xsd:complexType name="CT_DataColumns">
  <xsd:sequence>
    <xsd:element name="DataColumn" minOccurs="1" maxOccurs="unbounded" type="CT_DataColumn"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.5.2.7 CT_DataRecordset

Referenced by: CT_DataRecordsets

The **CT_DataRecordset** complex type specifies a recordset.

Child Elements:

PrimaryKeys : An optional CT_PrimaryKeys element. If the **RowOrder** attribute does not exist or equals 0, this element MUST exist. If the **RowOrder** attribute exists and is equal to 1, this element MUST NOT exist.

DataColumns : A CT_DataColumns element.

Attributes:

ID : An xsd:unsignedLong ([XMLSCHEMA2] section 3.3.21) attribute that specifies the identifier of this recordset. It MUST be unique amongst all the **DataRecordset** child elements of the containing CT_DataRecordsets.

ConnectionID : An xsd:unsignedLong ([XMLSCHEMA2] section 3.3.21) attribute that specifies the identifier of a CT_DataConnection in the document.

Command : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the **query** string that returns this recordset from a **data source**. It MUST NOT be empty and MUST use syntax from following table:

Type of data source	Syntax
OLE DB	A valid **SQL** query string as specified in [MS-ODBCSTR]
ODBC	A valid SQL query string as specified in [MS-ODBCSTR]
List	A valid SQL query string as specified in [MS-ODBCSTR]. MUST NOT contain a WHERE clause.
Workbook	The query string MUST be "SheetName=*sheet name*;RangeName=*range name*;", where *sheet name* is the name of a **sheet**, and *range name* is the name of a **range**, given in **A1 reference style**.
Custom	MUST be same as the **ConnectionString** attribute of the CT_DataConnection that this recordset uses.

Pages : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a comma-separated list of indices of the drawing pages that contain shapes bound to data in this recordset. The index of each drawing page in this list MUST match the **ID** attribute of one of the **PageMetaData** child elements of the CT_PagesMetaData in the ShapeInfo XML Part.

RowOrder : An optional xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies whether this recordset uses the **row** number as the **primary key** to bind rows of data in this recordset to shapes in a drawing page. It MUST be 1 or 0. A value of 1 specifies that the row number is used. If the optional **PrimaryKeys** element exists, this attribute MUST NOT exist or MUST be equal to 0. If the **PrimaryKeys** element does not exist, this attribute MUST exist and MUST be equal to 1.

```
<xsd:complexType name="CT_DataRecordset">
  <xsd:sequence>
    <xsd:element name="PrimaryKeys" minOccurs="0" maxOccurs="1" type="CT_PrimaryKeys"/>
    <xsd:element name="DataColumns" minOccurs="1" maxOccurs="1" type="CT_DataColumns"/>
  </xsd:sequence>
```

```
  <xsd:attribute name="ID" type="xsd:unsignedLong" use="required"/>
  <xsd:attribute name="ConnectionID" type="xsd:unsignedLong" use="required"/>
  <xsd:attribute name="Command" type="xsd:string" use="required"/>
  <xsd:attribute name="Pages" type="xsd:string" use="required"/>
  <xsd:attribute name="RowOrder" type="xsd:integer" use="optional"/>
</xsd:complexType>
```

2.4.5.2.8 CT_DataRecordsets

Referenced by: CT_Connections

The **CT_DataRecordsets** complex type specifies all the recordsets of the document.

Child Elements:

DataRecordset : An optional list of CT_DataRecordset elements. CT_DataRecordset elements are not present if the document does not have recordsets.

```
<xsd:complexType name="CT_DataRecordsets">
  <xsd:sequence>
    <xsd:element name="DataRecordset" minOccurs="0" maxOccurs="unbounded"
type="CT_DataRecordset"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.5.2.9 CT_Connections

Referenced by: Connections

The **CT_Connections** complex type specifies all the external **data sources** in the document.

Child Elements:

DataConnections : A CT_DataConnections element.

DataRecordsets : A CT_DataRecordsets element.

```
<xsd:complexType name="CT_Connections">
  <xsd:sequence>
    <xsd:element name="DataConnections" minOccurs="1" maxOccurs="1"
type="CT_DataConnections"/>
    <xsd:element name="DataRecordsets" minOccurs="1" maxOccurs="1" type="CT_DataRecordsets"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.6 DataGraphic XML Part

This part specifies information and formula expressions used to update the data graphics in a Web drawing.

2.4.6.1 Global Elements

2.4.6.1.1 DataGraphics

A CT_DataGraphics element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="DataGraphics" type="CT_DataGraphics"/>
```

2.4.6.2 Complex Types

2.4.6.2.1 CT_DataGraphicDef

Referenced by: CT_DataGraphicDefinitions

The **CT_DataGraphicDef** complex type specifies a reference to information stored in the DataGraphicDefinition XML Part used to update an image element data graphic during diagram update.

Attributes:

DefID : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the CT_IconSetDef. It MUST be the value of the **DefID** attribute of a CT_IconSetDef element in the DataGraphicDefinition XML Part. It MUST NOT be an empty string.

DefShapeID : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the name of the image element data graphic that contains the **Image** elements referencing the **embedded images** corresponding to the CT_IconSetDef element specified by **DefID**. It MUST NOT be an empty string.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_DataGraphicDef">
  <xsd:attribute name="DefID" type="xsd:string" use="required"/>
  <xsd:attribute name="DefShapeID" type="xsd:string" use="required"/>
</xsd:complexType>
```

2.4.6.2.2 CT_DataGraphicDefinitions

Referenced by: CT_DataGraphics_Shape

The **CT_DataGraphicDefinitions** complex type specifies all the references to information stored in the DataGraphicDefinition XML Part used to update all the image element data graphics in the drawing page during diagram update.

Child Elements:

DataGraphicDef : A list of CT_DataGraphicDef elements.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_DataGraphicDefinitions">
  <xsd:sequence>
    <xsd:element name="DataGraphicDef" minOccurs="1" maxOccurs="unbounded"
type="CT_DataGraphicDef"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.6.2.3 CT_Geometry

Referenced by: CT_Sheet

The **CT_Geometry** complex type specifies how to update a geometry element data graphic. It MUST exist in a containing CT_Sheet element that has a corresponding sheet element data graphic in the ShapeGraphic part of this document.

Attributes:

Index : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the index of the **XAML** Path element in the corresponding sheet element data graphic.

BoundPts : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the point indices in the XAML Path element that is recalculated. It MUST be ignored when the **Attribute** attribute exists. When the **Attribute** attribute does not exist, It MUST be in the following format:

Format	Usage
pointindex1,*pointindex2*	For **StartPoint** or **LineSegment** path primitives, as specified in the geometry path specifications of the associated geometry element data graphic.
pointindex1,*pointindex2*,*EFlag*	For **ArcSegment** path primitives, as specified in the geometry path specifications of the associated geometry element data graphic.

Where *pointindex1*, *pointindex2* and *EFlag* are specified in the following table:

Value	Description
pointindex1	Index of the primitive in the path data, as specified in the geometry path specifications of the associated geometry element data graphic.
pointindex2	X or Y coordinate. MUST be 0 for the x-coordinate and 1 for the y-coordinate.
EFlag	A string that specifies that this path data primitive is an **ArcSegment**, as specified in the geometry path specifications of the associated geometry element data graphic. MUST be "E" for Ellipse or "A" for Elliptical Arc.

Attribute : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the property of the XAML Path element that the **Formula** attribute is updating. It MUST be ignored when the **BoundPts** attribute exists. When the **BoundPts** attribute does not exist, it MUST be a string from the following table:

Value	Description
NoShowGeom	Updates the **Visibility** property of the XAML Path element. If this string exists, MUST NOT show the geometry this element specifies.
NoLineGeom	Updates the opacity of the **Stroke** brush of the XAML Path element. If this string exists, the opacity value MUST be set to 0.0.
NoFillGeom	Updates the opacity of the **Fill** brush of the XAML Path element. If this string exists, MUST NOT fill the geometry this element specifies.

Formula : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the XAML Path element. The result token value of the formula evaluation is converted to a string.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_Geometry">
  <xsd:attribute name="Index" type="xsd:integer" use="required"/>
  <xsd:attribute name="BoundPts" type="xsd:string" use="optional"/>
  <xsd:attribute name="Attribute" type="xsd:string" use="required"/>
  <xsd:attribute name="Formula" type="xsd:string" use="required"/>
</xsd:complexType>
```

2.4.6.2.4 CT_Sheet

Referenced by: CT_Formulas

The **CT_Sheet** complex type specifies information about the formula expressions used to update sheet element, text element and formatting element data graphics during diagram update.

When any attribute among the following is specified: **LocPinX**, **LocPinY**, **PinX** or **PinY**, then all of the other attributes of that list MUST be specified, to allow the shape transform of a sheet element data graphic to be fully specified.

When any attribute among the following is specified: **TextLocPinX**, **TextLocPinY**, **TextPinX** or **TextPinY** then all of the other attributes of that list MUST be specified, to allow the **Render transform** of the glyph canvas to be fully specified.

When any attribute among the following is specified: **FillBackground** or **FillForeground** then all of the other attributes of that list MUST be specified.

Child Elements:

Geometry : An optional list of CT_Geometry elements. CT_Geometry elements exist only if there are geometry element data graphics that correspond to the containing CT_Sheet element.

Attributes:

Name : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the name of the corresponding sheet element. It MUST be a value from the **Name** attribute of a sheet element in the ShapeGraphic part.

Angle : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **RenderTransform** property of the corresponding sheet element data graphic. The default value is `"0"`. The canvas is applied as a rotation around the origin, after the (-LocPinX, -LocPinY) translation is applied.

FlipX : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **RenderTransform** property of the corresponding sheet element data graphic. The default value is `"0"`. The canvas is applied as a flip transform along the X axis after the rotation is applied.

FlipY : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **RenderTransform** property of the corresponding sheet element data graphic. The default value is `"0"`. The canvas is applied as a flip transform along the Y axis after the rotation is applied.

LocPinX : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **RenderTransform** property of the corresponding sheet element data graphic. The default value is `"0"`. LocPinX and **LocPinY** translation is applied to the canvas before all other transforms, by (-LocPinX, -LocPinY).

LocPinY : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **RenderTransfrom** property of the corresponding sheet element data graphic. The default value is `"0"`.

PinX : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **RenderTransform** property of the corresponding sheet element data graphic. The default value is "0". PinX and **PinY** translation is applied to the canvas after all other transforms, by (PinX, PinY). For more information, see section 2.3.4.1.1.

PinY : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **RenderTransform** property of the corresponding sheet element data graphic. The default value is "0".

HideText : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **Visibility** property of the text contained in the corresponding text element data graphic. If this attribute does not exist then the text visibility is not recalculated.

FillBackground : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the background color which is part of the **Fill** property in the corresponding formatting element data graphic. If this attribute does not exist then the background color is not recalculated.

FillBackgroundTrans : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the transparency of the background color which is part of the **Fill** property in the corresponding formatting element data graphic. If this attribute does not exist then the transparency of the background color is not recalculated.

FillForeground : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the foreground color which is part of the **Fill** property in the corresponding formatting element data graphic. If this attribute does not exist then the foreground color is not recalculated.

FillForegroundTrans : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the transparency of the foreground color which is part of the **Fill** property in the corresponding formatting element data graphic. If this attribute does not exist then the transparency of the foreground color is not recalculated.

FillPattern : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **FillPattern** property in the corresponding formatting element data graphic. The result of the formula evaluation for this attribute MUST be an integer value as specified in the following table. If this attribute does not exist, the fill pattern defaults to 1.

Value	Description
Less than 0	MUST be ignored and defaults to 1.
0	No pattern
1	Solid pattern
Greater than 1 and less than 25	MUST be ignored and defaults to 1.
25	Gradient – left to right horizontal
26	Gradient – center out radial
27	Gradient - right to left horizontal
28	Gradient - top to bottom vertical
29	Gradient - center out vertical
30	Gradient - bottom to top vertical

Value	Description
31	Gradient - top left rectangle
32	Gradient - top right rectangle
33	Gradient - bottom left rectangle
34	Gradient - bottom right rectangle
35	Gradient - center out rectangle
36	Gradient - top left radial
37	Gradient - top right radial
38	Gradient - bottom left radial
39	Gradient - bottom right radial
40	Gradient - center out radial
Greater than 40	MUST be ignored and defaults to 0.

LineColor : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **Stroke** property in the corresponding formatting element data graphic. If this attribute does not exist then the line color is not recalculated.

LineColorTrans : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **Stroke** property in the corresponding formatting element data graphic. If this attribute does not exist then the transparency of the line color is not recalculated.

LineWeight : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the StrokeThickness property in the corresponding formatting element data graphic, as specified in section 1.22.31.1.11 of [MS-SLXV]. If this attribute does not exist then the line weight is not recalculated.

TextWidth : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **Canvas.RenderTransform** property in the corresponding glyph canvas in the text element data graphic. The default value is `"0"`.

TextHeight : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **Canvas.RenderTransform** property in the corresponding glyph canvas in the text element data graphic. The default value is `"0"`.

TextAngle : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **Glyphs.RenderTransform** property of the corresponding glyph canvas in the text element data graphic. The default value is `"0"`. The canvas is applied a rotation around the origin, after the (-TextLocPinX, -TextLocPinY) translation is applied.

TextLocPinX : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **Glyphs.RenderTransform** property in the corresponding glyph canvas in the text element data graphic. The default value is `"0"`. TextLocPinX and TextLocPinY translation is applied to the canvas before all other transforms, by (-TextLocPinX, -TextLocPinY).

TextLocPinY : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **Glyphs.RenderTransform** property in the corresponding glyph canvas in the text element data graphic. The default value is `"0"`.

TextPinX : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **Glyphs.RenderTransform** property in the corresponding glyph canvas in the text element data graphic. The default value is "0". TextPinX and **TextPinY** translation is applied to the canvas after all other transforms, by (TextPinX, TextPinY).

TextPinY : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the **Glyphs.RenderTransform** property in the corresponding glyph canvas in the text element data graphic. The default value is "0".

Rounding : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the corner rounding radius of the **Data** attribute in the **Path** element in the sheet element data graphic. If this attribute does not exist then the corner rounding is not recalculated.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_Sheet">
  <xsd:sequence>
    <xsd:element name="Geometry" minOccurs="0" maxOccurs="unbounded" type="CT_Geometry"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string"/>
  <xsd:attribute name="Angle" type="xsd:string" use="optional"/>
  <xsd:attribute name="FlipX" type="xsd:string" use="optional"/>
  <xsd:attribute name="FlipY" type="xsd:string" use="optional"/>
  <xsd:attribute name="LocPinX" type="xsd:string" use="optional"/>
  <xsd:attribute name="LocPinY" type="xsd:string" use="optional"/>
  <xsd:attribute name="PinX" type="xsd:string" use="optional"/>
  <xsd:attribute name="PinY" type="xsd:string" use="optional"/>
  <xsd:attribute name="HideText" type="xsd:string" use="optional"/>
  <xsd:attribute name="FillBackground" type="xsd:string" use="optional"/>
  <xsd:attribute name="FillBackgroundTrans" type="xsd:string" use="optional"/>
  <xsd:attribute name="FillForeground" type="xsd:string" use="optional"/>
  <xsd:attribute name="FillForegroundTrans" type="xsd:string" use="optional"/>
  <xsd:attribute name="FillPattern" type="xsd:string" use="optional"/>
  <xsd:attribute name="LineColor" type="xsd:string" use="optional"/>
  <xsd:attribute name="LineColorTrans" type="xsd:string" use="optional"/>
  <xsd:attribute name="LineWeight" type="xsd:string" use="optional"/>
  <xsd:attribute name="TextWidth" type="xsd:string" use="optional"/>
  <xsd:attribute name="TextHeight" type="xsd:string" use="optional"/>
  <xsd:attribute name="TextAngle" type="xsd:string" use="optional"/>
  <xsd:attribute name="TextLocPinX" type="xsd:string" use="optional"/>
  <xsd:attribute name="TextLocPinY" type="xsd:string" use="optional"/>
  <xsd:attribute name="TextPinX" type="xsd:string" use="optional"/>
  <xsd:attribute name="TextPinY" type="xsd:string" use="optional"/>
  <xsd:attribute name="Rounding" type="xsd:string" use="optional"/>
</xsd:complexType>
```

2.4.6.2.5 CT_Formulas

Referenced by: CT_DataGraphics_Shape

The **CT_Formulas** complex type specifies the CT_Sheet elements within the containing CT_DataGraphics_Shape element that has properties that are recalculated by formula expressions.

Child Elements:

Sheet : A list of CT_Sheet elements.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_Formulas">
  <xsd:sequence>
    <xsd:element name="Sheet" minOccurs="1" maxOccurs="unbounded" type="CT_Sheet"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.6.2.6 CT_FormulaReferences

Referenced by: CT_DataGraphics_Shape

The **CT_FormulaReferences** complex type specifies a list of formula references. A formula reference is a formula expression that is shared and referenced by other formula expressions within the containing CT_DataGraphics_Shape element. Each formula reference is specified as a string attribute. The name of the attribute is a formula identifier. The formula identifier MUST be unique amongst all formula identifiers in the CT_FormulaReferences element for the containing CT_DataGraphics_Shape element. The value of the attribute MUST be a formula expression.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_FormulaReferences">
  <xsd:anyAttribute namespace="##local" processContents="skip"/>
</xsd:complexType>
```

2.4.6.2.7 CT_DataGraphics_Shape

Referenced by: CT_DataGraphics_Page

The **CT_DataGraphics_Shape** complex type specifies information used to update the data graphics associated with a shape.

Child Elements:

DataGraphicDefinitions : An optional CT_DataGraphicDefinitions element. It MUST exist if **Formulas** does not exist.

Formulas : An optional CT_Formulas element. It MUST exist if **DataGraphicDefinitions** does not exist.

FormulaReferences : An optional CT_FormulaReferences element. It MUST NOT exist if **Formulas** does not exist.

Attributes:

ID : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies an identifier for the data graphic. It MUST NOT be an empty string and MUST be unique amongst the **ID** attributes of the CT_DataGraphics_Shape elements in this XML part.

ShapeName : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the identifier of the shape that references this data graphic. It MUST match the **Name** attribute of a **Canvas** element in the ShapeGraphic Part.

DataRefs : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the external data that is bound to this data graphic. If this attribute does not exist or if it is an empty string then the external data bound to this data graphic is unknown. If a string for this attribute does exist it MUST have the following format:

```
rowID1,colID1;rowID2,colID2;…;rowIDn,colIDn
```

Name	Description
rowID	An integer that specifies the **row** in the recordset that this shape is bound to. MUST be the value of the **BindingID** attribute of a CT_Binding element in the DataBinding XML Part.
colID	A string that specifies the **field** in the recordset that this shape is bound to. MUST be the value of the **Name** attribute of a CT_DataColumn element in the DataConnection XML Part.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_DataGraphics_Shape">
  <xsd:sequence>
    <xsd:element name="DataGraphicDefinitions" minOccurs="0" maxOccurs="1"
type="CT_DataGraphicDefinitions"/>
    <xsd:element name="Formulas" minOccurs="0" maxOccurs="1" type="CT_Formulas"/>
    <xsd:element name="FormulaReferences" minOccurs="0" maxOccurs="1"
type="CT_FormulaReferences"/>
  </xsd:sequence>
  <xsd:attribute name="ID" type="xsd:string" use="required"/>
  <xsd:attribute name="ShapeName" type="xsd:string" use="required"/>
  <xsd:attribute name="DataRefs" type="xsd:string" use="optional"/>
</xsd:complexType>
```

2.4.6.2.8 CT_DataGraphics_Page

Referenced by: CT_DataGraphics

The **CT_DataGraphics_Page** complex type specifies information used to update all the data graphics in a drawing page.

Child Elements:

Shape : An optional list of CT_DataGraphics_Shape elements. CT_DataGraphics_Shape elements are not present if the shapes of the drawing page are not associated with data graphics.

Attributes:

PageName : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the name of the drawing page.

ViewTransform : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a geometric transform used to convert the Cartesian coordinates of the graphic elements in the ShapeGraphic Part to the coordinate system of the **view** that displays the drawing page. It MUST have the following format:

```
m11 m12 m21 m22 dx dy
```

`m11`, `m12`, `m21`, `m22`, `dx`, `dy` are float values that represent a 3-by-3 affine transformation matrix as specified in the following table. All values MUST be separated by single spaces.

	Column 1	Column 2	Column 3
Row 1	m11	m12	0
Row 2	m12	m22	0
Row 3	dx	dy	1

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_DataGraphics_Page">
  <xsd:sequence>
    <xsd:element name="Shape" minOccurs="0" maxOccurs="unbounded"
type="CT_DataGraphics_Shape"/>
  </xsd:sequence>
  <xsd:attribute name="PageName" type="xsd:string"/>
  <xsd:attribute name="ViewTransform" type="xsd:string"/>
</xsd:complexType>
```

2.4.6.2.9 CT_DataGraphics

Referenced by: DataGraphics

The **CT_DataGraphics** complex type specifies information used to update all the data graphics in a Web drawing.

Child Elements:

Page : A list of CT_DataGraphics_Page elements.

Attributes:

ColorTable : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the list of colors used to render data graphics. It MUST have the following format:

```
R1,G1,B1 R2,G2,B2 … Rn,Gn,Bn
```

Each color MUST be specified using the **RGB** color model as described in the following table. The components of a color MUST be separated by commas. The colors MUST be separated by single spaces.

Name	Description
Rn	An integer value that specifies the red component of the color. MUST be greater than or equal to 0 and less than or equal to 255.
Gn	An integer value that specifies the green component of the color. MUST be greater than or equal to 0 and less than or equal to 255.
Bn	An integer value that specifies the blue component of the color. MUST be greater than or equal to 0 and less than or equal to 255.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_DataGraphics">
  <xsd:sequence>
    <xsd:element name="Page" minOccurs="1" maxOccurs="unbounded"
type="CT_DataGraphics_Page"/>
  </xsd:sequence>
  <xsd:attribute name="ColorTable" type="xsd:string"/>
</xsd:complexType>
```

2.4.7 DataGraphicDefinition XML Part

This part specifies information used to update the image element data graphics in a Web drawing.

2.4.7.1 Global Elements

2.4.7.1.1 DataGraphicDefs

A CT_DataGraphicDefs element.

```
<xsd:element name="DataGraphicDefs" type="CT_DataGraphicDefs"/>
```

2.4.7.2 Complex Types

2.4.7.2.1 CT_IconSetRule

Referenced by: CT_IconSetDef

The **CT_IconSetRule** complex type specifies a **rule** for an image element data graphic that contains an **embedded image**. The rule is specified by the **Formula** attribute and the **Image** attribute to be rendered when the condition specified in **Formula** has been met.

Attributes:

Name : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies an identifier for the rule. It MUST NOT be an empty string and MUST be unique amongst the identifiers specified by the **Name** attribute of the CT_IconSetRule elements in this XML part.

Formula : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that is a formula expression specifying a condition to be evaluated. The condition is met if the result of the formula evaluation is not equal to -1.

Image : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the name of the embedded image to be rendered when the condition specified by **Formula** has been met.

```
<xsd:complexType name="CT_IconSetRule">
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="Formula" type="xsd:string" use="required"/>
  <xsd:attribute name="Image" type="xsd:string" use="required"/>
</xsd:complexType>
```

2.4.7.2.2 CT_IconSetDef

Referenced by: CT_DataGraphicDefs

The **CT_IconSetDef** complex type specifies information used to update an image element data graphic that contains an embedded image during diagram update.

Child Elements:

IconSetRule : A list of CT_IconSetRule elements.

Attributes:

DefID : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies an identifier of the CT_IconSetDef. It MUST NOT be an empty string and MUST be unique amongst the identifiers specified by the **DefID** attribute of the CT_IconSetDef elements in this XML part.

```
<xsd:complexType name="CT_IconSetDef">
  <xsd:sequence>
    <xsd:element name="IconSetRule" minOccurs="1" maxOccurs="unbounded"
type="CT IconSetRule"/>
  </xsd:sequence>
  <xsd:attribute name="DefID" type="xsd:string" use="required"/>
</xsd:complexType>
```

2.4.7.2.3 CT_DataGraphicDefs

Referenced by: DataGraphicDefs

The **CT_DataGraphicDefs** complex type specifies information used to update all image element data graphic in the Web drawing.

Child Elements:

IconSetDef : An optional list of CT_IconSetDef elements. CT_IconSetDef elements are not present if the document does not have image element data graphic that contain **embedded images**.

```
<xsd:complexType name="CT_DataGraphicDefs">
  <xsd:sequence>
    <xsd:element name="IconSetDef" minOccurs="0" maxOccurs="unbounded" type="CT_IconSetDef"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.8 ShapeInfo XML Part

This part specifies information about the shapes on a drawing page in the document.

2.4.8.1 Global Elements

2.4.8.1.1 Page

A CT_Page element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="Page" type="CT_Page"/>
```

2.4.8.2 Complex Types

2.4.8.2.1 CT_PageInfo

Referenced by: CT_Pages

The **CT_PageInfo** complex type specifies the name of a drawing page in the document.

Attributes:

Name : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the name of a drawing page. It MUST be unique amongst all the drawing pages specified by the CT_PageInfo child elements of CT_Pages.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_PageInfo">
  <xsd:attribute name="Name" type="xsd:string"/>
</xsd:complexType>
```

2.4.8.2.2 CT_Pages

Referenced by: CT_Page

The **CT_Pages** complex type specifies the names of all the drawing pages in the document.

Child Elements:

PageInfo : A list of CT_PageInfo elements.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_Pages">
  <xsd:sequence>
    <xsd:element name="PageInfo" minOccurs="1" maxOccurs="unbounded" type="CT_PageInfo"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.8.2.3 CT_ShapeData

Referenced by: CT_ShapeDataItems

The **CT_ShapeData** complex type specifies a shape data item.

Attributes:

Name : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the label of the shape data item.

FormattedValue : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the value formatted for display. It MUST NOT be an empty string.

Value : An attribute that specifies the value of the shape data item. The type of this attribute is specified by **Type**.

Format : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the vFormatString used to format the value for display.

Type : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the **data type** of the shape data item. It MUST be a value from the following table:

Value	Meaning
0	A string value. The type of **Value** MUST be string.
1	A fixed list of values represented as a string. The type of **Value** MUST be string.
2	A numerical value. The type of **Value** MUST be double.
3	A **Boolean** value. The type of **Value** MUST be string and **Value** MUST be "TRUE" or "FALSE".
4	A variable list of values represented as a string. The type of **Value** MUST be string.
5	An OLE automation date value as specified in [MS-OAUT] section 2.2.25. The type of **Value** MUST be double.
6	A duration value. The type of **Value** MUST be double.
7	A currency value. The type of **Value** MUST be double.

LangID : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the vLanguageID used to format the value.

UnitLabel : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the vUnitLabel used to format the value.

CalendarID : An optional xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the vCalendar used to format a date value. The default value is 0. It MUST be ignored if **Type** is not equal to 5.

CurrencyID : An optional xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the vCurrencyID used to format a currency value. The default value is 0. It MUST be ignored if **Type** is not equal to 7.

ServerAction : An optional xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the vServerAction for the formatted value. The default value is 0.

Unit : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the vNumAny that **Value** is converted from before formatting it for display.

DisplayUnit : An optional xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the vNumAny that **Value** is converted to before formatting it for display. The default is the value of **Unit**.

BindingID : An optional xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the CT_Binding identifier for this shape data item. It MUST be one of the values specified by the **BindingID** attribute of the CT_Binding elements in the DataBinding XML Part. If this attribute does not exist then this shape data item is not bound to data.

HyperlinkID : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the CT_Hyperlink identifier for this shape data item. It MUST be the value of the **Name** attribute of a CT_Hyperlink child element of the corresponding CT_ShapeInfo element in this XML part. If this attribute does not exist then this shape data item is not associated with a hyperlink.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_ShapeData">
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="FormattedValue" type="xsd:string" use="required"/>
  <xsd:attribute name="Value" type="xsd:string" use="required"/>
```

```
    <xsd:attribute name="Format" type="xsd:string" use="required"/>
    <xsd:attribute name="Type" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:enumeration value="0"/>
          <xsd:enumeration value="1"/>
          <xsd:enumeration value="2"/>
          <xsd:enumeration value="3"/>
          <xsd:enumeration value="4"/>
          <xsd:enumeration value="5"/>
          <xsd:enumeration value="6"/>
          <xsd:enumeration value="7"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="LangID" type="xsd:integer" use="required"/>
    <xsd:attribute name="UnitLabel" type="xsd:integer" use="required"/>
    <xsd:attribute name="CalendarID" type="xsd:integer" use="optional"/>
    <xsd:attribute name="CurrencyID" type="xsd:integer" use="optional"/>
    <xsd:attribute name="ServerAction" type="xsd:integer" use="optional"/>
    <xsd:attribute name="Unit" type="xsd:integer" use="required"/>
    <xsd:attribute name="DisplayUnit" type="xsd:integer" use="optional"/>
    <xsd:attribute name="BindingID" type="xsd:integer" use="optional"/>
    <xsd:attribute name="HyperlinkID" type="xsd:string" use="optional"/>
  </xsd:complexType>
```

2.4.8.2.4 CT_ShapeDataItems

Referenced by: CT_ShapeInfo

The **CT_ShapeDataItems** complex type specifies the shape data items associated with a shape.

Child Elements:

ShapeData : A list of CT_ShapeData elements.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
  <xsd:complexType name="CT_ShapeDataItems">
    <xsd:sequence>
      <xsd:element name="ShapeData" minOccurs="1" maxOccurs="unbounded" type="CT_ShapeData"/>
    </xsd:sequence>
  </xsd:complexType>
```

2.4.8.2.5 CT_Hyperlink

Referenced by: CT_Hyperlinks

The **CT_Hyperlink** complex type specifies a hyperlink associated with a shape.

Attributes:

Name : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies an identifier for the hyperlink. It MUST NOT be an empty string and MUST be unique amongst the identifiers specified by the **Name** attribute of the CT_Hyperlink elements in this XML part.

Value : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the **URI** of an external resource referenced by the hyperlink. It MUST NOT be an empty string. If this attribute exists, **SubAddress**, **SubAddressShape**, and **Zoom** MUST be ignored. If this attribute does not exist, **SubAddress** MUST exist.

Description : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the description of the hyperlink. If this attribute does not exist then the hyperlink does not have a description.

SubAddress : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the identifier of the drawing page referenced by this hyperlink within the document. It MUST NOT be an empty string. If **SubAddress** does not exist, **SubAddressShape** and **Zoom** MUST be ignored.

SubAddressShape : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the identifier of the shape referenced by this hyperlink within the drawing page specified by **SubAddress**. It MUST NOT be an empty string. If **SubAddressShape** does not exist, the hyperlink references the entire drawing page.

Zoom : An optional xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the **zoom level** for the drawing page specified by **SubAddress**. If this attribute does not exist then the zoom level is 100%. If this attribute does exist then it MUST be a value from the following table:

Value	Description
-2	The drawing page is zoomed to fit the width of the **view**.
-1	The drawing page is zoomed to fit in the view.
n > 0	The drawing page is zoomed to n%. n MUST be greater than 0.

Default : An optional xsd:boolean ([XMLSCHEMA2] section 3.2.2) attribute that specifies the default hyperlink associated with the shape. It MUST exist if and only if this hyperlink is the default. Each shape MUST NOT have more than one default hyperlink. If this attribute exists, the value MUST be 1.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_Hyperlink">
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="Value" type="xsd:string" use="optional"/>
  <xsd:attribute name="Description" type="xsd:string" use="optional"/>
  <xsd:attribute name="SubAddress" type="xsd:string" use="optional"/>
  <xsd:attribute name="SubAddressShape" type="xsd:string" use="optional"/>
  <xsd:attribute name="Zoom" use="optional">
    <xsd:simpleType>
      <xsd:union>
        <xsd:simpleType>
          <xsd:restriction base="xsd:integer">
            <xsd:minExclusive value="0"/>
          </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType>
          <xsd:restriction base="xsd:integer">
            <xsd:enumeration value="-2"/>
            <xsd:enumeration value="-1"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:union>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="Default" type="xsd:boolean" use="optional"/>
</xsd:complexType>
```

2.4.8.2.6 CT_Hyperlinks

Referenced by: CT_ShapeInfo

The **CT_Hyperlinks** complex type specifies the hyperlinks associated with a shape.

Child Elements:

Hyperlink : A list of CT_Hyperlink elements.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_Hyperlinks">
  <xsd:sequence>
    <xsd:element name="Hyperlink" minOccurs="1" maxOccurs="unbounded" type="CT_Hyperlink"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.8.2.7 CT_ShapeInfo

Referenced by: CT_Page

The **CT_ShapeInfo** complex type specifies information about a shape.

Child Elements:

ShapeDataItems : An optional CT_ShapeDataItems element. CT_ShapeDataItems element is not present if the shape is not associated with shape data items.

Hyperlinks : An optional CT_Hyperlinks element. CT_Hyperlinks element is not present if the shape is not associated with hyperlinks.

Attributes:

Name : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the identifier of the shape. It MUST match exactly the **Name** of the **XAML** element that represents the shape as specified in the ShapeGraphic Part. It MUST NOT be an empty string and MUST be unique amongst all the identifiers specified by the **Name** attribute of the CT_ShapeInfo elements in this XML part.

DisplayName : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a display name for the shape.

Guid : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a **GUID** for the shape. If this attribute does not exist then the shape does not have a GUID.

Layout : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies geometric information for the **bounding rectangle** of the shape. The structure of this information is specified in the following table:

Name	Description
type	A string that specifies the shape type. MUST be "2D" and specifies a bi-dimensional shape.
bounds	A structure that specifies the minimum bounding rectangle for the shape as specified in the following table:

Name	Description
x	An integer that specifies the upper left x-coordinate of the bounding rectangle.
y	An integer that specifies the upper left y-coordinate of the bounding rectangle.
width	A positive integer that specifies the width of the bounding rectangle.
height	A positive integer that specifies the height of the bounding rectangle.

The coordinates of the bounding rectangle MUST be relative to the top, left corner of the drawing page that contains the shape. All the values MUST be specified in **pixels**. The string MUST be formatted using the JavaScript Object Notation as specified in [RFC4627] as follows:

```
{"type":"2D","bounds":{"x":x-val,"y":y-val,"width":width-val,"height":height-val}}
```

where **x-val**, **y-val**, **width-val**, and **height-val** are integers specifying the x, y, width and height fields, respectively.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT ShapeInfo">
  <xsd:sequence>
    <xsd:element name="ShapeDataItems" minOccurs="0" maxOccurs="1" type="CT ShapeDataItems"/>
    <xsd:element name="Hyperlinks" minOccurs="0" maxOccurs="1" type="CT_Hyperlinks"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="DisplayName" type="xsd:string"/>
  <xsd:attribute name="Guid" type="xsd:string" use="optional"/>
  <xsd:attribute name="Layout" type="xsd:string" use="required"/>
</xsd:complexType>
```

2.4.8.2.8 CT_Page

Referenced by: Page

The **CT_Page** complex type specifies shape information within a drawing page in the document.

Child Elements:

Pages : A CT_Pages element.

ShapeInfo : An optional list of CT_ShapeInfo elements. CT_ShapeInfo elements are not present if there are no shapes in the drawing page.

Attributes:

Name : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the name of the drawing page. It MUST NOT be an empty string. It MUST match the **Name** attribute of one of the CT_PageMetaData elements in the App XML Part. It MUST match a name in the **Name** attribute of one of the **PageInfo** child elements of the **Pages** element.

Zoom : An xsd:double ([XMLSCHEMA2] section 3.2.5) attribute that specifies the **zoom level** at which the drawing page is initially displayed. It MUST be a value from the following table:

Value	Description
-2.0	The drawing page is zoomed to fit the width of the **view**.

Value	Description
-1.0	The drawing page is zoomed to fit the entire drawing page in the view.
Any value greater than or equal to 0.0	The drawing page is zoomed to **Zoom***100%.

OffsetX : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the x-coordinate of the point on the drawing page that is centered in the view when the drawing page is initially displayed. The value is relative to the top, left corner of the drawing page and is expressed in the coordinate system of the view that displays the drawing page.

OffsetY : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the y-coordinate of the point on the drawing page that is centered in the view when the drawing page is initially displayed. The value is relative to the top, left corner of the drawing page and is expressed in the coordinate system of the view that displays the drawing page.

DefaultUnitsText : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the default typographic measurement unit to be used in formula evaluations for this drawing page if none is explicitly specified in a formula expression. It MUST be one of the following values:

Value	Meaning
48	Picas and points
49	Picas and points
50	Points
51	Picas
52	Ciceros and didots
53	Didots
54	Ciceros

DefaultUnitsAngle : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the default angle measurement unit to be used in formula evaluations for this drawing page if none is explicitly specified in a formula expression. It MUST be one of the following values:

Value	Meaning
80	Decimal degrees
81	Decimal degrees
82	Degrees-minutes-seconds
83	Radians
84	Minutes-seconds
85	Seconds

DefaultUnitsDuration : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the default duration measurement unit to be used in formula evaluations for this drawing page if none is explicitly specified in a formula expression. It MUST be one of the following values:

Value	Meaning
42	Days
43	Weeks
44	Days
45	Hours
46	Minutes
47	Seconds

DefaultUnitsPage : An xsd:integer ([XMLSCHEMA2] section 3.3.13) attribute that specifies the default length measurement in page units to be used in formula evaluations for this drawing page if none is explicitly specified in a formula expression. It MUST be one of the following values:

Value	Meaning
43	Weeks
44	Days
45	Hours
46	Minutes
47	Seconds
50	Points
51	Picas
53	Didots
54	Ciceros
63	Inches
64	Inches
65	Inches
66	Feet
67	Feet and inches
68	Miles (decimal)
69	Centimeters
70	Millimeters
71	Meters
72	Kilometers

Value	Meaning
73	Inches (fractional)
74	Miles (fractional)
75	Yards
76	Nautical miles

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_Page">
  <xsd:sequence>
    <xsd:element name="Pages" minOccurs="1" maxOccurs="1" type="CT_Pages"/>
    <xsd:element name="ShapeInfo" minOccurs="0" maxOccurs="unbounded" type="CT_ShapeInfo"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="Zoom" use="required">
    <xsd:simpleType>
      <xsd:union>
        <xsd:simpleType>
          <xsd:restriction base="xsd:double">
            <xsd:enumeration value="-2"/>
            <xsd:enumeration value="-1"/>
          </xsd:restriction>
        </xsd:simpleType>
        <xsd:simpleType>
          <xsd:restriction base="xsd:double">
            <xsd:minInclusive value="0"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:union>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="OffsetX" type="xsd:integer" use="required"/>
  <xsd:attribute name="OffsetY" type="xsd:integer" use="required"/>
  <xsd:attribute name="DefaultUnitsText" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:integer">
        <xsd:enumeration value="48"/>
        <xsd:enumeration value="49"/>
        <xsd:enumeration value="50"/>
        <xsd:enumeration value="51"/>
        <xsd:enumeration value="52"/>
        <xsd:enumeration value="53"/>
        <xsd:enumeration value="54"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="DefaultUnitsAngle" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:integer">
        <xsd:enumeration value="80"/>
        <xsd:enumeration value="81"/>
        <xsd:enumeration value="82"/>
        <xsd:enumeration value="83"/>
        <xsd:enumeration value="84"/>
        <xsd:enumeration value="85"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
  <xsd:attribute name="DefaultUnitsDuration" use="required">
    <xsd:simpleType>
```

```
        <xsd:restriction base="xsd:integer">
          <xsd:enumeration value="42"/>
          <xsd:enumeration value="43"/>
          <xsd:enumeration value="44"/>
          <xsd:enumeration value="45"/>
          <xsd:enumeration value="46"/>
          <xsd:enumeration value="47"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="DefaultUnitsPage" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:enumeration value="43"/>
          <xsd:enumeration value="44"/>
          <xsd:enumeration value="45"/>
          <xsd:enumeration value="46"/>
          <xsd:enumeration value="47"/>
          <xsd:enumeration value="50"/>
          <xsd:enumeration value="51"/>
          <xsd:enumeration value="53"/>
          <xsd:enumeration value="54"/>
          <xsd:enumeration value="63"/>
          <xsd:enumeration value="64"/>
          <xsd:enumeration value="65"/>
          <xsd:enumeration value="66"/>
          <xsd:enumeration value="67"/>
          <xsd:enumeration value="68"/>
          <xsd:enumeration value="69"/>
          <xsd:enumeration value="70"/>
          <xsd:enumeration value="71"/>
          <xsd:enumeration value="72"/>
          <xsd:enumeration value="73"/>
          <xsd:enumeration value="74"/>
          <xsd:enumeration value="75"/>
          <xsd:enumeration value="76"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
```

2.4.9 ShapeOutline XML Part

This part specifies geometric outline information for the shapes in a drawing page of the document. The outlines enable interactivity with the shapes on the drawing page. These outlines are in the image map format. For more details about image maps see [HTML] section 13.6.1.

2.4.9.1 Global Elements

2.4.9.1.1 Page

A CT_ShapeOutline_Page element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment defines the contents of this element.

```
  <xsd:element name="Page" type="CT_ShapeOutline_Page"/>
```

2.4.9.2 Complex Types

2.4.9.2.1 CT_Path

Referenced by: CT_Shape

The **CT_Path** complex type specifies geometric outline information for a shape.

Attributes:

Shape : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the type of outline. It MUST be `"poly"`. The default value is `"poly"`.

Data : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the Cartesian coordinates of the outline. It MUST have the following format:

```
x1,y1,x2,y2...xn,yn
```

`xn` and `yn` MUST be integer values and they MUST occur in pairs. Each `xn,yn` pair specifies the coordinates of a point that defines one vertex of the outline. The outline is formed by connecting the points in the order they appear in this string attribute. There MUST be at least 4 points. The first and the last point MUST be the same. The coordinates MUST be measured in **pixels** relative to the top, left corner of the drawing page that contains the shape. All values MUST be separated by commas.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment defines the contents of this complex type.

```
<xsd:complexType name="CT_Path">
  <xsd:attribute name="Shape" type="xsd:string" fixed="poly"/>
  <xsd:attribute name="Data" use="required">
    <xsd:simpleType>
      <xsd:restriction base="xsd:string">
        <xsd:pattern value="\d*(,\d*){3,}"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:attribute>
</xsd:complexType>
```

2.4.9.2.2 CT_Shape

Referenced by: CT_Shapes

The **CT_Shape** complex type specifies geometric outline information for the shape identified by the **Name** attribute.

Child Elements:

Path : A list of CT_Path elements.

Attributes:

Name : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the identifier of the shape. It MUST be one of the identifiers specified by the **Name** attribute of the CT_ShapeInfo elements in the ShapeInfo XML Part.

Layout : This attribute is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment defines the contents of this complex type.

```
<xsd:complexType name="CT_Shape">
  <xsd:sequence>
    <xsd:element name="Path" minOccurs="1" maxOccurs="unbounded" type="CT_Path"/>
```

```
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="Layout" type="xsd:string"/>
</xsd:complexType>
```

2.4.9.2.3 CT_Shapes

Referenced by: CT_ShapeOutline_Page

The **CT_Shapes** complex type specifies geometric outline information for multiple shapes.

Child Elements:

Shape : An optional list of CT_Shape elements. CT_Shape elements are not present if there are no shapes in the drawing page.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment defines the contents of this complex type.

```
<xsd:complexType name="CT_Shapes">
  <xsd:sequence>
    <xsd:element name="Shape" minOccurs="0" maxOccurs="unbounded" type="CT Shape"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.9.2.4 CT_ShapeOutline_Page

Referenced by: Page

The **CT_ShapeOutline_Page** complex type specifies geometric outline information for the shapes within a drawing page of the document.

Child Elements:

Shapes : A CT_Shapes element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment defines the contents of this complex type.

```
<xsd:complexType name="CT ShapeOutline Page">
  <xsd:sequence>
    <xsd:element name="Shapes" type="CT_Shapes"/>
  </xsd:sequence>
</xsd:complexType>
```

2.4.10 ShapeTextBinding XML Part

This part specifies information used to update the text element data graphics in a Web drawing.

2.4.10.1 Global Elements

2.4.10.1.1 Page

A CT_ShapeTextBinding_Page element.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this element.

```
<xsd:element name="Page" type="CT_ShapeTextBinding_Page"/>
```

2.4.10.2 Complex Types

2.4.10.2.1 CT_TextRun

Referenced by: CT_ShapeText

The **CT_TextRun** complex type specifies information about a text run of a text element data graphic.

Attributes:

Name : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the identifier of the corresponding text run in the text element data graphic. It MUST be unique amongst the **Name** attributes of the CT_TextRun elements in this XML part.

Formula : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the content of the text run in a text element data graphic during diagram update. The result token value of the formula evaluation is converted to a string. If the result token type is PtgStr1, the result token value is used without additional formatting. If the result token type is not PtgStr1, the result token value is formatted using the information specified by the **Format** attribute. It MUST exist if the **Color** attribute does not exist. It MUST NOT be an empty string.

Format : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies formatting information for the result of the evaluation of **Formula**. It MUST exist if and only if the **Formula** attribute exists. It MUST be ignored if the result type of the **Formula** evaluation is not PtgStr1. It MUST have the following format:

```
Formatting@ServerAction@UnitLabel@CurrencyID@DisplayUnit@LanguageID@CalendarID
```

Name	Description
Formatting	A vFormatString.
ServerAction	A vServerAction.
UnitLabel	A vUnitLabel.
CurrencyID	A vCurrencyID. MUST be ignored if the result of the **Formula** evaluation does not specify a currency.
DisplayUnit	A vUnitType that the result value of the **Formula** evaluation is converted to before formatting. MUST be zero or the numerical identifier of a token type, as specified in section 2.5.5, that is part of the vUnitType grouping. MUST be ignored if equal to zero.
LanguageID	A vLanguageID.
CalendarID	A vCalendar. MUST be ignored if the result of the **Formula** evaluation does not specify a date.

Color : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies a formula expression used to recalculate the color of the text run upon **refresh** in a text element data graphic during diagram update. The result of the formula evaluation is converted to a color value. The color value is a string representing an eight-digit hexadecimal number with a prefix of the character "#". The first two digits MUST be equal to FF, the next two specify the intensity of red, the next two specify

the intensity of green and the final two specify the intensity of blue. It MUST exist if the **Format** attribute does not exist. It MUST NOT be an empty string.

DataRefs : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the external data bound to **Formula**. If this attribute does not exist or if it is an empty string then the external data bound to **Formula** is unknown. This attribute is unused if **Formula** does not exist. If a string for this attribute does exist it MUST have the following format:

```
rowID,colID
```

Name	Description
rowID	An integer that specifies the external data **row** that the text run is bound to. MUST be the value of the **BindingID** attribute of a CT_Binding element in the DataBinding XML Part.
colID	A string that specifies the **field** in the recordset that the text run is bound to. MUST be the value of the **Name** attribute of a CT_DataColumn element in the DataConnection XML Part.

The `rowID` and `colID` values MUST be separated by a comma.

DataRefsColor : An optional xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the external data bound to **Color**. If this attribute does not exist or if it is an empty string then the external data bound to **Color** is unknown. This attribute is unused if **Color** does not exist. If a string for this attribute does exist it MUST have the format described by **DataRefs**.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_TextRun">
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
  <xsd:attribute name="Formula" type="xsd:string" use="optional"/>
  <xsd:attribute name="Format" type="xsd:string" use="optional"/>
  <xsd:attribute name="Color" type="xsd:string" use="optional"/>
  <xsd:attribute name="DataRefs" type="xsd:string" use="optional"/>
  <xsd:attribute name="DataRefsColor" type="xsd:string" use="optional"/>
</xsd:complexType>
```

2.4.10.2.2 CT_ShapeText

Referenced by: CT_ShapeTextBinding_Page

The **CT_ShapeText** complex type specifies information about the text element data graphic for a shape.

Child Elements:

TextRun : A CT_TextRun element.

Attributes:

Name : An xsd:string ([XMLSCHEMA2] section 3.2.1) attribute that specifies the identifier of the corresponding text element data graphic in the ShapeGraphic Part. It MUST be unique amongst all the **Name** attributes of the CT_ShapeText elements in this XML part.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT_ShapeText">
```

```
  <xsd:sequence>
    <xsd:element name="TextRun" minOccurs="1" maxOccurs="1" type="CT_TextRun"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string" use="required"/>
</xsd:complexType>
```

2.4.10.2.3 CT_ShapeTextBinding_Page

Referenced by: Page

The **CT_ShapeTextBinding_Page** complex type specifies information about the text element data graphics for shapes in a drawing page.

Child Elements:

ShapeText : An optional list of CT_ShapeText elements. CT_ShapeText elements are not present if there are no text element data graphics associated with the shapes of the drawing page.

Attributes:

Name : This attribute is unused and MUST be ignored.

The following W3C XML Schema ([XMLSCHEMA1] section 2.1) fragment specifies the contents of this complex type.

```
<xsd:complexType name="CT ShapeTextBinding Page">
  <xsd:sequence>
    <xsd:element name="ShapeText" minOccurs="0" maxOccurs="unbounded" type="CT_ShapeText"/>
  </xsd:sequence>
  <xsd:attribute name="Name" type="xsd:string"/>
</xsd:complexType>
```

2.5 Formula Evaluation and Shape Property Recalculation

2.5.1 Introduction

The following sections specify the elements of a formula expression.

2.5.2 Formula ABNF and Full Grammar Definition

A formula expression MUST conform to the following ABNF [RFC4234] grammar.

```
val = expression

expression = sp (operand / function-call)

operand = token

token = PtgAcre / PtgAngDD / PtgAngDft / PtgAngDMS / PtgAngRad / PtgBool /
        PtgColorRGB / PtgCy / PtgDataBinding / PtgDate / PtgEDay / PtgEHour /
        PtgEMin / PtgErr / PtgESec / PtgEWeek / PtgFunc / PtgFuncVar / PtgHectare /
        PtgJmp / PtgJmpF / PtgJmpLabel / PtgJmpT / PtgMissArg / PtgNoOp / PtgNum
        / PtgNumCM / PtgNumDft / PtgNumF / PtgNumFI / PtgNumI / PtgNumKM /
        PtgNumM / PtgNumMI / PtgNumMM / PtgNumMultiDim / PtgNumNM /
        PtgNumPct / PtgNumYards / PtgPageDft / PtgPop / PtgPnt / PtgPushTop /
```

PtgRecalcRef / PtgStr1 / PtgTDurDft / PtgTypCD / PtgTypCi / PtgTypDft / PtgTypDi / PtgTypPi / PtgTypPP / PtgTypPt / PtgUnsWord

```
function-call = if-expression / fixed-func / variable-func

if-expression = expression sp (PtgJmpF / PtgJmpT) sp expression sp PtgJmp sp
        PtgJmpLabel sp PtgPop sp expression PtgJmpLabel

fixed-func = Abs / ACos / Add / Ang360 / ASin / ATan / ATan2 / BitAnd / BitNot /
        BitOr / BitXor / Blend / CellIsThemed / Char / Cos / Cat / CosH / CY /
        Date / Deg / Div / EEQ / EGE / EGT / ELE / ELT / ENE / FEQ / FGE / FGT
        / FLE / FLT / FNE / FormatEx / HSL / Hue / HueDiff / Int / IntersectX /
        IntersectY / Intup / IsErr / IsErrNA / IsError / IsErrValue / Len / Ln / Log10
        / Lower / Lum / LumDiff / Magnitude / Mid / Modulus / MsoShade / MsoTint
        / Mul / Not / Now / Pct / Pi / Pow / Pnt / Rad / Rand / Replace / RGB /
        Round / Sat / SatDiff / SetAtRefEval / Shade / Sin / SinH / Sqrt /
        StrSameEx / Sub / Tan / TanH / TextHeight / Time / Tint / Tone / Trim /
        Trunc / UMinus / UPlus / Upper

variable-func = And / Bound / Ceiling / DateTime / DateValue / Day / DayOfYear /
        Find / Floor / Hour / Index / Left / Lookup / Max / Min / Minute / Month /
        Or / Right / Second / SetAtRef / SetAtRefExpr / ShapeText / Sign / StrSame
        / Substitute / Sum / TimeValue / WeekDay / Year

fixed-func-name = "ABS" / "ACOS" / "_ADD" / "ANG360" / "ASIN" / "ATAN" /
        "ATAN2" / "BITAND" / "BITNOT" / "BITOR" / "BITXOR" / "BLEND" / "CHAR" /
        "COS" / "_CAT" / "COSH" / "CY" / "DATE" / "DEG" / "_DIV" / "_EEQ" /
        "_EGE" / "_EGT" / "_ELE" / "_ELT" / "_ENE" / "_FEQ" / "_FGE" / "_FGT" /
        "_FLE" / "_FLT" / "_FNE" / "FORMATEX" / "HSL" / "HUE" / "HUEDIFF" /
        "INT" / "INTERSECTX" / "INTERSECTY" / "INTUP" / "ISERR" / "ISERRNA" /
        "ISERROR" / "ISERRVALUE" / "LEN" / "LN" / "LOG10" / "LOWER" / "LUM" /
        "LUMDIFF" / "MAGNITUDE" / "MID" / "MODULUS" / "MSOSHADE" / "MSOTINT" /
        "_MUL" / "NOT" / "NOW" / "_PCT" / "PI" / "POW" / "RAD" / "RAND /
        "REPLACE" / "RGB" / "ROUND" / "SAT" / "SATDIFF" / "SETATREFEVAL" /
        "SHADE" / "SIN" / "SINH" / "SQRT" / "STRSAMEEX" / "_SUB" / "TAN" /
        "TANH" / "TEXTHEIGHT" / "TIME" / "TINT" / "TONE" / "TRIM" / "TRUNC" /
        "_UMINUS" / "_UPLUS" / "UPPER"

var-func-name = "AND" / "BOUND" / "CEILING" / "DATETIME" / "DATEVALUE" / "DAY"
        / "DAYOFYEAR" / "FIND" / "FLOOR" / "HOUR" / "INDEX" / "LEFT" / "LOOKUP"
        / "MAX" / "MIN" / "MINUTE" / "MONTH" / "OR" / "RIGHT" / "SECOND" /
        "SETATREF" / "SETATREFEXPR" / "SIGN" / "STRSAME" / "SUBSTITUTE" / "SUM"
        / "TIMEVALUE" / "WEEDKAY" / "YEAR"

color-value = xsd-int

currency = xsd-unsignedInt

date-time-value = xsd-double

unit = xsd-unsignedInt

shape-name = string-value

shape-data = string-value

shape-data-type = xsd-unsignedInt
```

```
dimension = xsd-unsignedInt

num-args = xsd-unsignedInt

empty-value = ""

string-value = xsd-string

double-value = xsd-double

unsigned-integer-value = xsd-unsignedInt

bool-value = true-value / false-value

true-value = %x54/%x74 %x52/%x72 %x55/%x75 %x45/%x65

false-value = %x46/%x66 %x41/%x61 %x4C/%x6C %x53/%x73 %x45/%x65

xsd-double = ["+" / "-"] *digit ["." 1*digit] [("e" / "E") ["+" / "-"] 1*digit]

xsd-int = ["+" / "-"] 1*digit

xsd-unsignedInt = 1*digit

xsd-string = *(%x20-7E)

token-separator = %x3A           ; colon

sp = %x20                        ; space

digit = %x30-39                  ; Decimal digits (0-9)
```

2.5.3 Function Token Table

This section specifies the list of functions that can be called from a formula expression. The function display name, ABNF func-name and function type are provided for each function.

Display name	ABNF func-name	Type
Abs	ABS	PtgFunc
ACos	ACOS	PtgFunc
Add	_ADD	PtgFunc
And	AND	PtgFunc
Ang360	ANG360	PtgFunc
ASin	ASIN	PtgFunc
ATan	ATAN	PtgFunc
ATan2	ATAN2	PtgFunc
BitAnd	BITAND	PtgFunc
BitNot	BITNOT	PtgFunc
BitOr	BITOR	PtgFunc

Display name	ABNF func-name	Type
BitXor	BITXOR	PtgFunc
Blend	BLEND	PtgFunc
Bound	BOUND	PtgFuncVar
Ceiling	CEILING	PtgFuncVar
CellIsThemed	CELLISTHEMED	PtgFunc
Char	CHAR	PtgFunc
Cos	COS	PtgFunc
Cat	_CAT	PtgFunc
CosH	COSH	PtgFunc
CY	CY	PtgFunc
Date	DATE	PtgFunc
DateTime	DATETIME	PtgFuncVar
DateValue	DATEVALUE	PtgFuncVar
Day	DAY	PtgFuncVar
DayOfYear	DAYOFYEAR	PtgFuncVar
Deg	DEG	PtgFunc
Div	_DIV	PtgFunc
EEQ	_EEQ	PtgFunc
EGE	_EGE	PtgFunc
EGT	_EGT	PtgFunc
ELE	_ELE	PtgFunc
ELT	_ELT	PtgFunc
ENE	_ENE	PtgFunc
FEQ	_FEQ	PtgFunc
FGE	_FGE	PtgFunc
FGT	_FGT	PtgFunc
Find	FIND	PtgFuncVar
FLE	_FLE	PtgFunc
Floor	FLOOR	PtgFuncVar
FLT	_FLT	PtgFunc
FNE	_FNE	PtgFunc
FormatEx	FORMATEX	PtgFunc

Display name	ABNF func-name	Type
Hour	HOUR	PtgFuncVar
HSL	HSL	PtgFunc
Hue	HUE	PtgFunc
HueDiff	HUEDIFF	PtgFunc
Index	INDEX	PtgFuncVar
Int	INT	PtgFunc
IntersectX	INTERSECTX	PtgFunc
IntersectY	INTERSECTY	PtgFunc
Intup	INTUP	PtgFunc
IsErr	ISERR	PtgFunc
IsErrNA	ISERRNA	PtgFunc
IsError	ISSERROR	PtgFunc
IsErrValue	ISERRVALUE	PtgFunc
Left	LEFT	PtgFuncVar
Len	LEN	PtgFunc
Ln	LN	PtgFunc
Log10	LOG10	PtgFunc
Lookup	LOOKUP	PtgFuncVar
Lower	LOWER	PtgFunc
Lum	LUM	PtgFunc
LumDiff	LUMDIFF	PtgFunc
Magnitude	MAGNITUDE	PtgFunc
Max	MAX	PtgFuncVar
Mid	MID	PtgFunc
Min	MIN	PtgFuncVar
Minute	MINUTE	PtgFuncVar
Modulus	MODULUS	PtgFunc
Month	MONTH	PtgFuncVar
MsoShade	MSOSHADE	PtgFunc
MsoTint	MSOTINT	PtgFunc
Mul	_MUL	PtgFunc
Not	NOT	PtgFunc

Display name	ABNF func-name	Type
Now	NOW	PtgFunc
Or	OR	PtgFunc
Pct	_PCT	PtgFunc
Pi	PI	PtgFunc
Pow	POW	PtgFunc
Pnt	PNT	PtgFunc
Rad	RAD	PtgFunc
Rand	RAND	PtgFunc
Replace	REPLACE	PtgFunc
RGB	RGB	PtgFunc
Right	RIGHT	PtgFuncVar
Round	ROUND	PtgFunc
Sat	SAT	PtgFunc
SatDiff	SATDIFF	PtgFunc
Second	SECOND	PtgFuncVar
SetAtRef	SETATREF	PtgFuncVar
SetAtRefEval	SETATREFEVAL	PtgFunc
SetAtRefExpr	SETATREFEXPR	PtgFuncVar
Shade	SHADE	PtgFunc
ShapeText	SHAPETEXT	PtgFuncVar
Sign	SIGN	PtgFuncVar
Sin	SIN	PtgFunc
SinH	SINH	PtgFunc
Sqrt	SQRT	PtgFunc
StrSame	STRSAME	PtgFuncVar
StrSameEx	STRSAMEEX	PtgFunc
Sub	_SUB	PtgFunc
Substitute	SUBSTITUTE	PtgFuncVar
Sum	SUM	PtgFuncVar
Tan	TAN	PtgFunc
TanH	TANH	PtgFunc
TextHeight	TEXTHEIGHT	PtgFunc

Display name	ABNF func-name	Type
TextWidth	TEXTWIDTH	PtgFunc
Time	TIME	PtgFunc
TimeValue	TIMEVALUE	PtgFuncVar
Tint	TINT	PtgFunc
Tone	TONE	PtgFunc
Trim	TRIM	PtgFunc
Trunc	TRUNC	PtgFunc
UMinus	_UMINUS	PtgFunc
UPlus	_UPLUS	PtgFunc
Upper	UPPER	PtgFunc
WeekDay	WEEKDAY	PtgFuncVar
Year	YEAR	PtgFuncVar

2.5.4 Function Token Definitions

This section specifies the function tokens that can be contained in a formula expression. The definition of each function specifies the function name, the type and sequence of expected arguments, and the type of token returned. Information is also included about how the functions are evaluated.

2.5.4.1 Abs

The **Abs** function performs an absolute value calculation.

ABNF:

```
Abs = val " ABS():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: vNum

This function returns a vNum containing the absolute value of **Arg1**.The unit of the return value is equal to the unit of **Arg1**.

2.5.4.2 ACos

The **ACos** function performs the arccosine calculation.

ABNF:

```
ACos = val " ACOS():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgAngDft, PtgErr

This function returns a PtgAngDft containing the arccosine of the value of **Arg1**. If **Arg1** is less than -1 or greater than 1, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.3 Add

The **Add** function performs an addition calculation.

ABNF:

```
Add = val val " _ADD():128"
```

Required Arguments:

Name: **Arg1**

Type: vDoubleEx

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDoubleEx

An argument that specifies the second operand of the calculation.

Return Value:

Type: vNum, PtgCy, PtgErr

This function returns a vNum or a PtgCy containing the sum of **Arg1** and **Arg2**. The type of the return value is calculated by the following algorithm.
SET returnType = PtgNum
SET returnDimension = 0
SET returnCurrencyID = 0
IF Arg1.Type = PtgCy AND Arg2.Type != PtgCy THEN
    SET returnType = PtgCy
    SET returnCurrencyID = currencyID of Arg1
ELSE IF Arg1.Type != PtgCy AND Arg2.Type = PtgCy THEN

```
        SET returnType = PtgCy
        SET returnCurrencyID = currencyID of Arg2
ELSE IF Arg1.Type = PtgCy AND Arg2.Type = PtgCy THEN
        SET returnType = PtgCy
        IF currencyID of Arg1 = currencyID of Arg2 OR currencyID of Arg2 = 1 THEN
                SET returnCurrencyID = currencyID of Arg1
        ELSE IF currencyID of Arg1 = 1 THEN
                SET returnCurrencyID = currencyID of Arg2
        ELSE
                SET returnType = PtgErr
        END IF
ELSE IF Arg1.Type = PtgDate THEN
        SET returnType = Arg1.Type
        SET returnDimension = 1
ELSE IF Arg2.Type= PtgDate THEN
        SET returnType = Arg2.Type
        SET returnDimension = 1
ELSE IF Arg1.Unit = PtgNumDft AND Arg2.Unit is a vUnitType THEN
        SET returnType = Arg2.Unit
        SET returnDimension = 1
ELSE IF Arg1.Unit is a vUnitType THEN
        SET returnType = Arg1.Unit
        IF Arg2.Dimension = 0 OR Arg2.Dimension = Arg1.Dimension THEN
                SET returnDimension = Arg1.Dimension
        ELSE
                SET returnDimension = 1
        END IF
        IF (returnType = PtgAcre OR returnType = PtgHectare) AND returnDimension != 2 THEN
                SET returnType = PtgNumDft
        END IF
ELSE IF Arg2.Unit is a vUnitType THEN
        SET returnType = Arg2.Unit
        IF Arg1.Dimension = 0 OR Arg2.Dimension = Arg1.Dimension THEN
                SET returnDimension = Arg2.Dimension
        ELSE
                SET returnDimension = 1
        END IF

        IF (returnType = PtgAcre OR returnType = PtgHectare) AND returnDimension != 2 THEN
                SET returnType = PtgNumDft
        END IF
END IF
IF returnType = PtgCy THEN
        Return type is PtgCy with currencyID = returnCurrencyID
ELSE IF returnType = PtgErr THEN
        Return type is PtgErr with error code of #VALUE!
ELSE IF returnDimension = 0 THEN
        Return type is PtgNum
ELSE IF returnDimension = 1 THEN
        Return type is returnType
ELSE
        Return type is PtgNumMultiDim with unit = returnType and dimension = returnDimension
END IF
```

2.5.4.4 And

The **And** function converts an argument to a **Boolean** value according to the conversion specified by vBoolean.

ABNF:

```
And = val " AND():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand.

Return Value:

Type: PtgBool

This function returns a PtgBoolcontaining the value of **Arg1**.

2.5.4.5 Ang360

The **Ang360** function normalizes an angle.

ABNF:

```
Ang360 = val " ANG360():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: vAngle

This function returns a vAngle containing an angle equivalent to **Arg1**, normalized to be greater than or equal to zero and less than 2*pi, where the value of **Arg1** is assumed to have the unit of radians. If **Arg1** is a vAngle, the unit of the return value is equal to the unit of **Arg1**; otherwise the function returns a PtgAngDft.

2.5.4.6 ASin

The **ASin** function performs the arcsine calculation.

ABNF:

```
ASin = val " ASIN():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgAngDft, PtgErr

This function returns a PtgAngDft containing the arcsine of the value of **Arg1**. If **Arg1** is less than -1 or greater than 1, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.7 ATan

The **ATan** function performs the arctangent calculation.

ABNF:

```
ATan = val " ATAN():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgAngDft

This function returns a PtgAngDft containing the arctangent of the value of **Arg1**.

2.5.4.8 ATan2

The **ATan2** function calculates the angle between the positive x-axis and a vector.

ABNF:

```
ATan2 = val val " ATAN2():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the y-component of the vector.

Name: **Arg2**

Type: vDouble

An argument that specifies the x-component of the vector.

Return Value:

Type: PtgAngDft

This function returns a PtgAngDft containing the angle between the positive x-axis and the vector represented by **Arg1** and **Arg2**. The value is greater than -pi and less than or equal to pi. If **Arg1** is equal to zero and **Arg2** is equal to zero, the value is zero.

2.5.4.9 BitAnd

The **BitAnd** function performs the bitwise AND operation.

ABNF:

```
BitAnd = val val " BITAND():128"
```

Required Arguments:

Name: **Arg1**

Type: vUnsignedInt

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vUnsignedInt

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgUnsWord

This function returns a PtgUnsWord containing the value of the bitwise AND operation between **Arg1** and **Arg2**.

2.5.4.10 BitNot

The **BitNot** function performs the bitwise NOT operation.

ABNF:

```
BitNot = val " BITNOT():128"
```

Required Arguments:

Name: **Arg1**

Type: vUnsignedInt

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgUnsWord

This function returns a PtgUnsWord containing the value of the bitwise NOT operation on **Arg1**.

2.5.4.11 BitOr

The **BitOr** function performs the bitwise OR operation.

ABNF:

```
BitOr = val val " BITOR():128"
```

Required Arguments:

Name: **Arg1**

Type: vUnsignedInt

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vUnsignedInt

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgUnsWord

This function returns a PtgUnsWord containing the value of the bitwise OR operation between **Arg1** and **Arg2**.

2.5.4.12 BitXor

The **BitXor** function performs the bitwise XOR operation.

ABNF:

```
BitXor = val val " BITXOR():128"
```

Required Arguments:

Name: **Arg1**

Type: vUnsignedInt

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vUnsignedInt

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgUnsWord

This function returns a PtgUnsWord containing the value of the bitwise XOR operation between **Arg1** and **Arg2**.

2.5.4.13 Blend

The **Blend** function performs a blend of two colors.

ABNF:

```
Blend = val val val " BLEND():128"
```

Required Arguments:

Name: **Color1**

Type: vColor

An argument that specifies the first color.

Name: **Color2**

Type: vColor

An argument that specifies the second color.

Name: **Fraction**

Type: vDouble

An argument that specifies the fractional amount of **Color2** in the blended color.

Return Value:

Type: PtgColorRGB, PtgErr

This function returns a PtgColorRGB containing the blended color. If **Fraction** is less than 0 or greater than 1, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.14 Bound

The **Bound** function constrains a value by one or more ranges.

ABNF:

```
Bound = val val 1*(val val val) sp unsigned-integer-value ";BOUND():129"
```

Required Arguments:

Name: **Number**

Type: vDouble

An argument that specifies the value to be constrained.

Name: **BoundType**

Type: vUnsignedInt

An argument that specifies the bounding type. The valid values are described in the following table.

Value	Meaning
0	Constrain **Number** inclusive of any of the ranges specified
1	Constrain **Number** exclusive of all the ranges specified
2	Do not constrain **Number**

Name: **IgnoreRange**

Type: PtgBool

An argument that specifies if this range is not included in constraining **Number**.

Name: **ValueBeg**

Type: vDouble

An argument that specifies the beginning value of the range.

Name: **ValueEnd**

Type: vDouble

An argument that specifies the ending value of the range.

Optional Arguments:

Additional ranges MUST be specified using 1 or more additional groups of **IgnoreRange**, **ValueBeg**, and **ValueEnd**.

Return Value:

Type: vNum, PtgErr

This function returns a vNum containing the constrained value of **Number,** as described in the following table.

Condition	Result
BoundType = 0	If **Number** is in at least one range with **IgnoreRange** equal to FALSE, the function returns **Number.** Otherwise, consider the set of all **ValueBeg** and **ValueEnd** values that belong to a range with **IgnoreRange** equal to FALSE. The function returns the value in this set that is closest to **Number**. If more than one value has the same minimum distance from **Number**, the function returns the value in this set that has minimum distance from **Number** and that appears earliest in the argument list.
BoundType = 1	If **Number** is not in any range with **IgnoreRange** equal to FALSE, the function returns **Number**. Otherwise, consider the set of all **ValueBeg** and **ValueEnd** values that belong to a range with **IgnoreRange** equal to FALSE and that are not nested inside another range with **IgnoreRange** equal to FALSE. The function returns the value in this set

Condition	Result
	that is closest to **Number**. If **Number** is in the exact middle of a **ValueBeg** value and a **ValueEnd** value in this set, the function returns the **ValueEnd** value.
BoundType = 2	The function returns **Number**.
Otherwise	The function returns a PtgErr with an error code of #VALUE!.

If **Number** is a vAngle, the return value is normalized to be greater than or equal to zero and less than 2*pi. The unit of the return value is equal to the unit of **Number**. If **Number** is a ptgPnt or the wrong number of arguments is used, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.15 Cat

The **Cat** function performs the concatenation of two strings.

ABNF:

```
Cat = val val " _CAT():128"
```

Required Arguments:

Name: **Arg1**

Type: vString

An argument that specifies the first string.

Name: **Arg2**

Type: vString

An argument that specifies the second string.

Return Value:

Type: PtgStr1

This function returns a PtgStr1 containing the concatenation of **Arg1** and **Arg2**.

2.5.4.16 Ceiling

The **Ceiling** function performs a ceiling calculation.

ABNF:

```
Ceiling = val [val] sp ("1" / "2") ";CEILING():129"
```

Required Arguments:

Name: **Number**

Type: vDouble

An argument that specifies the value to be rounded.

Optional Arguments:

Name: **Multiple**

Type: vDouble

An argument that specifies the rounding increment. The default value is 1 if **Number** is greater than or equal to 0, and -1 if **Number** is less than 0.

Return Value:

Type: vNum, PtgErr

This function returns a vNum containing the next multiple of **Multiple** after **Number** that is further from zero. If **Number** is a multiple of **Multiple**, the value is equal to **Number**. If **Number** is equal to zero or **Multiple** is equal to zero, the value is zero. The unit of the return value is equal to the unit of **Number**. If **Number** and **Multiple** do not have the same sign, the function returns a PtgErr with an error code of #NUM!.

2.5.4.17 CellIsThemed

The **CellIsThemed** function MUST be ignored.

ABNF:

```
CellIsThemed = val " CELLISTHEMED():128"
```

Required Arguments:

Name: **Arg1**

Type: vBoolean

This argument is unused and MUST be ignored.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of FALSE.

2.5.4.18 Char

The **Char** function performs a conversion from an integer to the corresponding character in the **American National Standards Institute (ANSI) character set**.

ABNF:

```
Char = val " CHAR():128"
```

Required Arguments:

Name: **Number**

Type: vUnsignedInt

An argument that specifies an integer.

Return Value:

Type: PtgStr1, PtgErr

This function returns a PtgStr1 containing the character that corresponds to **Number**. If **Number** is less than 1 or greater than 255, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.19 Cos

The **Cos** function performs the cosine calculation.

ABNF:

```
Cos = val " COS():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgNum

This function returns a PtgNum containing the cosine of the value of **Arg1**, where the value of **Arg1** is assumed to have the unit of radians.

2.5.4.20 CosH

The **CosH** function performs the hyperbolic cosine calculation.

ABNF:

```
CosH = val " COSH():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgNum

This function returns a PtgNum containing the hyperbolic cosine of **Arg1**, where the value of **Arg1** is assumed to have the unit of radians.

2.5.4.21 CY

The **CY** function performs conversion of a value to a currency.

ABNF:

```
CY = val val val " 3;CY():129"
```

Required Arguments:

Name: **Number**

Type: vDouble

An argument that specifies the value to be converted to a currency.

Name: **CyId**

Type: vCurrencyID

An argument that specifies the currency.

Name: **Format**

Type: vString

An argument that specifies the formatting information. It MUST be a numeric format string, as described in [MSDN-FormattingTypes].

Return Value:

Type: PtgCY

This function returns a PtgCY containing **Number**, **CyId,** and **Format**.

2.5.4.22 Date

The **Date** function returns a date, according to the Gregorian calendar, from values representing a year, month, and day.

ABNF:

```
Date = val val val " DATE():128"
```

Required Arguments:

Name: **Year**

Type: vUnsignedInt

An argument that specifies a year. A value between 0 and 29 corresponds to the range of years between 2000 and 2029. A value between 30 and 100 corresponds to the range of years between 1930 and 2000. A value greater than 100 corresponds to that year. It MUST be less than or equal to 9999.

Name: **Month**

Type: vSignedInt

An argument that specifies an offset in months from December 1st of the previous year.

Name: **Day**

Type: vSignedInt

An argument that specifies an offset in days from the last day of the previous month.

Return Value:

Type: PtgDate, PtgErr

This function returns a PtgDate containing the date, according to the Gregorian calendar, represented by **Year**, **Month**, and **Day**. If the arguments specify a date before January 1st, 1899 or between January 1st, 1900 and November 30th, 1900 inclusively, the function returns a PtgErr with an error code of #NUM!. If either **Month** or **Day** cannot be interpreted as vSignedInt, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.23 DateTime

The **DateTime** function converts a value to a date and time.

ABNF:

```
DateTime = val [val] sp ("1" / "2") ";DATETIME():129"
```

Required Arguments:

Name: **DateTimeArg**

Type: vAny

An argument that specifies a value representing a date and time.

Optional Arguments:

Name: **Locale**

Type: vLanguageID

An argument that specifies an **language code identifier (LCID)** to use when parsing **DateTimeArg**.

Return Value:

Type: PtgDate, PtgErr

If **DateTimeArg** is a PtgStr1, this function attempts to parse it using date and time format strings according to .NET globalization **rules** for **Locale**. If **Locale** is not specified, the LCID is specified by the **DocumentLanguage** field of the Properties **GlobalElement** in the App XML Part of the document. For more information about .NET globalization rules, see [MSDN-ENCLOC]. If the string is successfully parsed, the function returns a PtgDate containing the parsed date and time. If the string is not successfully parsed, the function returns a PtgErr with an error code of #VALUE!.

If **DateTimeArg** is not a PtgStr1, the function returns a PtgDate containing the value of **DateTimeArg** interpreted as a vDouble. If **DateTimeArg** cannot be interpreted as a vDouble, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.24 DateValue

The **DateValue** function returns the date component from a value representing a date and time.

ABNF:

```
DateValue = val [val] sp ("1" / "2") ";DATEVALUE():129"
```

Required Arguments:

Name: **DateTimeArg**

Type: vAny

An argument that specifies a value representing a date and time.

Optional Arguments:

Name: **Locale**

Type: vLanguageID

An argument that specifies an **LCID** to use when parsing **DateTimeArg**.

Return Value:

Type: PtgDate, PtgErr

This function attempts a conversion, as described by the DateTime function, of **DateTimeArg** to a PtgDate. If the conversion succeeds, the function returns a PtgDate containing the date component of **DateTimeArg**. If the conversion fails, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.25 Day

The **Day** function returns the day of the month, according to the Gregorian calendar, from a value representing a date and time.

ABNF:

```
Day = val [val] sp ("1" / "2") ";DAY():129"
```

Required Arguments:

Name: **DateTimeArg**

Type: vAny

An argument that specifies a date and time value.

Optional Arguments:

Name: **Locale**

Type: vLanguageID

An argument that specifies an **LCID** to use when parsing **DateTimeArg**.

Return Value:

Type: PtgUnsWord, PtgErr

This function attempts a conversion, as described by the DateTime function, of **DateTimeArg** to a PtgDate. If the conversion succeeds, the function returns a PtgUnsWord containing the day of the month component of **DateTimeArg.** The value is greater than or equal to 1 and less than or equal to 31. If the conversion fails, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.26 DayOfYear

The **DayOfYear** function returns the day of the year, according to the Gregorian calendar, from a value representing a date and time.

ABNF:

```
DayOfYear = val [val] sp ("1" / "2") ";DAYOFYEAR():129"
```

Required Arguments:

Name: **DateTimeArg**

Type: vAny

An argument that specifies a date and time value.

Optional Arguments:

Name: **Locale**

Type: vLanguageID

An argument that specifies an **LCID** to use when parsing **DateTimeArg**.

Return Value:

Type: PtgUnsWord, PtgErr

This function attempts a conversion, as described by the DateTime function, of **DateTimeArg** to a PtgDate. If the conversion succeeds, the function returns a PtgUnsWord containing the day of the year component from **DateTimeArg.** The value is greater than or equal to 1 and less than or equal to 366. If the conversion fails, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.27 Deg

The **Deg** function performs conversion to an angle in degrees.

ABNF:

```
Deg = val " DEG():128"
```

Required Arguments:

Name: **Angle**

Type: vDouble

An argument that specifies an angle.

Return Value:

Type: PtgAngDD

This function returns a PtgAngDD containing the value of **Angle**.

2.5.4.28 Div

The **Div** function performs a division calculation.

ABNF:

```
Div = val val " _DIV():128"
```

Required Arguments:

Name: **Arg1**

Type: vDoubleEx

An argument that specifies the first operand of the division operation.

Name: **Arg2**

Type: vDoubleEx

An argument that specifies the second operand of the division operation.

Return Value:

Type: vNum, PtgCy, PtgErr

This function returns a vNum or a PtgCy containing the result of **Arg1** divided by **Arg2**. The type of the return value is calculated by the following algorithm.

```
SET returnType = PtgNum
SET returnError = #VALUE!
SET returnDimension = 0
SET returnCurrencyID = 0
IF Arg2.Value = 0 THEN
        SET returnType = PtgErr
        SET returnError = #DIV/0
ELSE IF Arg1.Type = PtgCy AND Arg2.Type != PtgCy THEN
        SET returnType = PtgCy
        SET returnCurrencyID = currencyID of Arg1
ELSE IF Arg1.Type != PtgCy AND Arg2.Type = PtgCy THEN
        SET returnType = PtgCy
        SET returnCurrencyID = currencyID of Arg2
ELSE IF Arg1.Type = PtgCy AND Arg2.Type = PtgCy THEN
        SET returnType = PtgCy
        IF currencyID of Arg1 = currencyID of Arg2 OR currencyID of Arg2 = 1 THEN
                SET returnCurrencyID = currencyID of Arg1
        ELSE IF currencyID of Arg1 = 1 THEN
                SET returnCurrencyID = currencyID of Arg2
        ELSE
                SET returnType = PtgErr
                SET returnError = #VALUE!
        END IF
ELSE IF Arg1.Unit is a vUnitType  AND Arg2.Unit is not a vUnitType THEN
        SET returnType = Arg1.Unit
```

```
        SET returnDimension = Arg1.Dimension
ELSE IF Arg1.Unit is not a vUnitType AND Arg2.Unit is a vUnitType THEN
        SET returnType = Arg2.Unit
        SET returnDimension = Arg2.Dimension
ELSE
        IF Arg1.Unit = PtgNumPct  AND Arg2.Unit = PtgNumPct THEN
                SET returnDimension = 1
        ELSE IF Arg1.Unit = PtgNumPct THEN
                SET returnDimension = -Arg2.Dimension
        ELSE IF Arg2.Unit = PtgNumPct THEN
                SET returnDimension = Arg1.Dimension
        ELSE
                SET returnDimension = Arg1.Dimension – Arg2.Dimension
        END IF
        IF Arg1.Unit is a vUnitType THEN
                SET returnType = Arg1.Unit
        ELSE
                SET returnType = Arg2.Unit
        END IF
END IF
IF returnType = PtgCy THEN
        Return type is PtgCy with currencyID = returnCurrencyID
ELSE IF returnType = PtgErr THEN
        Return type is PtgErr with error code of returnError
ELSE IF (returnType = PtgAcre  OR returnType = PtgHectare) AND returnDimension != 2 THEN
        Return type is PtgNumMultiDim  with unit = PtgNumDft  and dimension = returnDimension
ELSE IF returnDimension = 0 THEN
        Return type is PtgNum
ELSE IF returnDimension = 1 THEN
        Return type is returnType
ELSE
        Return type is PtgNumMultiDim with unit = returnType and dimension = returnDimension
END IF
```

2.5.4.29 EEQ

The **EEQ** function calculates if **Arg1** is equal to **Arg2**.

ABNF:

```
EEQ = val val " _EEQ():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Arg1** is equal to **Arg2**, and a value of FALSE otherwise.

2.5.4.30 EGE

The **EGE** function calculates if **Arg1** is greater than or equal to **Arg2**.

ABNF:

```
EGE = val val " _EGE():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Arg1** is greater than or equal to **Arg2**, and a value of FALSE otherwise.

2.5.4.31 EGT

The **EGT** function calculates if **Arg1** is greater than **Arg2**.

ABNF:

```
EGT = val val " _EGT():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Arg1** is greater than **Arg2**, and a value of FALSE otherwise.

2.5.4.32 ELE

The **ELE** function calculates if **Arg1** is less than or equal to **Arg2**.

ABNF:

```
ELE = val val " _ELE():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Arg1** is less than or equal to **Arg2**, and a value of FALSE otherwise.

2.5.4.33 ELT

The **ELT** function calculates if **Arg1** is less than **Arg2**.

ABNF:

```
ELT = val val " _ELT():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Arg1** is less than **Arg2**, and a value of FALSE otherwise.

2.5.4.34 ENE

The **ENE** function calculates if **Arg1** is not equal to **Arg2**.

ABNF:

```
ENE = val val " _ENE():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Arg1** is not equal to **Arg2**, and a value of FALSE otherwise.

2.5.4.35 FEQ

The **FEQ** function calculates if the absolute value of the difference between **Arg1** and **Arg2** is less than or equal to 1E-9 (0.000000001).

ABNF:

```
FEQ = val val " _FEQ():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if the absolute value of the difference between **Arg1** and **Arg2** is less than or equal to 1E-9, and a value of FALSE otherwise.

2.5.4.36 FGE

The **FGE** function calculates if the difference between **Arg1** and **Arg2** is greater than or equal to -1E-9 (-0.000000001).

ABNF:

```
FGE = val val " _FGE():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if the difference between **Arg1** and **Arg2** is greater than or equal to -1E-9, and a value of FALSE otherwise.

2.5.4.37 FGT

The **FGT** function calculates if **Arg1** is greater than **Arg2** by at least the amount of 2^{-30}.

ABNF:

```
FGT = val val " _FGT():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Arg1** is greater than **Arg2** by at least the amount of 2^{-30}, and a value of FALSE otherwise.

2.5.4.38 Find

The **Find** function performs a search for the first instance of a text string within another text string.

ABNF:

```
Find = val val [val [val]] sp ("2" / "3" / "4") ";FIND():129"
```

Required Arguments:

Name: **FindText**

Type: vString

An argument that specifies the string to be found.

Name: **WithinText**

Type: vString

An argument that specifies the string to search within.

Optional Arguments:

Name: **StartNum**

Type: vUnsignedInt

An argument that specifies the one-based position in the **WithinText** string at which this function starts a search. The default value is one.

Name: **IgnoreCase**

Type: vBoolean

An argument that specifies whether the search is case insensitive. A value of TRUE specifies that case is ignored. The default value is FALSE.

Return Value:

Type: PtgUnsWord, PtgErr

This function returns a PtgUnsWord containing the one-based position in **WithinText** at which **FindText** is found. If **FindText** is empty and **StartNum** is less than or equal to the number of characters in **WithinText**, the function returns a PtgUnsWord containing the value of **StartNum**. If **FindText** is empty and **StartNum** is greater than the number of characters in **WithinText**, the function returns a PtgUnsword containing the value of the number of characters in **WithinText** plus one. If **FindText** is not found, or, if **StartNum** is equal to zero or greater than the number of characters in **WithinText**, the function returns a PtgErr with an error code of #VALUE!. The search is performed according to .NET globalization **rules** based on the **LCID** in the **DocumentLanguage** field of the Properties **GlobalElement** in the App XML Part of the document. For more information about .NET globalization rules, see [MSDN-ENCLOC].

2.5.4.39 FLE

The **FLE** function calculates if the difference between **Arg1** and **Arg2** is less than or equal to 1E-9 (0.000000001).

ABNF:

```
FLE = val val " _FLE():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if the difference between **Arg1** and **Arg2** is less than or equal to 1E-9, and a value of FALSE otherwise.

2.5.4.40 Floor

The **Floor** function performs a floor calculation.

ABNF:

```
Floor = val [val] sp ("1" / "2") ";FLOOR():129"
```

Required Arguments:

Name: **Number**

Type: vDouble

An argument that specifies the value to be rounded.

Optional Arguments:

Name: **Multiple**

Type: vDouble

An argument that specifies the rounding increment.

Return Value:

Type: vNum, PtgErr

If **Multiple** is not specified, the function returns a vNum containing the largest integer less than or equal to **Number**. If **Multiple** is greater than zero, the function returns a vNum containing the largest multiple of **Multiple** less than or equal to **Number**. If **Multiple** is less than zero, the function returns a vNum containing the smallest multiple of **Multiple** greater than or equal to **Number**. If **Number** is equal to zero or **Multiple** is equal to zero, the value is zero. The unit of the return value is equal to the unit of **Number**. If **Number** and **Multiple** do not have the same sign, the function returns a PtgErr with an error code of #NUM!.

2.5.4.41 FLT

The **FLT** function calculates if **Arg1** is less than **Arg2** by at least the amount of 2^{-30}.

ABNF:

```
FLT = val val " _FLT():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Arg1** is less than **Arg2** by at least the amount of 2^{-30}, and a value of FALSE otherwise.

2.5.4.42 FNE

The **FNE** function calculates if **Arg1** differs from **Arg2** by at least the amount of 2^{-30}.

ABNF:

```
FNE = val val " _FNE():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDouble

An argument that specifies the second operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Arg1** differs from **Arg2** by at least the amount of 2^{-30}, and a value of FALSE otherwise.

2.5.4.43 FormatEx

The **FormatEx** function formats a value as a string.

ABNF:

```
FormatEx = val val val val val val " FORMATEX():128"
```

Required Arguments:

Name: **FormatValue**

Type: vDoubleEx

An argument that specifies the value to be formatted.

Name: **Format**

Type: PtgStr1

An argument that specifies the formatting information used to format **FormatValue**. It MUST have the following format.

"*Formatting*@*ServerAction*@*UnitLabel*@*CurrencyID*"

The elements of the string are separated by "@" and are defined in the following paragraphs.

Formatting: A vFormatString that specifies the formatting information.

If *Formatting* is an empty string, a default value from the following table is used.

Value	Meaning
"{0} {1}"	Default string format. This is used if the type of the source token of **FormatValue** is PtgStr1, PtgEDay, PtgEHour, PtgEMin, PtgESec, PtgEWeek, or PtgTDurDft.
"{0:0.####} {1}"	Default numeric format for numbers with less than 16 digits to the left of the decimal point. This is used if the type of the source token of **FormatValue** is PtgBool, PtgCy, PtgMissArg, PtgNum, PtgUnsWord, or a vUnitType other than PtgDate, PtgEDay, PtgEHour, PtgEMin, PtgESec, PtgEWeek, and PtgTDurDft.
"{0:0.####e0} {1}"	Default numeric format for numbers with 16 or more digits to the left of the decimal point. This is used if the type of the source token of **FormatValue** is PtgBool, PtgCy, PtgMissArg, PtgNum, PtgUnsWord, or a vUnitType other than PtgDate, PtgEDay, PtgEHour, PtgEMin, PtgESec, PtgEWeek, and PtgTDurDft.
"G"	Default date and time format when the time value is not equal to zero. This is used if the type of the source token of **FormatValue** is PtgDate.

Value	Meaning
"d"	Default date and time format when the time value is equal to zero. This is used if the type of the source token of **FormatValue** is PtgDate.

If *Formatting* is not empty and **FormatValue** is a PtgDate, *Formatting* MUST be a vFormatString with valid syntax for formatting a date and time value. If *Formatting* is not empty and **FormatValue** is not a PtgDate, *Formatting* MUST be a vFormatString with valid syntax for formatting a numeric value.

ServerAction: A vServerAction that specifies the custom formatting to apply to a string after all other formatting has been performed.

UnitLabel: A vUnitLabel that specifies the **rules** for formatting the unit of the resulting string.

CurrencyID: A vCurrencyID that specifies the currency identifier.

Name: **SrcUnit**

Type: vUnitType

An argument that specifies the unit of **FormatValue**. The value is unused and MUST be ignored. If **FormatValue** is a vUnitType, this argument is unused and MUST be ignored.

Name: **DstUnit**

Type: vNum

An argument that specifies the unit displayed in the resulting string. The value is unused and MUST be ignored. If the type is PtgNum, the type of **SrcUnit** is used.

Name: **LangID**

Type: vLanguageID

An argument that specifies an **LCID** used to format **FormatValue**.

Name: **CalID**

Type: vCalendar

An argument that specifies a calendar system used to format **FormatValue**. If **FormatValue** is not a PtgDate, this argument is unused and MUST be ignored.

Return Value:

Type: PtgStr1, PtgErr

This function returns a PtgStr1 containing **FormatValue**, converted into the unit specified by **DstUnit**, formatted as a string as specified by **Format**, **LangID**, and **CalID**.

FormatValue is converted into the unit specified by **DstUnit** as follows. If **FormatValue** is not a vUnitType, the value of **FormatValue** is multiplied by a factor that converts the unit of **SrcUnit** into the custom internal unit type associated with the type of **SrcUnit**. The resulting value, or the value of **FormatValue** if **FormatValue** is a vUnitType, is multiplied by a factor that converts the custom internal unit type associated with the type of **DstUnit** into the unit of **DstUnit**

If **FormatValue** is a PtgDate, **FormatValue** is formatted using date and time format strings, as described in [MSDN-FormattingTypes]. If **FormatValue** is not a PtgDate, the value of **FormatValue** is formatted using numeric format strings, as described in [MSDN-FormattingTypes].

If the formatting does not succeed or an argument is invalid, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.44 Hour

The **Hour** function returns the hour from a value representing a date and time.

ABNF:

```
Hour = val [val] sp ("1" / "2") ";HOUR():129"
```

Required Arguments:

Name: **DateTimeArg**

Type: vAny

An argument that specifies a date and time value.

Optional Arguments:

Name: **Locale**

Type: vLanguageID

An argument that specifies an **LCID** to use when parsing **DateTimeArg**.

Return Value:

Type: PtgUnsWord, PtgErr

This function attempts a conversion, as described by the DateTime function, of **DateTimeArg** to a PtgDate. If the conversion succeeds, the function returns a PtgUnsWord containing the hour component from **DateTimeArg.** The value is greater than or equal to zero and less than or equal to 23. If the conversion fails, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.45 HSL

The **HSL** function transforms a color specified in the **hue-saturation-luminance (HSL) color space** into a color specified in the **RGB** color space. **HSL** component values are in the range 0 to 240, inclusive.

ABNF:

```
HSL = val val val " HSL():128"
```

Required Arguments:

Name: **Hue**

Type: vUnsignedInt

An argument that specifies the hue component.

Name: **Saturation**

Type: vUnsignedInt

An argument that specifies the saturation component.

Name: **Luminance**

Type: vUnsignedInt

An argument that specifies the luminance component.

Return Value:

Type: PtgColorRGB

This function returns a PtgColorRGB containing the color value specified by **Hue**, **Saturation** and **Luminance**. If **Hue** is greater than 240, this function sets the value to 240 and performs the operation.

2.5.4.46 Hue

The **Hue** function calculates the hue component, as specified in the **HSL color space**, of a color value.

ABNF:

```
Hue = val " HUE():128"
```

Required Arguments:

Name: **Arg1**

Type: vColor

An argument that specifies the color.

Return Value:

Type: PtgUnsWord

This function returns a PtgUnsWord containing the hue of **Arg1**. The value is less than or equal to 240.

2.5.4.47 HueDiff

The **HueDiff** function calculates the difference in hue, as specified in the **HSL color space**, between two color values.

ABNF:

```
HueDiff = val val " HUEDIFF():128"
```

Required Arguments:

Name: **Arg1**

Type: vColor

An argument that specifies the first color.

Name: **Arg2**

Type: vColor

An argument that specifies the second color.

Return Value:

Type: PtgNum

This function returns a PtgNum containing a value that is equal to the hue of **Arg1** minus the hue of **Arg2**. This value is greater than or equal to -240 and less than or equal to 240.

2.5.4.48 Index

The **Index** function performs a search of a delimited text string for the substring specified by a **zero-based index**.

ABNF:

```
Index = val val [val [val]] sp ("2" / "3" / "4") ";INDEX():129"
```

Required Arguments:

Name: **IndexPosition**

Type: vSignedInt

An argument that specifies a zero-based index.

Name: **List**

Type: vString

An argument that specifies a string consisting of a list of delimited substrings. If the list begins and/or ends with **Delimiter**, an empty string is assumed to exist at the beginning and/or end of **List**, respectively. Consecutive delimiters indicate an empty string between them.

Optional Arguments:

Name: **Delimiter**

Type: vString

An argument that specifies the delimiter that separates substrings inside **List**. If this argument is an empty string, the default value ";" is used.

Name: **ErrorValue**

Type: vString

An argument that specifies a string to be returned if **Index** is out of range. The default value is an empty string.

Return Value:

Type: PtgStr1

This function returns a PtgStr1 equal to the substring at the zero-based index location **IndexPosition** in **List**, which is delimited by **Delimiter**. If **IndexPosition** is less than zero or greater than or equal to the number of items in **List**, the function returns a PtgStr1 that specifies **ErrorValue**. The search is performed according to .NET globalization **rules** based on the **LCID** in the **DocumentLanguage** field

of the Properties **GlobalElement** in the App XML Part of the document. For more information about .NET globalization rules, see [MSDN-ENCLOC].

2.5.4.49 Int

The **Int** function performs a rounding calculation down to the next lesser integer value.

ABNF:

```
Int = val " INT():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the value to be rounded.

Return Value:

Type: vNum

This function returns a vNum containing the next lesser integer value if **Arg1** is not an integer. The unit of the return value is equal to the unit of **Arg1.**

2.5.4.50 IntersectX

The **IntersectX** function calculates the x-coordinate of the point where two lines intersect. Each line is defined by a point on the line and an angle.

ABNF:

```
IntersectX = val val val val val val " INTERSECTX():128"
```

Required Arguments:

Name: **X1**

Type: vDouble

An argument that specifies the x-coordinate of the point defining the first line.

Name: **Y1**

Type: vDouble

An argument that specifies the y-coordinate of the point defining the first line.

Name: **Angle1**

Type: vDouble

An argument that specifies the angle of the first line relative to the positive x-axis.

Name: **X2**

Type: vDouble

An argument that specifies the x-coordinate of the point defining the second line.

Name: **Y2**

Type: vDouble

An argument that specifies the y-coordinate of the point defining the second line.

Name: **Angle2**

Type: vDouble

An argument that specifies the angle of the second line relative to the positive x-axis.

Return Value:

Type: vNum, PtgErr

This function returns a vNum containing the x-coordinate of the point where two lines intersect. If any of the arguments **X1**, **Y1**, **X2**, **Y2** is a vUnitType, the unit of the return value is equal to the unit of the first of those arguments that is a vUnitType. If none of those arguments is a vUnitType, the function returns a PtgNum. If **Angle1** equals **Angle2**, the function returns a PtgErr with an error code of #DIV/0.

2.5.4.51 IntersectY

The **IntersectY** function calculates the y-coordinate of the point where two lines intersect. Each line is defined by a point on the line and an angle.

ABNF:

```
IntersectY = val val val val val val " INTERSECTY():128"
```

Required Arguments:

Name: **X1**

Type: vDouble

An argument that specifies the x-coordinate of the point defining the first line.

Name: **Y1**

Type: vDouble

An argument that specifies the y-coordinate of the point defining the first line.

Name: **Angle1**

Type: vDouble

An argument that specifies the angle of the first line relative to the positive x-axis.

Name: **X2**

Type: vDouble

An argument that specifies the x-coordinate of the point defining the second line.

Name: **Y2**

Type: vDouble

An argument that specifies the y-coordinate of the point defining the second line.

Name: **Angle2**

Type: vDouble

An argument that specifies the angle of the second line relative to the positive x-axis.

Return Value:

Type: vNum, PtgErr

This function returns a vNum containing the y-coordinate of the point where two lines intersect. If any of the arguments **X1**, **Y1**, **X2**, **Y2** is a vUnitType, the unit of the return value is equal to the unit of the first of those arguments that is a vUnitType. If none of those arguments is a vUnitType, the function returns a PtgNum. If **Angle1** equals **Angle2**, the function returns a PtgErr with an error code of #DIV/0.

2.5.4.52 Intup

The **Intup** function performs a rounding calculation up to the next greater integer value.

ABNF:

```
Intup = val " INTUP():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the value to be rounded.

Return Value:

Type: vNum

This function returns a vNum containing the next greater integer value if **Arg1** is not an integer. The unit of the return value is equal to the unit of **Arg1**.

2.5.4.53 IsErr

The **IsErr** function calculates if the operand is a PtgErr that does not have an error code of #N/A.

ABNF:

```
IsErr = val " ISERR():128"
```

Required Arguments:

Name: **Token**

Type: vAny

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Token** is a PtgErr and the value of **Token** is not equal to #N/A, and a value of FALSE otherwise.

2.5.4.54 IsErrNA

The **IsErrNA** function calculates if the operand is a PtgErr with an error code of #N/A.

ABNF:

```
IsErrNa = val " ISERRNA():128"
```

Required Arguments:

Name: **Token**

Type: vAny

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Token** is a PtgErr and the value of **Token** is equal to #N/A, and a value of FALSE otherwise.

2.5.4.55 IsError

The **IsError** function calculates if the operand is a PtgErr.

ABNF:

```
IsError = val " ISERROR():128"
```

Required Arguments:

Name: **Token**

Type: vAny

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Token** is a PtgErr, and a value of FALSE otherwise.

2.5.4.56 IsErrValue

The **IsErrValue** function calculates if the operand is a PtgErr with an error code of #VALUE!.

ABNF:

```
IsErrValue = val " ISERRVALUE():128"
```

Required Arguments:

Name: **Token**

Type: vAny

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgBool

This function returns a PtgBool with a value of TRUE if **Token** is a PtgErr and the value of **Token** is equal to #VALUE!, and a value of FALSE otherwise.

2.5.4.57 Left

The **Left** function returns the first character or characters in a text string.

ABNF:

```
Left = val [val] sp ("1" / "2") ";LEFT():129"
```

Required Arguments:

Name: **Text**

Type: vString

An argument that specifies the text string.

Optional Arguments:

Name: **NumChars**

Type: vSignedLong

An argument that specifies the number of characters to be returned. The default value is one.

Return Value:

Type: PtgStr1

This function returns a PtgStr1 containing the first **NumChars** characters of **Text.** If **NumChars** is less than zero or greater than the number of characters in **Text**, the function returns a PtgStr1 containing **Text**.

2.5.4.58 Len

The **Len** function performs a calculation of the length of a string.

ABNF:

```
Len = val " LEN():128"
```

Required Arguments:

Name: **Text**

Type: vString

An argument that specifies a string.

Return Value:

Type: PtgUnsWord

This function returns a PtgUnsWord containing the number of characters in **Text**.

2.5.4.59 Ln

The **Ln** function performs the natural logarithm calculation.

ABNF:

```
Ln = val " LN():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: vNum, PtgErr

This function returns a vNum containing the natural logarithm of **Arg1**. The unit of the return value is equal to the unit of **Arg1**. If **Arg1** is less than or equal to zero, the function returns a PtgErr with an error code of #NUM!.

2.5.4.60 Log10

The **Log10** function performs the base 10 logarithm calculation.

ABNF:

```
Log10 = val " LOG10():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: vNum, PtgErr

This function returns a vNum containing the base 10 logarithm of **Arg1**. The unit of the return value is equal to the unit of **Arg1**. If the value of **Arg1** is less than or equal to zero, the function returns a PtgErr with an error code of #NUM!.

2.5.4.61 Lookup

The **Lookup** function calculates the index location of a substring within a delimited text string.

ABNF:

```
Lookup = val val [val] sp ("2" / "3") ";LOOKUP():129"
```

Required Arguments:

Name: **Key**

Type: vString

An argument that specifies the substring to find.

Name: **List**

Type: vString

An argument that specifies a string consisting of a list of delimited substrings. If the list begins and/or ends with **Delimiter**, an empty string is assumed to exist at the beginning and/or end of **List**, respectively. Consecutive delimiters indicate an empty string between them.

Optional Arguments:

Name: **Delimiter**

Type: vString

An argument that specifies the delimiter that separates substrings inside **List**. If this argument is an empty string, the default value ";" is used.

Return Value:

Type: PtgNum

This function returns a PtgNum containing the **zero-based index** location of the substring in **List** that matches **Key**. The leading and trailing spaces of substrings in **List** are ignored for the comparison. If **Key** contains **Delimiter** or if **Key** is not found in **List**, the function returns a PtgNum containing a value of -1. The comparison is performed according to .NET globalization **rules** based on the **LCID** in the **DocumentLanguage** field of the Properties **GlobalElement** in the App XML Part of the document. For more information about .NET globalization rules, see [MSDN-ENCLOC].

2.5.4.62 Lower

The **Lower** function performs lower case conversion.

ABNF:

```
Lower = val " LOWER():128"
```

Required Arguments:

Name: **Arg1**

Type: vString

An argument that specifies the string to convert.

Return Value:

Type: PtgStr1

This function returns a PtgStr1 containing the string converted to lower case. The conversion is performed according to .NET globalization **rules** based on the **LCID** in the **DocumentLanguage** field of the Properties **GlobalElement** in the App XML Part of the document. For more information about .NET globalization rules, see [MSDN-ENCLOC].

2.5.4.63 Lum

The **Lum** function calculates the luminance component, as specified in the **HSL color space**, of a color value.

ABNF:

```
Lum = val " LUM():128"
```

Required Arguments:

Name: **Arg1**

Type: vColor

An argument that specifies the color.

Return Value:

Type: PtgUnsWord

This function returns a PtgUnsWord containing the luminance of **Arg1**. The value is less than or equal to 240.

2.5.4.64 LumDiff

The **LumDiff** function calculates the difference in luminance, as specified in the **HSL color space**, between two color values.

ABNF:

```
LumDiff = val val " LUMDIFF():128"
```

Required Arguments:

Name: **Arg1**

Type: vColor

An argument that specifies the first color.

Name: **Arg2**

Type: vColor

An argument that specifies the second color.

Return Value:

Type: PtgNum

This function returns a PtgNum containing a value that is equal to the luminance of **Arg1** minus the luminance of **Arg2**. This value is greater than or equal to -240 and less than or equal to 240.

2.5.4.65 Magnitude

The **Magnitude** function performs a magnitude calculation of a vector.

ABNF:

```
Magnitude = val val val val " MAGNITUDE():128"
```

Required Arguments:

Name: **ConstantRise**

Type: vDouble

An argument that specifies the constant factor for the rise.

Name: **Rise**

Type: vDouble

An argument that specifies the rise of the vector.

Name: **ConstantRun**

Type: vDouble

An argument that specifies the constant factor for the run.

Name: **Run**

Type: vDouble

An argument that specifies the run of the vector.

Return Value:

Type: PtgNum, PtgErr

The function returns a PtgNum containing the calculated magnitude. The magnitude calculation is performed using the following formula:

SQRT((ConstantRise*Rise)^2 + (ConstantRun*Run)^2))

If **Rise** or **Run** is a PtgNumMultiDim, the function returns a PtgErr with an error code of #DIM!.

2.5.4.66 Max

The **Max** function performs a search for the largest number from a list of operands.

ABNF:

```
Max = val *val sp unsigned-integer-value ";MAX():129"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand.

Optional Arguments:

Zero or more operands of type vDouble.

Return Value:

Type: vNum

This function returns a vNum containing the first occurrence of the largest operand. The unit of the return value is equal to the unit of that operand.

2.5.4.67 Mid

The **Mid** function returns part of a string, starting at the position specified, based on the number of characters specified.

ABNF:

```
Mid = val val val " MID():128"
```

Required Arguments:

Name: **Text**

Type: vString

An argument that specifies the text string.

Name: **StartPos**

Type: vSignedLong

An argument that specifies the one-based index in **Text** which represents the beginning of the substring to be returned.

Name: **NumChars**

Type: vSignedLong

An argument that specifies the number of characters to be returned.

Return Value:

Type: PtgStr1, PtgErr

This function returns a PtgStr1 containing a substring of **Text**, starting from **StartPos** up to the character at **StartPos** plus **NumChars** minus one. If **StartPos** is zero, the function returns a PtgErr with an error code of #VALUE!. If **StartPos** is less than zero or greater than the number of characters in **Text**, the function returns a PtgStr1 containing an empty string. If **StartPos** plus **NumChars** minus one exceeds the length of **Text**, or if **NumChars** is less than zero, the function returns a PtgStr1 containing the substring of **Text** starting from **StartPos** up to the end of **Text**.

2.5.4.68 Min

The **Min** function performs a search for the smallest number from a list of operands.

ABNF:

```
Min = val *val sp unsigned-integer-value ";MIN():129"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand.

Optional Arguments:

Zero or more operands of type vDouble.

Return Value:

Type: vNum

This function returns a vNum containing the first occurrence of the smallest operand. The unit of the return value is equal to the unit of that operand.

2.5.4.69 Minute

The **Minute** function returns the minute from a value representing a date and time.

ABNF:

```
Minute = val [val] sp ("1" / "2") ";MINUTE():129"
```

Required Arguments:

Name: **DateTimeArg**

Type: vAny

An argument that specifies a date and time value.

Optional Arguments:

Name: **Locale**

Type: vLanguageID

An argument that specifies an **LCID** to use when parsing **DateTimeArg**.

Return Value:

Type: PtgUnsWord, PtgErr

This function attempts a conversion, as described by the DateTime function, of **DateTimeArg** to a PtgDate. If the conversion succeeds, the function returns a PtgUnsWord containing the minute component from **DateTimeArg.** The value is greater than or equal to zero and less than or equal to 59. If the conversion fails, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.70 Modulus

The **Modulus** function performs a modulus calculation.

ABNF:

```
Modulus = val val " MODULUS():128"
```

Required Arguments:

Name: **Number**

Type: vDouble

An argument that specifies the number to be divided by **Divisor**.

Name: **Divisor**

Type: vFloat

An argument that specifies the divisor of the calculation.

Return Value:

Type: vNum, PtgErr

This function returns a vNum containing the remainder when **Number** is divided by **Divisor**. If **Number** is a vNum, the unit of the return value is equal to the unit of **Number**; otherwise the function returns a PtgNum. If **Divisor** is zero, the function returns a PtgErr with an error code of #DIV/0.

2.5.4.71 Month

The **Month** function returns the month, according to the Gregorian calendar, from a value representing a date and time.

ABNF:

```
Month = val [val] sp ("1" / "2") ";MONTH():129"
```

Required Arguments:

Name: **DateTimeArg**

Type: vAny

An argument that specifies a date and time value.

Optional Arguments:

Name: **Locale**

Type: vLanguageID

An argument that specifies an **LCID** to use when parsing **DateTimeArg**.

Return Value:

Type: PtgUnsWord, PtgErr

This function attempts a conversion, as described by the DateTime function, of **DateTimeArg** to a PtgDate. If the conversion succeeds, the function returns a PtgUnsWord containing the month component from **DateTimeArg**. The value is greater than or equal to 1 and less than or equal to 12. If the conversion fails, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.72 MsoShade

The **MsoShade** function performs a modification to a color by decreasing the luminance component, as specified in the **HSL color space**, by a percentage.

ABNF:

```
MsoShade = val val " MSOSHADE():128"
```

Required Arguments:

Name: **Color**

Type: vColor

An argument that specifies the color.

Name: **Delta**

Type: vSignedInt

An argument that specifies the percentage to decrease the luminance of **Color**. If the value is less than zero, the luminance is increased.

Return Value:

Type: PtgColorRGB

This function returns a PtgColorRGB containing the shaded color. If **Delta** is greater than zero, the luminance of **Color** is decreased by the percentage specified by the absolute value of **Delta**. If **Delta** is less than zero, the luminance of **Color** is increased by the following amount: the percentage specified by **Delta** multiplied by the result of 240 minus the luminance of **Color**. If **Delta** is less than -100, the function sets the value to -100 and performs the operation. If **Delta** is greater than 100, the function sets the value to 100 and performs the operation.

2.5.4.73 MsoTint

The **MsoTint** function modifies a color by increasing the luminance component, as specified in the **HSL color space**, by a percentage.

ABNF:

```
MsoTint = val val " MSOTINT():128"
```

Required Arguments:

Name: **Color**

Type: vColor

An argument that specifies the color.

Name: **Delta**

Type: vSignedInt

An argument that specifies the percentage to increase the luminance of **Color**. If the value is less than zero, the luminance is decreased.

Return Value:

Type: PtgColorRGB

This function returns a PtgColorRGB containing the tinted color. If **Delta** is greater than zero, the luminance of **Color** is increased by the following amount: the percentage specified by **Delta** multiplied by the result of 240 minus the luminance of **Color**. If **Delta** is less than zero, the luminance of **Color** is decreased by the percentage specified by the absolute value of **Delta**. If **Delta** is less than -100, the function sets the value to -100 and performs the operation. If **Delta** is greater than 100, the function sets the value to 100 and performs the operation.

2.5.4.74 Mul

The **Mul** function performs a multiplication calculation.

ABNF:

```
Mul = val val " _MUL():128"
```

Required Arguments:

Name: **Arg1**

Type: vDoubleEx

An argument that specifies the first operand of the multiplication operation.

Name: **Arg2**

Type: vDoubleEx

An argument that specifies the second operand of the multiplication operation.

Return Value:

Type: vNum, PtgCy, PtgErr

This function returns a vNum or a PtgCy containing **Arg1** multiplied by **Arg2**. The type of the return value is calculated by the following algorithm.
SET return type = PtgNum
SET returnDimension = 0

```
SET returnCurrencyID = 0
IF Arg1.Type = PtgCy AND Arg2.Type != PtgCy THEN
        SET returnType = PtgCy
        SET returnCurrencyID = currencyID of Arg1
ELSE IF Arg1.Type != PtgCy AND Arg2.Type = PtgCy THEN
        SET returnType = PtgCy
        SET returnCurrencyID = currencyID of Arg2
ELSE IF Arg1.Type = PtgCy AND Arg2.Type = PtgCy THEN
        SET returnType = PtgCy
        IF currencyID of Arg1 = currencyID of Arg2 OR currencyID of Arg2 = 1 THEN
                SET returnCurrencyID = currencyID of Arg1
        ELSE IF currencyID of Arg1 = 1 THEN
                SET returnCurrencyID = currencyID of Arg2
        ELSE
                SET returnType = PtgErr
        END IF
ELSE IF Arg1.Unit = PtgNumPct  AND Arg2.Unit = PtgNumPct
        SET returnType = PtgNumPct
        SET returnDimension = 1
ELSE IF Arg1.Unit = PtgNumPct
        SET returnType = PtgNumPct
        SET returnDimension = Arg2.dimension
ELSE IF Arg2.Unit = PtgNumPct
        IF Arg1.Unit is a vUnitType
                SET returnType = Arg1.Unit
                SET returnDimension = Arg1.Dimension
        ELSE
                SET returnType = Arg2.Unit
                SET returnDimension = Arg1.Dimension
        END IF
ELSE IF Arg1.Unit is a vUnitType THEN
        SET returnType = Arg1.Unit
        SET returnDimension = Arg1.Dimension + Arg2.Dimension
ELSE IF Arg2.Unit is a vUnitType THEN
        SET returnType = Arg2.Unit
        SET returnDimension = Arg1.Dimension + Arg2.Dimension
END IF
IF returnType = PtgCy THEN
        Return type is PtgCy with currencyID = returnCurrencyID
ELSE IF returnType = PtgErr THEN
        Return type is PtgErr with error code of #VALUE!
ELSE IF (returnType = PtgAcre OR returnType = PtgHectare) AND returnDimension != 2 THEN
        Return type is PtgNumMultiDim with unit = PtgNumDft  and dimension = returnDimension
ELSE IF returnDimension = 0 THEN
        Return type is PtgNum
ELSE IF returnDimension = 1 THEN
        Return type is returnType
ELSE
        Return type is PtgNumMultiDim with unit = returnType and dimension = returnDimension
END IF
```

2.5.4.75 Not

The **Not** function performs the **Boolean** NOT operation.

ABNF:

```
Not = val " NOT():128"
```

Required Arguments:

Name: **Value**

Type: vBoolean

An argument that specifies the operand.

Return Value:

Type: PtgBool

This function returns a PtgBool containing the value of the Boolean NOT operation on **Value**.

2.5.4.76 Now

The **Now** function returns the current date and time.

ABNF:

```
Now = "NOW():128"
```

Return Value:

Type: PtgDate

This function returns a PtgDate containing the value of current date and time.

2.5.4.77 Or

The **Or** function converts an argument to a **Boolean** value according to the conversion specified by vBoolean.

ABNF:

```
Or = val " OR():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand.

Return Value:

Type: PtgBool

This function returns a PtgBool containing the value of **Arg1**.

2.5.4.78 Pct

The **Pct** function performs a percent conversion on the operand.

ABNF:

```
Pct = val " _PCT():128"
```

Required Arguments:

Name: **Value**

Type: vDouble

An argument that specifies the operand that will be converted to a percent.

Return Value:

Type: PtgNumPct

This function returns a PtgNumPct containing **Value** / 100.

2.5.4.79 Pi

The **Pi** function returns the mathematical constant pi.

ABNF:

```
Pi = "PI():128"
```

Return Value:

Type: PtgNum

This function returns PtgNum a containing the value of the mathematical constant pi.

2.5.4.80 Pnt

The **Pnt** function MUST be ignored.

ABNF:

```
Pnt = val val " PNT():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

This argument is unused and MUST be ignored.

Name: **Arg2**

Type: vDouble

This argument is unused and MUST be ignored.

Return Value:

Type: PtgNoOp

This function returns a PtgNoOp.

2.5.4.81 Pow

The **Pow** function performs an exponentiation calculation.

ABNF:

```
Pow = val val " POW():128"
```

Required Arguments:

Name: **Base**

Type: vDouble

An argument that specifies the number to be raised to the power of **Exponent**.

Name: **Exponent**

Type: vDouble

An argument that specifies the exponent by which to raise **Base**.

Return Value:

Type: vNum, PtgErr

This function returns a vNum or PtgErr containing **Base** raised to the power of **Exponent**. The type of the return value is calculated by the following algorithm.

```
SET returnType = PtgNum
SET returnError = #NUM!
SET returnDimension = 0
IF Base.Value = 0 THEN
        IF Exponent.Value = 0 THEN
                SET returnType = PtgErr
                SET returnError = #NUM!
        ELSE IF Exponent.Value < 0 THEN
                SET returnType = PtgErr
                SET returnError = #DIV/0
        END IF
ELSE IF Base.Value < 0 AND Exponent is not an integer THEN
        SET returnType = PtgErr
        SET returnError = #DIM!
ELSE IF Base.Unit = PtgNumPct  THEN
        SET returnType = PtgNumPct
        SET returnDimension = 1
ELSE
        SET returnType = Base.Unit
        SET returnDimension = Base.Dimension * Exponent.Value
END IF
IF returnType = PtgErr THEN
        Return type is PtgErr with error code of returnError
ELSE IF returnDimension = 0 THEN
        Return type is PtgNum
ELSE IF returnDimension = 1 THEN
        Return type is returnType
```

ELSE
Return type is PtgNumMultiDim with unit = returnType and dimension = returnDimension
END IF

2.5.4.82 Rad

The **Rad** function converts a value to an angle in radians.

ABNF:

```
Rad = val " RAD():128"
```

Required Arguments:

Name: **Angle**

Type: vDouble

An argument that specifies an angle.

Return Value:

Type: PtgAngRad

This function returns a PtgAngRad. If **Angle** is a vAngle, the value is equal to the value of **Angle**. Otherwise, **Angle** is interpreted as an angle in degrees and the return value is equal to:

pi / 180.0 * value of **Angle**

2.5.4.83 Rand

The **Rand** function generates a random number.

ABNF:

```
Rand = "RAND():128"
```

Return Value:

Type: PtgNum

This function returns a PtgNum containing a random double precision **floating-point number** greater than or equal to 0.0 and less than 1.0.

2.5.4.84 Replace

The **Replace** function replaces part of a text string with another text string.

ABNF:

```
Replace = val val val val " REPLACE():128"
```

Required Arguments:

Name: **SourceText**

Type: vString

An argument that specifies the string to perform replacement on.

Name: **StartPos**

Type: vSignedLong

An argument that specifies the one-based starting position in **SourceText** where replacement begins.

Name: **NumChars**

Type: vSignedLong

An argument that specifies the number of characters in **SourceText** to be replaced by **ReplaceText**.

Name: **ReplaceText**

Type: vString

An argument that specifies the string to use for replacement.

Return Value:

Type: PtgStr1, PtgErr

This function returns a PtgStr1 containing **SourceText** modified with the replaced text. If **StartPos** is less than or equal to zero or greater than the number of characters in **SourceText**, the function returns a PtgStr1 containing **SourceText** with **ReplaceText** appended at the end. If **StartPos** plus **NumChars** minus one exceeds the length of **SourceText**, or if **NumChars** is less than zero, the function returns a PtgStr1 containing **SourceText** truncated starting from **StartPos** and with **ReplaceText** appended at the end.

2.5.4.85 RGB

The **RGB** function calculates an **RGB** color value as a combination of red, green and blue components.

ABNF:

```
RGB = val val val " RGB():128"
```

Required Arguments:

Name: **Red**

Type: vUnsignedInt

An argument that specifies the intensity of red.

Name: **Green**

Type: vUnsignedInt

An argument that specifies the intensity of green.

Name: **Blue**

Type: vUnsignedInt

An argument that specifies the intensity of blue.

Return Value:

Type: PtgColorRGB

This function returns a PtgColorRGB containing a color value. The byte that specifies red in the PtgColorRGB is equal to **Red**. The byte that specifies green in the PtgColorRGB is equal to **Green**. The byte that specifies blue in the PtgColorRGB is equal to **Blue**. If the value of any of the arguments is greater than 0xFF, the function sets the argument's value to the result of the bitwise AND operation between the original value and 0xFF, before performing the operation.

2.5.4.86 Right

The **Right** function returns the last character or characters in a string.

ABNF:

```
Right = val [val] sp ("1" / "2") ";RIGHT():129"
```

Required Arguments:

Name: **Text**

Type: vString

An argument that specifies the text string.

Optional Arguments:

Name: **NumChars**

Type: vSignedLong

An argument that specifies the number of characters to be returned. The default value is one.

Return Value:

Type: PtgStr1

This function returns a PtgStr1 containing the last **NumChars** characters of **Text**. If **NumChars** is less than zero or greater than the number of characters in **Text**, the function returns a PtgStr1 containing **Text**. If **NumChars** is equal to zero, the function returns a PtgStr1 containing an empty string.

2.5.4.87 Round

The **Round** function performs a rounding calculation.

ABNF:

```
Round = val val " ROUND():128"
```

Required Arguments:

Name: **Number**

Type: vDouble

An argument that specifies the number to round.

Name: **Digits**

Type: vSignedInt

An argument that specifies the decimal place to use for the rounding operation.

Return Value:

Type: vNum

This function returns a vNum containing the rounded value of **Number**, as described by the following table.

Condition	Return value
Digits > 0	The function returns **Number** rounded to **Digits** places to the right of the decimal point.
Digits = 0	The function returns **Number** rounded to an integer.
Digits < 0	The function returns **Number** rounded to negative **Digits** places to the left of the decimal point.

The unit of the return value is equal to the unit of **Number**. If **Digits** is less than -308 or greater than 15, the function returns **Number**.

2.5.4.88 Sat

The **Sat** function calculates the saturation component, as specified in the **HSL color space**, of a color value.

ABNF:

```
Sat = val " SAT():128"
```

Required Arguments:

Name: **Arg1**

Type: vColor

An argument that specifies the color.

Return Value:

Type: PtgUnsWord

This function returns a PtgUnsWord containing the saturation of **Arg1**. The value is less than or equal to 240.

2.5.4.89 SatDiff

The **SatDiff** function calculates the difference in saturation, as specified in the **HSL color space**, between two color values.

ABNF:

```
SatDiff = val val " SATDIFF():128"
```

Required Arguments:

Name: **Arg1**

Type: vColor

An argument that specifies the first color.

Name: **Arg2**

Type: vColor

An argument that specifies the second color.

Return Value:

Type: PtgNum

This function returns a PtgNum containing a value that is equal to the saturation of **Arg1** minus the saturation of **Arg2**. This value is greater than or equal to -240 and less than or equal to 240.

2.5.4.90 Second

The **Second** function returns the second component from a value representing a date and time.

ABNF:

```
Second = val [val] sp ("1" / "2") ";SECOND():129"
```

Required Arguments:

Name: **DateTimeArg**

Type: vAny

An argument that specifies a date and time value.

Optional Arguments:

Name: **Locale**

Type: vLanguageID

An argument that specifies an **LCID** to use when parsing **DateTimeArg**.

Return Value:

Type: PtgUnsWord, PtgErr

This function attempts a conversion, as described by the DateTime function, of **DateTimeArg** to a PtgDate. If the conversion succeeds, the function returns a PtgUnsWord containing the second component from **DateTimeArg.** The value is greater than or equal to zero and less than or equal to 59. If the conversion fails, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.91 SetAtRef

The **SetAtRef** function conditionally returns an argument.

ABNF:

```
SetAtRef = val [val [val]] sp ("1" / "2" / "3") ";SETATREF():129"
```

Required Arguments:

Name: **Arg1**

Type: vAny

An argument that specifies the operand to return.

Optional Arguments:

Name: **Unused**

Type: vAny

This argument is unused and MUST be ignored.

Name: **IgnoreArg**

Type: vAny

An argument that specifies whether **Arg1** is returned. The default value is FALSE.

Return Value:

Type: vAny

This function returns a vAny containing **Arg1** if **IgnoreArg** is FALSE. If **IgnoreArg** is TRUE, the function returns a PtgUnsWord containing a value of 0.

2.5.4.92 SetAtRefEval

The **SetAtRefEval** function returns an argument.

ABNF:

```
SetAtRefEval = val " SETATREFEVAL():128"
```

Required Arguments:

Name: **Arg1**

Type: vAny

An argument that specifies the operand to return.

Return Value:

Type: vAny

This function returns a vAny containing **Arg1**.

2.5.4.93 SetAtRefExpr

The **SetAtRefExpr** function returns an argument.

ABNF:

```
SetAtRefExpr = [val] sp ("0" / "1") ";SETATREFEXPR():129"
```

Optional Arguments:

Name: **Arg1**

Type: vAny

An argument that specifies the operand to return. The default value is a PtgUnsWord with a value of 0.

Return Value:

Type: vAny

This function returns a vAny containing **Arg1**.

2.5.4.94 Shade

The **Shade** function decreases the luminance, as specified in the **HSL color space**, of a color value.

ABNF:

```
Shade = val val " SHADE():128"
```

Required Arguments:

Name: **Color**

Type: vColor

An argument that specifies the color.

Name: **Delta**

Type: vSignedInt

An argument that specifies the amount to modify the luminance of **Color**.

Return Value:

Type: PtgColorRGB

This function returns a PtgColorRGB containing a color value with decreased luminance. If **Delta** is less than zero, the luminance is increased.

2.5.4.95 ShapeText

The **ShapeText** function is unsupported.

ABNF:

```
ShapeText = val [val] " SHAPETEXT():129"
```

Required Arguments:

Name: **Arg1**

Type: PtgMissArg

This argument is unused and MUST be ignored.

Optional Arguments:

Name: **Arg2**

Type: vUnsignedInt

This argument is unused and MUST be ignored.

Return Value:

Type: undefined

This function is not evaluated and does not return a valid result.

2.5.4.96 Sign

The **Sign** function returns the sign of a given number.

ABNF:

```
Sign = val [val] sp ("1" / "2") ";SIGN():129"
```

Required Arguments:

Name: **Number**

Type: vDouble

An argument that specifies the operand of the calculation.

Optional Arguments:

Name: **FuzzValue**

Type: vDouble

An argument that specifies the tolerance. If **FuzzValue** is negative, the absolute value of **FuzzValue** is used. The default value is 1E-9 (0.000000001).

Return Value:

Type: PtgNum

This function returns a PtgNum containing the sign of **Number**. If **Number** is greater than the absolute value of **FuzzValue**, the function returns 1. If **Number** is less than the negative of the absolute value of **FuzzValue**, the function returns -1. Otherwise, the function returns 0.

2.5.4.97 Sin

The **Sin** function performs the sine calculation.

ABNF:

```
Sin = val " SIN():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgNum

This function returns a PtgNum containing the sine of **Arg1**, where the value of **Arg1** is assumed to have the unit of radians.

2.5.4.98 SinH

The **SinH** function performs the hyperbolic sine calculation.

ABNF:

```
SinH = val " SINH():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgNum

This function returns a PtgNum containing the hyperbolic sine of **Arg1**, where the value of **Arg1** is assumed to have the unit of radians.

2.5.4.99 Sqrt

The **Sqrt** function performs a square root calculation.

ABNF:

```
Sqrt = val " SQRT():128"
```

Required Arguments:

Name: **Number**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: vNum, PtgErr

This function returns a vNum containing the square root of **Number**. The unit of the return value is equal to the unit of **Number**. If the dimension of **Number** is greater than 2, the function returns a PtgNumMultiDim with value equal to the square root of **Number**, unit equal to the unit of **Number**, and dimension equal to the dimension of **Number** divided by 2. If **Number** is less than 0, the function returns a PtgErr with an error code of #NUM!. If the dimension of **Number** is not divisible by two, the function returns a PtgErr with an error code of #DIM!.

2.5.4.100 StrSame

The **Strsame** function determines whether two text strings are the same.

ABNF:

```
StrSame = val val [val] sp ("2" / "3") ";STRSAME():129"
```

Required Arguments:

Name: **FirstText**

Type: vString

An argument that specifies the first string to be compared.

Name: **SecondText**

Type: vString

An argument that specifies the second string to be compared.

Optional Arguments:

Name: **IgnoreCase**

Type: vBoolean

An argument that specifies whether the comparison is case insensitive. A value of TRUE indicates that case is ignored. The default value is FALSE.

Return Value:

Type: PtgBool

This function returns a PtgBool containing TRUE if the strings are the same and FALSE otherwise. The comparison is performed according to .NET globalization **rules** based on the **LCID** with a value of 1033, which maps to US English. For more information about .NET globalization rules, see [MSDN-ENCLOC].

2.5.4.101 StrSameEx

The **StrSameEx** function determines whether two text strings are the same.

ABNF:

```
StrSameEx = val val val val " STRSAMEEX():128"
```

Required Arguments:

Name: **FirstText**

Type: vString

An argument that specifies the first string to be compared.

Name: **SecondText**

Type: vString

An argument that specifies the second string to be compared.

Name: **Locale**

Type: vLanguageID

An argument that specifies the **LCID** of the culture used by the string comparison, according to .Net globalization **rules**. A value of zero specifies the invariant culture. For more information about .NET globalization rules, see [MSDN-ENCLOC].

Name: **Flag**

Type: vSignedLong

An argument that specifies additional rules to be used in the comparison. The value is a combination of values from the following table. The rules given in the following table correspond to members of the .NET CompareOptions enumeration, as described in [MSDN-CompareOptions].

Value	Meaning
0x00	No additional rules apply.
0x01	Ignore case.
0x10	Ignore non-spacing combining characters.
0x100	Ignore symbols.
0x1000	All non-alphanumeric characters come before all alphanumeric characters.
0x10000	Ignore differences between hiragana and katakana characters that represent the same phonetic sound.
0x20000	Ignore character width, or differences between the single-byte and double-byte representations of the same character.

Return Value:

Type: PtgBool

This function returns a PtgBool containing TRUE if the strings are the same and FALSE otherwise. If the **Locale** or **Flag** arguments are invalid, this function returns a PtgBool containing a value of FALSE.

2.5.4.102 Sub

The **Sub** function performs a subtraction calculation.

ABNF:

```
Sub = val val " _SUB():128"
```

Required Arguments:

Name: **Arg1**

Type: vDoubleEx

An argument that specifies the first operand of the calculation.

Name: **Arg2**

Type: vDoubleEx

An argument that specifies the second operand of the calculation.

Return Value:

Type: vNum, PtgCy, PtgErr

This function returns a vNum or a PtgCy containing **Arg1** minus **Arg2**. The type of the return value is calculated by the following algorithm.

```
SET returnType = PtgNum
SET returnDimension = 0
SET returnCurrencyID = 0
IF Arg1.Type = PtgCy AND Arg2.Type != PtgCy THEN
        SET returnType = PtgCy
        SET returnCurrencyID = currencyID of Arg1
ELSE IF Arg1.Type != PtgCy AND Arg2.Type = PtgCy THEN
        SET returnType = PtgCy
        SET returnCurrencyID = currencyID of Arg2
ELSE IF Arg1.Type = PtgCy AND Arg2.Type = PtgCy THEN
        SET returnType = PtgCy
        IF currencyID of Arg1 = currencyID of Arg2 OR currencyID of Arg2 = 1 THEN
                SET returnCurrencyID = currencyID of Arg1
        ELSE IF currencyID of Arg1 = 1 THEN
                SET returnCurrencyID = currencyID of Arg2
        ELSE
                SET returnType = PtgErr
        END IF
ELSE IF Arg1.Type = PtgDate  AND Arg2.Type = PtgDate THEN
        SET returnType = PtgTDurDft
        SET returnDimension = 1
ELSE IF Arg1.Unit = PtgNumDft  AND Arg2.Unit is a vUnitType THEN
        SET returnType = Arg2.Unit
        SET returnDimension = 1
ELSE IF Arg1.Unit is a vUnitType THEN
        SET returnType = Arg1.Unit
        IF Arg2.Dimension = 0 OR Arg2.Dimension = Arg1.Dimension THEN
                SET returnDimension = Arg1.Dimension
        ELSE
                SET returnDimension = 1
        END IF
        IF (returnType = PtgAcre  OR returnType = PtgHectare) AND returnDimension != 2 THEN
                SET returnType = PtgNumDft
        END IF
ELSE IF Arg2.Unit is a vUnitType THEN
        SET returnType = Arg2.Unit
        IF Arg1.Dimension = 0 OR Arg2.Dimension = Arg1.Dimension THEN
                SET returnDimension = Arg2.Dimension
        ELSE
```

```
            SET returnDimension = 1
        END IF

        IF (returnType = PtgAcre OR returnType = PtgHectare) AND returnDimension != 2 THEN
            SET returnType = PtgNumDft
        END IF
END IF
IF returnType = PtgCy THEN
        Return type is PtgCy with currencyID = returnCurrencyID
ELSE IF returnType = PtgErr THEN
        Return type is PtgErr with error code of #VALUE!
ELSE IF returnDimension = 0 THEN
        Return type is PtgNum
ELSE IF returnDimension = 1 THEN
        Return type is returnType
ELSE
        Return type is PtgNumMultiDim with unit = returnType and dimension = returnDimension
END IF
```

2.5.4.103 Substitute

The **Substitute** function performs text substitution by finding a substring in a text string and replacing it with another text string.

ABNF:

```
Substitute = val val val [val [val]] sp ("3" / "4" /"5") ";SUBSTITUTE():129"
```

Required Arguments:

Name: **SourceText**

Type: vString

An argument that specifies the string to perform substitution on.

Name: **FindText**

Type: vString

An argument that specifies the substring in **SourceText** to be substituted.

Name: **SubstituteText**

Type: vString

An argument that specifies the string to use for substitution.

Optional Arguments:

Name: **WhichInstance**

Type: vSignedLong

An argument that specifies which instance, among multiple instances of **FindText** in **SourceText,** is to be substituted. The default value is 0.

Name: **IgnoreCase**

Type: vBoolean

An argument that specifies whether the search for **FindText** in **SourceText** is case insensitive. A value of TRUE specifies that case is ignored. The default value is FALSE.

Return Value:

Type: PtgStr1, PtgErr

This function returns a PtgStr1 containing **SourceText** modified with the substituted text. If **WhichInstance** is zero, the function returns a PtgErr with an error code of #VALUE!. If **WhichInstance** is less than zero or greater than the number of **FindText** instances in **SourceText**, or if **FindText** is not found in **SourceText**, the function returns a PtgStr1 containing **SourceText** without any substitution. The search is performed according to .NET globalization **rules** based on the **LCID** in the **DocumentLanguage** field of the Properties **GlobalElement** in the App XML Part of the document. For more information about .NET globalization rules, see [MSDN-ENCLOC].

2.5.4.104 Sum

The **Sum** function performs the sum calculation.

ABNF:

```
Sum = val *val sp unsigned-integer-value ";SUM():129"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the first operand of the calculation.

Optional Arguments:

Zero or more subsequent operands of type vDouble.

Return Value:

Type: vNum

This function returns a vNum containing the sum of all operands. If any of the arguments is a vUnitType, the unit of the return value is equal to the unit of the first argument that is a vUnitType. Otherwise, the function returns a PtgNum.

2.5.4.105 Tan

The **Tan** function performs the tangent calculation.

ABNF:

```
Tan = val " TAN():128"
```

Required Arguments:

Name: **Angle**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgNum

This function returns a PtgNum containing the tangent of the value of **Arg1**, where the value of **Arg1** is assumed to have the unit of radians.

2.5.4.106 TanH

The **TanH** function performs the hyperbolic tangent calculation.

ABNF:

```
TanH = val " TANH():128"
```

Required Arguments:

Name: **Angle**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: PtgNum

This function returns a PtgNum containing the hyperbolic tangent of the value of **Arg1**, where the value of **Arg1** is assumed to have the unit of radians.

2.5.4.107 TextHeight

The **TextHeight** function calculates the height of the text specified in a CT_ShapeText element in the ShapeTextBinding XML Part.

ABNF:

```
TextHeight = val val val val " TEXTHEIGHT():128"
```

Required Arguments:

Name: **Unused**

Type: vAny

This argument is unused and MUST be ignored.

Name: **Unused**

Type: vAny

This argument is unused and MUST be ignored.

Name: **Sheet**

Type: vString

An argument that specifies the **Name** of a CT_ShapeText element in the ShapeTextBinding XML Part.

Name: **OriginalHeight**

Type: vDouble

An argument that specifies the height of the text in **Sheet** prior to any diagram updates.

Return Value:

Type: PtgNum

This function returns a PtgNum containing the height of the text element.

2.5.4.108 TextWidth

The **TextWidth** function calculates the width of the text specified in a CT_ShapeText element in the ShapeTextBinding XML Part.

ABNF:

```
TextWidth = val val val val " TEXTWIDTH():128"
```

Required Arguments:

Name: **Unused**

Type: vAny

This argument is unused and MUST be ignored.

Name: **Unused**

Type: vAny

This argument is unused and MUST be ignored.

Name: **Sheet**

Type: vString

An argument that specifies the **Name** of a CT_ShapeText element in the ShapeTextBinding XML Part.

Name: **OriginalWidth**

Type: vDouble

An argument that specifies the width of the text in **Sheet** prior to any diagram updates.

Return Value:

Type: PtgNum

This function returns a PtgNum containing the width of the text element.

2.5.4.109 Time

The **Time** function returns a time from values representing an hour, minute, and second.

ABNF:

```
Time = val val val " TIME():128"
```

Required Arguments:

Name: **Hour**

Type: vUnsignedInt

An argument that specifies an offset in hours from midnight.

Name: **Minute**

Type: vUnsignedInt

An argument that specifies an offset in minutes from **Hour**.

Name: **Second**

Type: vUnsignedInt

An argument that specifies an offset in seconds from **Minute**.

Return Value:

Type: PtgDate

This function returns a PtgDate containing a time of day. The date-time-value component of the return value is the signed fractional value of the double precision number representing the input date-time, as specified in [MS-OAUT] section 2.2.25.

2.5.4.110 TimeValue

The **TimeValue** function returns the time component from a value representing a date and time.

ABNF:

```
TimeValue = val [val] sp ("1" / "2") ";TIMEVALUE():129"
```

Required Arguments:

Name: **DateTimeArg**

Type: vAny

An argument that specifies a value representing a date and time.

Optional Arguments:

Name: **Locale**

Type: vLanguageID

An argument that specifies an **LCID** to use when parsing **DateTimeArg**.

Return Value:

Type: PtgDate, PtgErr

This function attempts a conversion, as described by the DateTime function, of **DateTimeArg** to a PtgDate. If the conversion succeeds, the function returns a PtgDate containing the time component of **DateTimeArg**. If the conversion fails, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.111 Tint

The **Tint** function increases the luminance, as specified in the **HSL color space**, of a color value.

ABNF:

```
Tint = val val " TINT():128"
```

Required Arguments:

Name: **Color**

Type: vColor

An argument that specifies the color.

Name: **Delta**

Type: vSignedInt

An argument that specifies the amount to modify the luminance of **Color**.

Return Value:

Type: PtgColorRGB

This function returns a PtgColorRGB containing a color value with increased luminance. If **Delta** is less than zero, the luminance is decreased.

2.5.4.112 Tone

The **Tone** function decreases the saturation, as specified in the **HSL color space**, of a color value.

ABNF:

```
Tone = val val " TONE():128"
```

Required Arguments:

Name: **Color**

Type: vColor

An argument that specifies the color.

Name: **Delta**

Type: vSignedInt

An argument that specifies the amount to modify the saturation of **Color**.

Return Value:

Type: PtgColorRGB

This function returns a PtgColorRGB containing a color value with decreased saturation. If **Delta** is less than zero, the saturation is increased.

2.5.4.113 Trim

The **Trim** function performs the removal of all **whitespace** from a string except for single spaces between words.

ABNF:

```
Trim = val " TRIM():128"
```

Required Arguments:

Name: **Arg1**

Type: vString

An argument that specifies the string from which to remove whitespace.

Return Value:

Type: PtgStr1

This function returns a PtgStr1 containing the string with whitespace removed. The following **Unicode** characters are considered as whitespace.

Value	Meaning
"\u0009"	Character Tabulation
"\u000a"	Line Feed
"\u000b"	Line Tabulation
"\u000c"	Form Feed
"\u000d"	Carriage Return
"\u0020"	Space
"\u00a0"	No-Break Space
"\u1680"	Ogham Space Mark
"\u180e"	Mongolian Vowel Separator
"\u2000"	En Quad
"\u2001"	Em Quad
"\u2002"	En Space
"\u2003"	Em Space
"\u2004"	Three-Per-Em Space
"\u2005"	Four-Per-Em Space
"\u2006"	Six-Per-Em Space

Value	Meaning
"\u2007"	Figure Space
"\u2008"	Punctuation Space
"\u2009"	Thin Space
"\u200a"	Hair Space
"\u202f"	Narrow No-Break Space
"\u205f"	Medium Mathematical Space
"\u3000"	Ideographic Space

2.5.4.114 Trunc

The **Trunc** function performs a truncation operation.

ABNF:

```
Trunc = val val " TRUNC():128"
```

Required Arguments:

Name: **Number**

Type: vDouble

An argument that specifies the number to truncate.

Name: **Digits**

Type: vSignedInt

An argument that specifies the decimal place to use for the truncation operation.

Return Value:

Type: vNum

This function returns a vNum containing the truncated value of **Number**, as described by the following table.

Condition	Result
Digits > 0	The function returns **Number** truncated to **Digits** places to the right of the decimal point.
Digits = 0	The function returns **Number** truncated to an integer.
Digits < 0	The function returns **Number** truncated to negative **Digits** places to the left of the decimal point.

The unit of the return value is equal to the unit of **Number**. If **Digits** is less than -308 or greater than 15, the function returns **Number**.

2.5.4.115 UMinus

The **UMinus** function performs the negation calculation.

ABNF:

```
UMinus = val " _UMINUS():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: vNum

This function returns a vNum containing the negation of the value of **Arg1**. The unit of the return value is equal to the unit of **Arg1**.

2.5.4.116 UPlus

The **UPlus** function performs the identity calculation.

ABNF:

```
UPlus = val " _UPLUS():128"
```

Required Arguments:

Name: **Arg1**

Type: vDouble

An argument that specifies the operand of the calculation.

Return Value:

Type: vNum

This function returns a vNum containing the value of **Arg1**. The unit of the return value is equal to the unit of **Arg1**.

2.5.4.117 Upper

The **Upper** function performs upper case conversion.

ABNF:

```
Upper = val " UPPER():128"
```

Required Arguments:

Name: **Arg1**

Type: vString

An argument that specifies the string to convert.

Return Value:

Type: PtgStr1

This function returns a PtgStr1 containing the string converted to upper case. The conversion is performed according to .NET globalization **rules** based on the **LCID** in the **DocumentLanguage** field of the Properties **GlobalElement** in the App XML Part of the document. For more information about .NET globalization rules, see [MSDN-ENCLOC].

2.5.4.118 WeekDay

The **WeekDay** function returns the day of the week, according to the Gregorian calendar, from a value representing a date and time.

ABNF:

```
WeekDay = val [val] sp ("1" / "2") ";WEEKDAY():129"
```

Required Arguments:

Name: **DateTimeArg**

Type: vAny

An argument that specifies a date and time value.

Optional Arguments:

Name: **Locale**

Type: vLanguageID

An argument that specifies an **LCID** to use when parsing **DateTimeArg**.

Return Value:

Type: PtgUnsWord, PtgErr

This function attempts a conversion, as described by the DateTime function, of **DateTimeArg** to a PtgDate. If the conversion succeeds, the function returns a PtgUnsWord containing the weekday component from **DateTimeArg.** The value is greater than or equal to 1 and less than or equal to 7, where a value of 1 corresponds to Monday. If the conversion fails, the function returns a PtgErr with an error code of #VALUE!.

2.5.4.119 Year

The **Year** function returns the year, according to the Gregorian calendar, from a value representing a date and time.

ABNF:

```
Year = val [val] sp ("1" / "2") ";YEAR():129"
```

Required Arguments:

Name: **DateTimeArg**

Type: vAny

An argument that specifies a date and time value.

Optional Arguments:

Name: **Locale**

Type: vLanguageID

An argument that specifies an **LCID** to use when parsing **DateTimeArg**.

Return Value:

Type: PtgUnsWord, PtgErr

This function attempts a conversion, as described by the DateTime function, of **DateTimeArg** to a PtgDate. If the conversion succeeds, the function returns a PtgUnsWord containing the year component from **DateTimeArg**. The value is greater than or equal to 1 and less than or equal to 9999. If the conversion fails, the function returns a PtgErr with an error code of #VALUE!.

2.5.5 Parse Token Table

This section specifies the parse tokens (Ptgs) that can be part of a formula expression. The numerical token type ID and token type name are provided for each token.

ID	Ptg
36	PtgAcre
81	PtgAngDD
80	PtgAngDft
82	PtgAngDMS
83	PtgAngRad
97	PtgBool
106	PtgColorRGB
111	PtgCy
1	PtgDataBinding
40	PtgDate
44	PtgEDay
45	PtgEHour
46	PtgEMin
224	PtgErr
47	PtgESec
43	PtgEWeek
128	PtgFunc

ID	Ptg
129	PtgFuncVar
37	PtgHectare
163	PtgJmp
160	PtgJmpF
164	PtgJmpLabel
161	PtgJmpT
229	PtgMissArg
240	PtgNoOp
32	PtgNum
69	PtgNumCM
64	PtgNumDft
66	PtgNumF
67	PtgNumFI
65	PtgNumI
72	PtgNumKM
71	PtgNumM
68	PtgNumMI
70	PtgNumMM
41	PtgNumMultiDim
76	PtgNumNM
33	PtgNumPct
75	PtgNumYards
63	PtgPageDft
225	PtgPnt
147	PtgPop
149	PtgPushTop
2	PtgRecalcRef
96	PtgStr1
42	PtgTDurDft
52	PtgTypCD
54	PtgTypCi
48	PtgTypDft

ID	Ptg
53	PtgTypDi
51	PtgTypPi
49	PtgTypPP
50	PtgTypPt
98	PtgUnsWord

2.5.6 Parse Token Definitions

This section specifies the tokens that can be part of a formula expression. The definition of each token specifies the **TokenValue** and **TokenType** as described in the parse tokens section. In the ABNF string specified for each token, the **TokenValue** is the left side of the token-separator and the **TokenType** is the right side of the `token-separator`.

2.5.6.1 PtgAcre

The **PtgAcre** structure is a PtgNumMultiDim that specifies a Unit Number with a unit of acres.

ABNF:

```
PtgAcre = double-value ",36,2" token-separator "41"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.2 PtgAngDD

The **PtgAngDD** structure specifies a Unit Number with a unit of decimal degrees.

ABNF:

```
PtgAngDD = double-value token-separator "81"
```

`double-value` is an angleInternalUnitNumber.

2.5.6.3 PtgAngDft

The **PtgAngDft** structure specifies a Unit Number. The unit is specified by the **DefaultUnitsAngle** attribute of CT_Page.

ABNF:

```
PtgAngDft = double-value token-separator "80"
```

`double-value` is an angleInternalUnitNumber.

2.5.6.4 PtgAngDMS

The **PtgAngDMS** structure specifies a Unit Number with a unit of degrees-minutes-seconds.

ABNF:

```
PtgAngDMS = double-value token-separator "82"
```

`double-value` is an angleInternalUnitNumber.

2.5.6.5 PtgAngRad

The **PtgAngRad** structure specifies a Unit Number with a unit of radians.

ABNF:

```
PtgAngRad = double-value token-separator "83"
```

`double-value` is an angleInternalUnitNumber.

2.5.6.6 PtgBool

The **PtgBool** structure specifies a **Boolean** value.

ABNF:

```
PtgBool = bool-value token-separator "97"
```

`bool-value` specifies a Boolean. It MUST be a case insensitive value from the following table:

Value	Meaning
false	Value is FALSE
true	Value is TRUE

2.5.6.7 PtgColorRGB

The **PtgColorRGB** structure specifies an **RGB** color value represented as a 4-byte signed integer.

ABNF:

```
PtgColorRGB = color-value token-separator "106"
```

The most significant byte of `color-value` MUST be equal to 0xFF. The second most significant byte of `color-value` specifies the intensity of red. The third most significant byte of `color-value` specifies the intensity of green. The least significant byte of `color-value` specifies the intensity of blue.

2.5.6.8 PtgCy

The **PtgCy** structure specifies a currency value.

ABNF:

```
PtgCy = double-value "," currency "," string-value token-separator "111"
```

`double-value` is a double precision **floating-point number**.

`currency` is an unsigned integer that specifies the currency. It MUST be a value defined by vCurrencyID.

`string-value` specifies the formatting information. It MUST be a numeric format string, as described in [MSDN-FormattingTypes].

2.5.6.9 PtgDataBinding

The **PtgDataBinding** structure specifies a reference to a shape data item whose value is bound to external data, as described in Data Binding.

ABNF:

```
PtgDataBinding = shape-name "," shape-data "," shape-data-type "," dimension token-separator
"1"
```

`shape-name` specifies the **Name** attribute of the CT_ShapeInfo element in the ShapeInfo XML Part of this document that contains the shape data item.

`shape-data` specifies the **Name** attribute of a CT_ShapeData element in the ShapeInfo XML Part of this document. This element is the referenced shape data item and is a subordinate element of the element referenced by `shape-name`.

`shape-data-type` specifies the measurement unit of the value referenced by `shape-data`. It MUST be an entry under the ID column in the Parse Token Table in the Parse Token Table section.

`dimension` specifies the dimension of the value referenced by `shape-data`.

2.5.6.10 PtgDate

The **PtgDate** structure specifies a Unit Number with a unit of time.

ABNF:

```
PtgDate = date-time-value token-separator "40"
```

`date-time-value` specifies a date and time, as specified in [MS-OAUT] section 2.2.25.

2.5.6.11 PtgEDay

The **PtgEDay** structure specifies a Unit Number with a unit of days.

ABNF:

```
PtgEDay = double-value token-separator "44"
```

`double-value` is a durationInternalUnitNumber.

2.5.6.12 PtgEHour

The **PtgEHour** structure specifies a Unit Number with a unit of hours.

ABNF:

```
PtgEHour = double-value token-separator "45"
```

`double-value` is a durationInternalUnitNumber.

2.5.6.13 PtgEMin

The **PtgEMin** structure specifies a Unit Number with a unit of minutes.

ABNF:

```
PtgEMin = double-value token-separator "46"
```

`double-value` is a durationInternalUnitNumber.

2.5.6.14 PtgErr

Th**PtgErr** structure specifies an error code.

ABNF:

```
PtgErr = string-value token-separator "224"
```

`string-value` specifies a string and MUST be a value from the following table.

Value	Meaning
#DIM!	A dimensional value that exceeds the dimension range
#DIV/0	Division by 0
#VALUE!	An argument or operand of the wrong type
#REF!	A reference to a cell that does not exist
#NUM!	An invalid number
#N/A	Not available value

2.5.6.15 PtgESec

The **PtgESec** structure specifies a Unit Number with a unit of seconds.

ABNF:

```
PtgESec = double-value token-separator "47"
```

`double-value` is a durationInternalUnitNumber.

2.5.6.16 PtgEWeek

The **PtgEWeek** structure specifies a Unit Number with a unit of weeks.

ABNF:

```
PtgEWeek = double-value token-separator "43"
```

`double-value` is a durationInternalUnitNumber.

2.5.6.17 PtgFunc

The **PtgFunc** structure specifies a function with a fixed number of arguments.

ABNF:

```
PtgFunc = fixed-func-name "()" token-separator "128"
```

`fixed-func-name` specifies a function name which MUST be an entry under the ABNF func-name column in the Function Token Table as specified in the Function Token Table section. The corresponding entry under the Type column for that function MUST be PtgFunc.

2.5.6.18 PtgFuncVar

The **PtgFuncVar** structure specifies a function with a variable number of arguments.

ABNF:

```
PtgFuncVar = num-args ";" var-func-name "()" token-separator "129"
```

`num-args` specifies the number of arguments.

`var-func-name` specifies a function name which MUST be an entry under the ABNF func-name column in the Function Token Table as specified in the Function Token Table section. The corresponding entry under the Type column for that function MUST be PtgFuncVar.

2.5.6.19 PtgHectare

The **Ptg** structure is a PtgNumMultiDim that specifies a Unit Number with a unit of hectares.

ABNF:

```
PtgHectare = double-value ",37,2" token-separator "41"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.20 PtgJmp

The **PtgJmp** structure is a control token that specifies the position of the next token to be evaluated in the formula expression.

ABNF:

```
PtgJmp = string-value token-separator "163"
```

`string-value` is the label that specifies the position.

2.5.6.21 PtgJmpF

The **PtgJmpF** structure is a control token that specifies the position of the next token to be evaluated in the formula expression if the token that is currently at the top of the evaluation stack evaluates to a value of FALSE. If the token that is currently at the top of the evaluation stack evaluates to a value of TRUE, it is popped from the evaluation stack.

ABNF:

```
PtgJmpF = string-value token-separator "160"
```

`string-value` is the label that specifies the position.

2.5.6.22 PtgJmpLabel

The **PtgJmpLabel** structure is a control token that specifies a position in the formula expression.

ABNF:

```
PtgJmpLabel = string-value token-separator "164"
```

`string-value` is the unique label of the current position.

2.5.6.23 PtgJmpT

The **PtgJmpT** structure is a control token that specifies the position of the next token to be evaluated in the formula expression if the token that is currently at the top of the evaluation stack evaluates to a value of TRUE. If the token that is currently at the top of the evaluation stack evaluates to a value of FALSE, it is popped from the evaluation stack.

ABNF:

```
PtgJmpT = string-value token-separator "161"
```

`string-value` is the label that specifies the position.

2.5.6.24 PtgMissArg

The **PtgMissArg** structure specifies a function argument that does not have a value.

ABNF:

```
PtgMissArg = empty-value token-separator "229"
```

2.5.6.25 PtgNoOp

The **PtgNoOp** structure is a token that specifies the operand is not added to the evaluation stack.

ABNF:

```
PtgNoOp = [double-value] token-separator "240"
```

The `double-value`, if present, is unused and MUST be ignored.

2.5.6.26 PtgNum

The **PtgNum** structure specifies a double precision **floating-point number**.

ABNF:

```
PtgNum = double-value token-separator "32"
```

`double-value` is a double precision floating-point number.

2.5.6.27 PtgNumCM

The **PtgNumCM** structure specifies a Unit Number with a unit of centimeters.

ABNF:

```
PtgNumCM = double-value token-separator "69"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.28 PtgNumDft

The **PtgNumDft** structure specifies a Unit Number. If the **DocumentMeasurementSystem** attribute of CT_Properties in the App XML Part has a value of "Metric," the unit is millimeters. If **DocumentMeasurementSystem** has a value of "US Units," the unit is inches.

ABNF:

```
PtgNumDft = double-value token-separator "64"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.29 PtgNumF

The **PtgNumF** structure specifies a Unit Number with a unit of feet.

ABNF:

```
PtgNumF = double-value token-separator "66"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.30 PtgNumFI

The **PtgNumFI** structure specifies a Unit Number with a unit of feet and inches.

ABNF:

```
PtgNumFI = double-value token-separator "67"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.31 PtgNumI

The **PtgNumI** structure specifies a Unit Number with a unit of inches.

ABNF:

```
PtgNumI = double-value token-separator "65"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.32 PtgNumKM

The **PtgNumKM** structure specifies a Unit Number with a unit of kilometers.

ABNF:

```
PtgNumKM = double-value token-separator "72"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.33 PtgNumM

The **PtgNumM** structure specifies a Unit Number with a unit of meters.

ABNF:

```
PtgNumM = double-value token-separator "71"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.34 PtgNumMI

The **PtgNumMI** structure specifies a Unit Number with a unit of miles.

ABNF:

```
PtgNumMI = double-value token-separator "68"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.35 PtgNumMM

The **PtgNumMM** structure specifies a Unit Number with a unit of millimeters.

ABNF:

```
PtgNumMM = double-value token-separator "70"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.36 PtgNumMultiDim

The **PtgNumMultiDim** structure specifies a multidimensional Unit Number.

ABNF:

```
PtgNumMultiDim = double-value "," unit "," dimension token-separator "41"
```

`double-value` is a lengthInternalUnitNumber.

`unit` is an integer that specifies the unit of measurement. It MUST be equal to an entry under the ID column in the Parse Token Table as specified in the Parse Token Table section. The corresponding entry under the Ptg column for that token MUST be a vUnitType.

`dimension` is an integer that specifies the number of dimensions.

2.5.6.37 PtgNumNM

The **PtgNumNM** structure specifies a Unit Number with a unit of nautical miles.

ABNF:

```
PtgNumNM = double-value token-separator "76"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.38 PtgNumPct

The **PtgNumPct** structure specifies a Unit Number expressed as a percentage. The value is normalized such that the value of 1 corresponds to 100 percent.

ABNF:

```
PtgNumPct = double-value token-separator "33"
```

`double-value` is a double precision **floating-point number**.

2.5.6.39 PtgNumYards

The **PtgNumYards** structure specifies a Unit Number with a unit of yards.

ABNF:

```
PtgNumYards = double-value token-separator "75"
```

`double-value` is a lengthInternalUnitNumber.

2.5.6.40 PtgPageDft

The **PtgPageDft** structure specifies a Unit Number.

ABNF:

```
PtgPageDft = double-value token-separator "63"
```

The type of `double-value` is determined by the unit specified in the **DefaultUnitsPage** attribute of CT_Page as follows:

Value	Meaning
DefaultUnitsPage is greater than or equal to 43 and less than or equal to 47	`double-value` is a durationInternalUnitNumber
DefaultUnitsPage is greater than or equal to 50 and less than or equal to 54	`double-value` is a typographicInternalUnitNumber
DefaultUnitsPage is greater than or equal to 63 and less than or equal to 76	`double-value` is a lengthInternalUnitNumber

2.5.6.41 PtgPnt

The **PtgPnt** token is not supported and MUST NOT be used.

ABNF:

```
PtgPnt = string-value token-separator "225"
```

2.5.6.42 PtgPop

The **PtgPop** token specifies the removal of the top element of the evaluation stack.

ABNF:

```
PtgPop = [string-value] token-separator "147"
```

The `string-value`, if present, is unused and MUST be ignored.

2.5.6.43 PtgPushTop

The **PtgPushTop** token specifies the duplication of the top element of the evaluation stack and the insertion of this duplicate on to the top of the evaluation stack.

ABNF:

```
PtgPushTop = [string-value] token-separator "149"
```

The `string-value`, if present, is unused and MUST be ignored.

2.5.6.44 PtgRecalcRef

The **PtgRecalcRef** structure specifies a reference to a formula expression.

ABNF:

```
PtgRecalcRef = string-value token-separator "2"
```

`string-value` specifies the name of the referenced formula expression and MUST match the name of one of the CT_FormulaReferences elements in the Data Graphics XML Part of this document.

2.5.6.45 PtgStr1

The **PtgStr1** structure specifies a string.

ABNF:

```
PtgStr1 = string-value token-separator "96"
```

`string-value` is a string. String length MUST be less than or equal to 65535.

2.5.6.46 PtgTDurDft

The **PtgTDurDft** structure specifies a Unit Number. The unit is specified by the **DefaultUnitsDuration** attribute of CT_Page.

ABNF:

```
PtgTDurDft = double-value token-separator "42"
```

`double-value` specifies a durationInternalUnitNumber.

2.5.6.47 PtgTypCD

The **PtgTypCD** structure specifies a Unit Number with a unit of ciceros and didots.

ABNF:

```
PtgTypCD = double-value token-separator "52"
```

`double-value` specifies a typographicInternalUnitNumber.

2.5.6.48 PtgTypCi

The **PtgTypCi** structure specifies a Unit Number with a unit of ciceros.

ABNF:

```
PtgTypCi = double-value token-separator "54"
```

`double-value` specifies a typographicInternalUnitNumber.

2.5.6.49 PtgTypDft

The **PtgTypDft** structure specifies a Unit Number. The unit is specified by the **DefaultUnitsText** attribute of CT_Page.

ABNF:

```
PtgTypDft = double-value token-separator "48"
```

`double-value` is a typographicInternalUnitNumber.

2.5.6.50 PtgTypDi

The **PtgTypDi** structure specifies a Unit Number with a unit of didots.

ABNF:

```
PtgTypDi = double-value token-separator "53"
```

`double-value` specifies a typographicInternalUnitNumber.

2.5.6.51 PtgTypPi

The **PtgTypPi** structure specifies a Unit Number with a unit of picas.

ABNF:

```
PtgTypPi = double-value token-separator "51"
```

`double-value` is a typographicInternalUnitNumber.

2.5.6.52 PtgTypPP

The **PtgTypPP** structure specifies a Unit Number with a unit of pica points.

ABNF:

```
PtgTypPP = double-value token-separator "49"
```

`double-value` is a typographicInternalUnitNumber.

2.5.6.53 PtgTypPt

The **PtgTypPt** structure specifies a Unit Number with a unit of points.

ABNF:

```
PtgTypPt = double-value token-separator "50"
```

`double-value` is a typographicInternalUnitNumber.

2.5.6.54 PtgUnsWord

The **PtgUnsWord** structure specifies a 2-byte unsigned integer.

ABNF:

```
PtgUnsWord = unsigned-integer-value token-separator "98"
```

`unsigned-integer-value` specifies an integer and MUST be greater than or equal to 0 and less than or equal to 65535.

2.5.7 Custom Input Type Definitions

This section specifies custom input types. Custom input types are used to specify the token types for function arguments. Functions that require a particular class of token inputs, and thus require a conversion from many possible source token types, declare arguments with custom input types and rely on the custom input type to derive a valid input argument from a source token. The definition of each custom input type specifies how the source token is derived, the computed properties of the custom input type and under which cases the input argument is invalid.

2.5.7.1 vBoolean

The **vBoolean** custom input type specifies a PtgBool that is derived from a source token that MUST be a vNumAny, PtgStr1, PtgCy, or PtgMissArg. It contains the following computed properties.

Value

This property specifies a **Boolean** derived from the source token as follows.

If the source token is a PtgBool or PtgStr1 with a value of TRUE (case insensitive), **Value** is equal to TRUE. If the source token is a PtgBool or PtgStr1 with a value of FALSE (case insensitive), **Value** is equal to FALSE.

If the source token is a vNum, PtgUnsWord or PtgCy with a value of zero, **Value** is equal to FALSE. If the source token is a vNum, PtgUnsWord or PtgCy with a value not equal to zero, **Value** is equal to TRUE.

If the source token is a PtgMissArg, **Value** is equal to FALSE.

In all other cases, the input argument is not valid.

Type

This property specifies a token type of PtgBool.

2.5.7.2 vColor

The **vColor** custom input type specifies a PtgColorRGB derived from a source token that MUST be a PtgColorRGB, vNumAny, PtgStr1, PtgCy, or PtgMissArg. It contains the following computed properties.

Value

This property specifies an **RGB** color value represented as a 4-byte signed integer derived from the source token value as follows.

If the source token type is a PtgColorRGB, **Value** is equal to the source token value.

If the source token can be interpreted as a vSignedLong, **Value** is equal to the color in the color table that is indexed by the value of the source token interpreted as a vSignedLong; if the value of the source token interpreted as a vSignedLong is not a valid index in the color table, **Value** is equal to the color at index zero in the color table.

In all other cases, the input argument is not valid.

Type

This property specifies a token type of PtgColorRGB.

2.5.7.3 vDouble

The **vDouble** custom input type specifies a vNum derived from a source token that MUST be a vNumAny, PtgStr1, PtgCy, or PtgMissArg. It contains the following computed properties.

Value

This property specifies a double precision **floating-point number** derived from the source token value as follows.

If the source token value can be converted to a double, as described in [MSDN-ToDouble], **Value** is the result of the conversion.

If the source token value is equal to TRUE (case insensitive), **Value** is equal to 1. If the source token value is equal to FALSE (case insensitive), **Value** is equal to 0.

If the source token is a PtgMissArg, **Value** is equal to 0.

In all other cases, the input argument is not valid.

Type

This property specifies a token type. If the source token is a vUnitType, **Type** is equal to the token type of the source token. Otherwise, **Type** is equal to the token type of PtgNum.

Dimension

This property specifies the dimension of the value. If the source token type is a PtgNumMultiDim, **Dimension** is equal to the dimension of the source token. If the source token is any other vUnitType, **Dimension** is equal to one. Otherwise, **Dimension** is equal to zero.

Unit

This property specifies the measurement unit of the value. If the source token is a PtgNumMultiDim, **Unit** is equal to the unit of the source token. Otherwise, **Unit** is equal to **Type**.

2.5.7.4 vDoubleEx

The **vDoubleEx** custom input type specifies either a vNum or PtgCy derived from a source token that MUST be a vNumAny, PtgStr1, PtgCy, or PtgMissArg. It is the vDouble custom input type with the following exceptions and additional computed properties.

Type

This property specifies a token type. If the source token is a PtgCy, **Type** is equal to the token type of PtgCy. Otherwise, **Type** is equal to the type of the source token interpreted as a vDouble.

Currency

This property specifies the currency ID. If the source token is a PtgCy, **Currency** is equal to the currency of the source token. Otherwise, **Currency** is equal to 0 which specifies the unknown currency type, as described in vCurrencyID.

2.5.7.5 vFloat

The **vFloat** custom input type specifies a single precision **floating-point number** derived from a source token that MUST be a vNumAny, PtgStr1, PtgCy, or PtgMissArg. It is a vDouble with the additional restriction that the **Value** property MUST conform to the range of values specified for a single precision floating-point number, as defined in [IEEE754].

If the source token is not a vDouble or the value of the source token interpreted as a vDouble does not conform to the range of values specified for a single precision floating-point number, the input argument is not valid.

2.5.7.6 vSignedInt

The **vSignedInt** custom input type specifies a signed integer derived from a source token that MUST be a vNumAny, PtgStr1, PtgCy, or PtgMissArg. It is a vSignedLong with the additional restriction that the **Value** property MUST be greater than or equal to -32768 and less than or equal to 32767.

If the source token is not a vSignedLong or the value of the source token interpreted as a vSignedLong is less than -32768 or greater than 32767, the input argument is not valid.

2.5.7.7 vSignedLong

The **vSignedLong** custom input type specifies a signed long integer derived from a source token that MUST be a vNumAny, PtgStr1, PtgCy, or a PtgMissArg. It contains the following computed properties.

Value

This property specifies a signed long integer derived from the source token value as follows.

If the source token value can be converted to a double, as described in [MSDN-ToDouble], **Value** is equal to a signed long integer calculated as follows.

> Use the conversion method described in [MSDN-ToDouble] to obtain a double value. If the double value is not an integer, round the value towards zero to the next integer. If the integer is less than -2147483648 or greater than 2147483647, the input argument is not valid.

If the source token value is equal to TRUE (case insensitive), **Value** is equal to 1. If the source token value is equal to FALSE (case insensitive), **Value** is equal to 0.

If the source token is a PtgMissArg, **Value** is equal to 0.

In all other cases, the input argument is not valid.

Type

This property specifies a token type. If the source token is a vNumAny, **Type** is equal to the token type of the source token. Otherwise, **Type** is equal to the token type of PtgNum.

2.5.7.8 vString

The **vString** custom input type specifies a PtgStr1 derived from a source token that MUST be a vNumAny, PtgStr1, or PtgMissArg. It contains the following computed properties.

Value

This property specifies a string derived from the source token value as follows.

If the source token is a PtgStr1, **Value** is equal to the source token value.

If the source token is a vNumAny, **Value** is the string form of the source token value.

If the source token is a PtgMissArg, **Value** is equal to the empty string.

In all other cases, the input argument is not valid.

Type

This property specifies a token type of PtgStr1.

2.5.7.9 vUnsignedInt

The **vUnsignedInt** custom input type specifies an unsigned integer derived from a source token that MUST be a vNumAny, PtgStr1, PtgCy, or a PtgMissArg. It is a vUnsignedLong with the additional restriction that **Value** MUST be less than 65536. **Value** is equal to the result of the value of the source token interpreted as a vSignedLong modulo 65536.

2.5.7.10 vUnsignedLong

The **vUnsignedLong** custom input type specifies an unsigned long integer derived from a source token that MUST be a vNumAny, PtgStr1, PtgCy, or a PtgMissArg. It contains the following computed properties.

Value

This property specifies an unsigned long integer derived from the source token value as follows.

If the source token value can be converted to a double, as described in [MSDN-ToDouble], **Value** is equal to an unsigned long integer calculated as follows.

> Use the conversion method described in [MSDN-ToDouble] to obtain a double value. If the double value is not an integer, round the value towards zero to the next integer. If the integer is greater than or equal to zero, **Value** is equal to the integer modulo 4294967296. Otherwise **Value** is equal to the sum of the integer modulo 4294967296 and 4294967296.

If the source token value is equal to TRUE (case insensitive), **Value** is equal to 1. If the source token value is equal to FALSE (case insensitive), **Value** is equal to 0.

If the source token is a PtgMissArg, **Value** is equal to 0.

In all other cases, the input argument is not valid.

Type

This property specifies a token type. If the source token is a vUnitType, **Type** is equal to the token type of the source token. Otherwise, **Type** is equal to the token type of PtgNum.

2.5.8 Custom Token Groupings

This section defines groupings of tokens, referred to as custom token groupings. A custom token grouping specifies a set of tokens that represents a specific concept. Custom token groupings can contain other custom token groupings. The association of a token to a custom token grouping is not exclusive. These grouping do not exist in the format of the file. The purpose of these groupings is to improve the readability of the document by allowing tokens representing similar concepts to be referred to collectively.

2.5.8.1 vAngle

The **vAngle** custom token grouping is an aggregation of types that represent an angle. The vAngle custom token grouping contains the following tokens.

PtgAngDD, PtgAngDft, PtgAngDMS, PtgAngRad

2.5.8.2 vAny

The **vAny** custom token grouping is an aggregation of types that represent data. The vAny custom token grouping contains the following tokens.

vNumAny, PtgStr1, PtgColorRGB, PtgCy, PtgErr

2.5.8.3 vNum

The **vNum** custom token grouping is an aggregation of types that represent either a Unit Number or a double precision **floating-point number**. The vNum custom token grouping contains the following tokens.

vUnitType, PtgNum

2.5.8.4 vNumAny

The **vNumAny** custom token grouping is an aggregation of types that represent any number. The vNumAny custom token grouping contains the following tokens.

vNum, PtgBool, PtgUnsWord

2.5.8.5 vUnitType

The **vUnitType** custom token grouping is an aggregation of types that represent a Unit Number. The vUnitType custom token grouping contains the following tokens.

vAngle, PtgAcre, PtgDate, PtgEDay, PtgEHour, PtgEMin, PtgESec, PtgEWeek, PtgHectare, PtgNumCM, PtgNumDft, PtgNumF, PtgNumFI, PtgNumI, PtgNumKM, PtgNumM, PtgNumMI, PtgNumMM, PtgNumMultiDim, PtgNumNM, PtgNumPct, PtgNumYards, PtgPageDft, PtgTDurDft, PtgTypCD, PtgTypCi, PtgTypDft, PtgTypDi, PtgTypPi, PtgTypPP, PtgTypPt

2.5.9 Custom Internal Unit Types

This section specifies internal unit types. All internal unit values MUST conform to a double precision **floating-point number**.

2.5.9.1 angleInternalUnitNumber

The **angleInternalUnitNumber** custom internal unit type specifies a value with a unit of radians.

2.5.9.2 durationInternalUnitNumber

The **durationInternalUnitNumber** custom internal unit type specifies a value with a unit of days.

2.5.9.3 lengthInternalUnitNumber

The **lengthInternalUnitNumber** custom internal unit type specifies a value with a unit of inches.

2.5.9.4 typographicInternalUnitNumber

The **typographicInternalUnitNumber** custom internal unit type specifies a value with a unit of inches.

2.5.10 Custom Structures

This section specifies custom structures. A custom structure specifies a type and a set of valid values.

2.5.10.1 vCalendar

The **vCalendar** custom structure is an unsigned integer that specifies the calendar system to use when formatting dates and times.

It MUST be a value from the following table.

Value	Description
0	Western
1	Arabic Hijri
2	Hebrew Lunar
3	Chinese National
4	Japanese Emperor
5	Thai Buddhist
6	Korean Danki
7	Japanese Saka Era
8	Transliterated English
9	Transliterated French
10	Gregorian US English
11	Gregorian Middle East French
12	Gregorian Arabic
13	Um-al-Qura

2.5.10.2 vCurrencyID

The **vCurrencyID** custom structure is an unsigned integer that specifies a currency.

It MUST be a value from the following table.

Value	Meaning	Value	Meaning
0	Unknown currency type	1	Omit currency symbol
10	European Union: euro	11	United States: dollar
12	Austria: schilling	13	Australia: dollar
14	Belgium: franc	15	Canada: dollar
16	Switzerland: franc	17	China (Mainland): yuan
18	Germany: mark	19	Denmark: krone
20	Spain: peseta	21	Finland: markka
22	France: franc	23	United Kingdom: pound
24	Greece: drachma	25	Hong Kong SAR: dollar
26	Hungary: forint	27	Indonesia: rupiah
28	Ireland: punt	29	Israel: shekel
30	Italy: lira	31	Japan: yen
32	Korea, Republic of (South): won	33	Luxembourg: franc
34	Mexico: peso	35	Malaysia: ringgit
36	Netherlands: guilder	37	Norway: krone
38	New Zealand: dollar	39	Philippines: peso
40	Poland: zloty (obsolete, use 89)	41	Portugal: escudo
42	Romania: leu	43	Russia: rouble (obsolete, use 90)
44	Sweden: kroner	45	Singapore: dollar
46	Thailand: baht	47	Taiwan: dollar
48	European Currency Unit (ECU)	49	Yugoslavia: dinar (obsolete, use 91)
50	South Africa: rand	56	Argentina: peso
57	Bermuda: dollar	58	Bolivia: boliviano
59	Brazil: cruzeiro real (obsolete, use 88)	60	Bahamas: dollar
61	Chile: peso	62	Colombia: peso
63	Costa Rica: colon	64	Czech Republic: koruna
65	Dominican Republic: peso	66	Ecuador: sucre
67	Egypt: pound	68	Honduras: lempira
69	India: rupee	70	Jamaica: dollar

Value	Meaning	Value	Meaning
71	Jordan: dinar	72	Kuwait: dinar
73	Macao SAR: pataca	74	Nicaragua: cordoba oro
75	Panama: balboa	76	Peru: nuevo sol
77	Pakistan: rupee	78	Paraguay: guarani
79	Saudi Arabia: riyal	80	Slovenia: tolar
81	Slovakia: koruna	82	El Salvador: colon
83	Turkey: new lira	84	Trinidad and Tobago: dollar
85	Uruguay: peso	86	Venezuela: bolivar
87	Viet Nam: dong	88	Brazil: real
89	Poland: zloty	90	Russia: rouble
91	Serbia: dinar	92	Belarus: ruble
93	Ukraine: hryvnia	94	Afghanistan: afghani
95	Albania: lek	96	Algeria: dinar
97	Andorra: peseta	98	Angola: kwanza
99	East Caribbean Dollar	100	Armenia: dram
101	Aruba: guilder	102	Azerbaijan: manat
103	Bahrain: dinar	104	Bangladesh: taka
105	Barbados: dollar	106	Belarus: ruble
107	Belize: dollar	108	CFA Franc BCEAO
109	Bhutan: ngultrum	110	Bosnia and Herzegovina: convertible marks
111	Botswana: pula	112	Brunei: dollar
113	Bulgaria: lev (historic)	114	Bulgaria: lev
115	Burundi: franc	116	Cambodia: riel
117	CFA Franc BEAC	118	Cape Verde: escudo
119	Cayman Islands: dollar	120	Comoros: franc
121	Congo (DRC): franc	122	Croatia: kuna
123	Cuba: peso	124	Cyprus: pound
125	Djibouti: franc	126	Timor-Leste: escudo
127	Eritrea: nakfa	128	Estonia: kroon
129	Ethiopia: birr	130	Falkland Islands (Islas Malvinas): pound
131	Fiji Islands: dollar	132	CFP Franc

Value	Meaning	Value	Meaning
133	The Gambia: dalasi	134	Georgia: lari
135	Ghana: cedi	136	Gibraltar: pound
137	Guatemala: quetzal	138	Guinea: franc
139	Guinea-Bissau: peso	140	Guyana: dollar
141	Haiti: gourde	142	Iceland: krona
143	Iran: rial	144	Iraq: dinar
145	Kazakhstan: tenge	146	Kenya: shilling
147	North Korean: won	148	Kyrgyzstan: som
149	Laos: kip	150	Latvia: lats
151	Lebanon: pound	152	Lesotho: loti
153	Liberia: dollar	154	Libya: dinar
155	Lithuania: litus	156	Former Yugoslav Republic of Macedonia: denar
157	Madagascar: franc	158	Malawi: kwacha
159	Maldives: rufiyaa	160	Malta: lira
161	Mauritania: ouguiya	162	Mauritius: rupee
163	Moldova: leu	164	Mongolia: tugrik
165	Morocco: dirham	166	Mozambique: metical
167	Myanmar: kyat	168	Namibia: dollar
169	Nepal: rupee	170	Bonaire, Curaçao, Saba, Sint Eustatius, and Sint Maarten: guilder
171	Nigeria: naira	172	Oman: rial
173	Papua New Guinea: kina	174	Qatar: rial
175	Rwanda: franc	176	Saint Helena, Ascension and Tristan da Cunha: pound
177	Samoa: tala	178	São Tomé and Príncipe: dobra
179	Seychelles: rupee	180	Seirra Leone: leone
181	Solomon Islands: dollar	182	Somalia: shilling
183	Sri Lanka: rupee	184	Sudan: dinar
185	Suriname: guilder	186	Swaziland: lilangeni
187	Syria: pound	188	Tajikistan: ruble
189	Tajikistan: somoni	190	Tanzania: shilling
191	Tonga: pa'anga	192	Tunisia: dinar

Value	Meaning	Value	Meaning
193	Turkmenistan: manat	194	Uganda: shilling
195	United Arab Emirates: dirham	196	Uzbekistan: sum
197	Vanuatu: vatu	198	Yemen: rial
199	Zambia: kwacha	200	Zimbabwe: dollar
201	Venezuela: bolivar fuerte	202	Madagascar: ariary
203	Serbia: dinar		

2.5.10.3 vFormatString

The **vFormatString** custom structure is a string that specifies the formatting information to use when representing a value as a string. It can be used to format either a numeric value or a date and time value.

The formatting information has the following syntax when formatting a numeric value.

{*index*[:*formatString*]} [{*index*}]

Elements in square brackets are optional. Curly brackets are required characters.

The formatting information has the following syntax when formatting a date and time value.

formatString

index and *formatString* fields are described in the following table.

Value	Meaning
index	The item to be formatted. Index zero is the value to be displayed, and index 1 is the optional display unit string. MUST be 0 or 1.
formatString	A numeric format string or a date and time format string, as described in [MSDN-FormattingTypes].

2.5.10.4 vLanguageID

The **vLanguageID** custom structure is an unsigned integer that specifies an **LCID**.

2.5.10.5 vServerAction

The **vServerAction** custom structure is an unsigned integer that specifies the custom formatting to apply to a string.

It MUST be a value from the following table.

Value	Meaning
0	Apply no custom formatting

Value	Meaning
1	Convert string to upper case
2	Convert string to lower case

2.5.10.6 vUnitLabel

The **vUnitLabel** custom structure is an unsigned integer that specifies the form and casing **rules** for formatting the unit of a numeric value as a string. Casing is performed using a **culture name** of "".

It MUST be a value from the following table.

Value	Meaning
0	No unit displayed
1	Localized abbreviated form, upper case
2	Localized long form, upper case
3	Universal upper case
4	Localized abbreviated form, lower case
5	Localized long form, lower case
6	Universal lower case

3 Structure Examples

The Visio Graphics Service (.vdw) File Format contains a "VisioServerData" stream as well as other supporting **streams**. This section describes what some of the components within the "VisioServerData" stream look like for some simple documents.

3.1 Document with a Shape on a Page

This section describes some of the contents of the "VisioServerData" stream for a Web drawing consisting of one drawing page with a rectangle with blue fill color and the text "Sample Text" on it.

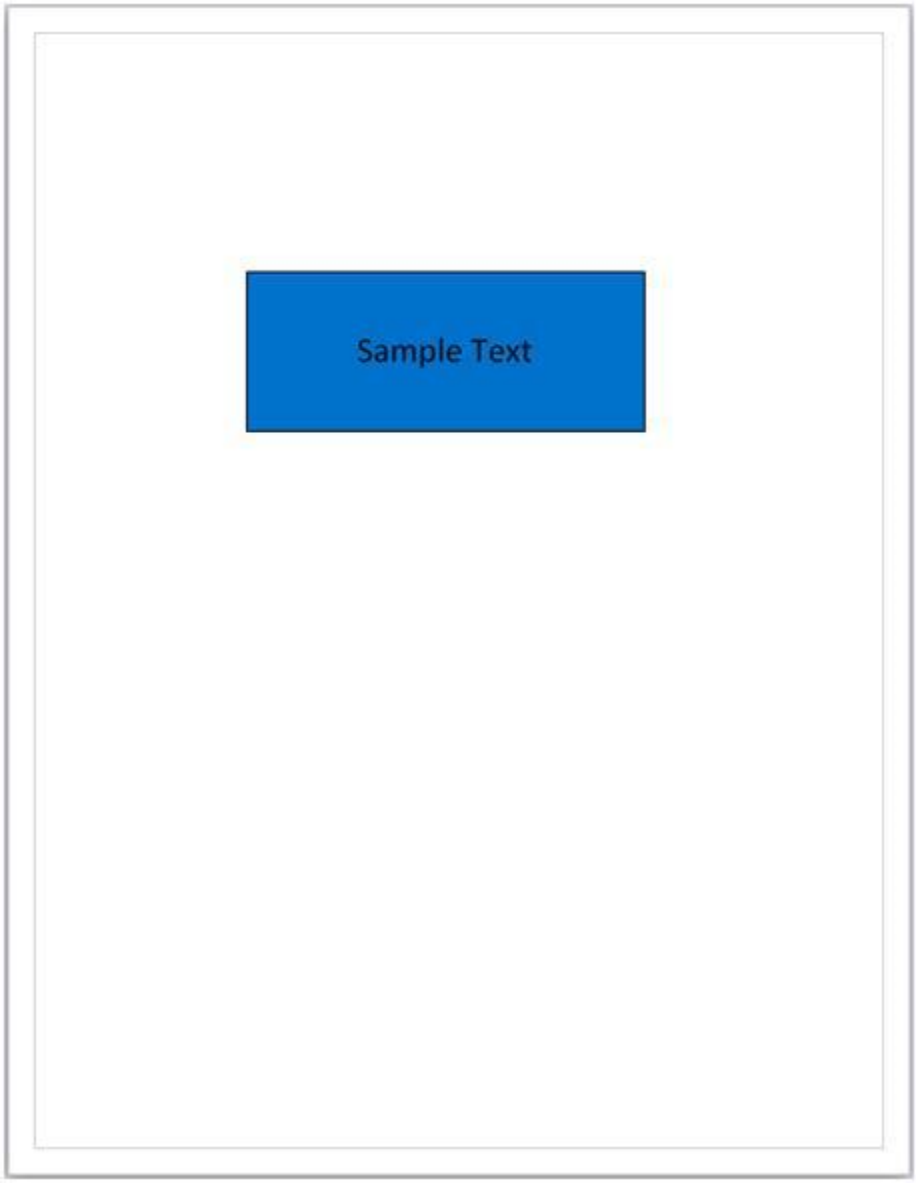

Figure 2: Document with Shape on a Page

3.1.1 App XML Part

```
<Properties
    xmlns="http://schemas.openxmlformats.org/officeDocument/2006/extended-properties"
xmlns:vt="http://schemas.openxmlformats.org/officeDocument/2006/docPropsVTypes">
    <Application>Microsoft Visio 2010</Application>
    <Company>Microsoft Corporation</Company>
    <AppVersion>14.0.4023.1000</AppVersion>
    <DocumentVersion>14.0.14</DocumentVersion>
    <DocumentLanguage>1033</DocumentLanguage>
    <DocumentMeasurementSystem>US Units</DocumentMeasurementSystem>
    <DocumentCurrencyID>11</DocumentCurrencyID>
    <PagesMetaData>
        <PageMetaData ID="1" Name="Page-1"/>
    </PagesMetaData>
</Properties>
```

The following table provides more information about element values in the preceding sample App XML.

Element Name	Value	Notes
Application	Microsoft Visio 2010	The name of the application that created the document is "Microsoft Visio 2010".
Company	Microsoft Corporation	The name of the company that created the document is "Microsoft Corporation".
AppVersion	14.0.4023.1000	Version of the application that created the document is 14.0.4023.1000.
DocumentVersion	14.0.14	Version of the document is 14.0.14.
DocumentLanguage	1033	**LCID** of the document is 1033, which is the .Net language identifier for US English.
DocumentMeasurementSystem	US Units	Measurement system of the document is US Units.
DocumentCurrencyID	11	The currency ID that is used in the document is 11, which is the identifier for United States dollar.
ID	1	Index of the drawing page is 1.
Name	Page-1	Name of the drawing page is "Page-1".

3.1.2 ShapeInfo XML Part

```
<Page
    Name="Page-1"
    Zoom="-1.000000"
    OffsetX="407"
    OffsetY="528"
    DefaultUnitsText="50"
    DefaultUnitsAngle="81"
    DefaultUnitsDuration="44"
    DefaultUnitsPage="73">
    <Pages>
        <PageInfo Name="Page-1" />
    </Pages>
    <ShapeInfo
        Name=" 1"
        DisplayName="Sheet.1"
```

```
        Layout="{"type":"2D","bounds":{"x":215,"y":239,"width":362,"height":146}}" />
</Page>
```

The following table provides more information about attribute values in the preceding sample ShapeInfo XML.

Attribute Name	Value	Notes
Name	Page-1	The name of the drawing page is "Page-1".
Zoom	-1.000000	The **zoom level** at which the drawing page is initially displayed is -1.0, which specifies that the drawing page is zoomed to fit the entire drawing page in the **view**.
OffsetX	407	The x-coordinate in pixels of the point on the drawing page that is centered in the view when the drawing page is initially displayed is 407. The value is relative to the top, left corner of the drawing page and is expressed in the coordinate system of the view that displays the drawing page.
OffsetY	528	The y-coordinate in pixels of the point on the drawing page that is centered in the view when the drawing page is initially displayed is 528. The value is relative to the top, left corner of the drawing page and is expressed in the coordinate system of the view that displays the drawing page.
DefaultUnitsText	50	"Points" is used as the default typographic measurement unit in formula evaluations for this drawing page.
DefaultUnitsAngle	81	"Decimal degrees" is used as the default angle measurement unit in formula evaluations for this drawing page.
DefaultUnitsDuration	44	"Days" is used as the default duration measurement unit in formula evaluations for this drawing page.
DefaultUnitsPage	73	"Inches (fractional)" is used as the default unit for length measurement in formula evaluations for this drawing page.
Name	_1	The shape identifier is "_1".
DisplayName	Sheet.1	The shape name used for display purposes is "Sheet.1".
Layout	{"type":"2D","bounds":{"x":215,"y":239,"width":362,"height":146}}	215 and 239 are the upper left x-coordinate and y-coordinate, respectively, of the **bounding rectangle** of the shape in **pixels**. 362 and 146 are the width and height, respectively, of the bounding rectangle of the shape in pixels.

3.1.3 ShapeOutline XML Part

```
<Page>
    <Shapes>
```

```
        <Shape Name=" 1">
            <Path Shape="poly" Data="216,384,576,384,576,240,216,240,216,384"/>
        </Shape>
    </Shapes>
</Page>
```

The following table provides more information about attribute values in the preceding sample ShapeOutline XML.

Attribute Name	Value	Notes
Name	_1	The shape identifier of the shape is "_1".
Shape	Poly	The type of outline specified for the shape is polygon.
Data	216,384,576,384,576,240,216,240,216,384	The Cartesian coordinates in **pixels** of each vertex of the polygon are (216,384), (576,384), (576,240), (216,240). The first vertex (216,384) is repeated to close the polygon.

3.1.4 XAML

```
<Canvas
    Height="1056"
    Name="Workspace"
    Width="816"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">

    <Canvas Name="D">
        <Canvas.Background>
            <SolidColorBrush Color="#FFFFFFFF"/>
        </Canvas.Background>

        <Canvas
            Tag="Shape"
            Name="_1"
            RenderTransform="1, 0, 0, 1, 216, -672">
            <Rectangle
                Canvas.Left="-0.5"
                Canvas.Top="911.5"
                Height="145"
                Name=" 1 G0 P0"
                Width="361"
                StrokeThickness="1"
                StrokeDashCap="Round"
                StrokeEndLineCap="Round"
                StrokeLineJoin="Round"
                StrokeStartLineCap="Round">

                <Rectangle.Stroke>
                    <SolidColorBrush Color="#FF000000"/>
                </Rectangle.Stroke>

                <Rectangle.Fill>
                    <SolidColorBrush Color="#FF0070C0"/>
                </Rectangle.Fill>

            </Rectangle>

            <Canvas>
                <Glyphs
```

```
                    FontRenderingEmSize="32"
                    FontUri="CommonFonts/4C1AABE5-BA71-4F2C-8263-66731F995C4B.odttf"
                    OriginX="100.5"
                    OriginY="993.6"
                    UnicodeString="Sample Text">

                        <Glyphs.Fill>
                            <SolidColorBrush Color="#FF000000"/>
                        </Glyphs.Fill>

                </Glyphs>
            </Canvas>
        </Canvas>
    </Canvas>
</Canvas>
```

The following table provides more information about elements in the preceding sample XAML.

Node	Notes
Top most canvas node with name "Workspace"	Canvas for the container of the whole drawing page. 816 and 1056 are the width and height respectively at 100% zoom in **pixels**.
Canvas node with name "D"	Canvas for the drawing page.
Canvas node with name "_1"	Canvas for the shape. Identifier of the shape is "_1".
Rectangle node	Graphical representation of the shape.
Rectangle.Stroke node	The line color of the shape is black.
Rectangle.Fill node	The fill color of the shape is blue.
Canvas node with Glyphs element as its child node	Canvas that groups all the text of the shape.
Glyphs node	Representation of the text in the shape.
FontRenderingEmSize attribute	The pixel size of the text is 32.
FontUri attribute	**Font** of the text is Calibri.
UnicodeString attribute	The text in the shape is "Sample Text".
Glyphs.Fill node	The color of the text is black.

3.2 Document with Recalculated Visual Properties

This section describes components of the "VisioServerData" stream for a Web drawing consisting of one drawing page containing a rectangle shape with text element and formatting element data graphics. The rectangle has a data binding to a cell within a **row** of data in a recordset named "Current Product List" in the "Northwind" database. The two **fields** that identify the cells within the row are "ProductID" and "ProductName". The "ProductID" field is used as the **primary key**. The rectangle shape has associated text that displays the label "ProductName" and its value. Also, the fill color of the rectangle shape is determined by the value of the "ProductName" field as specified in the following table.

Value of the "ProductName" field	Fill color of the rectangle
"Chai"	Red
"Coffee"	Yellow
Neither "Chai" nor "Coffee"	Blue

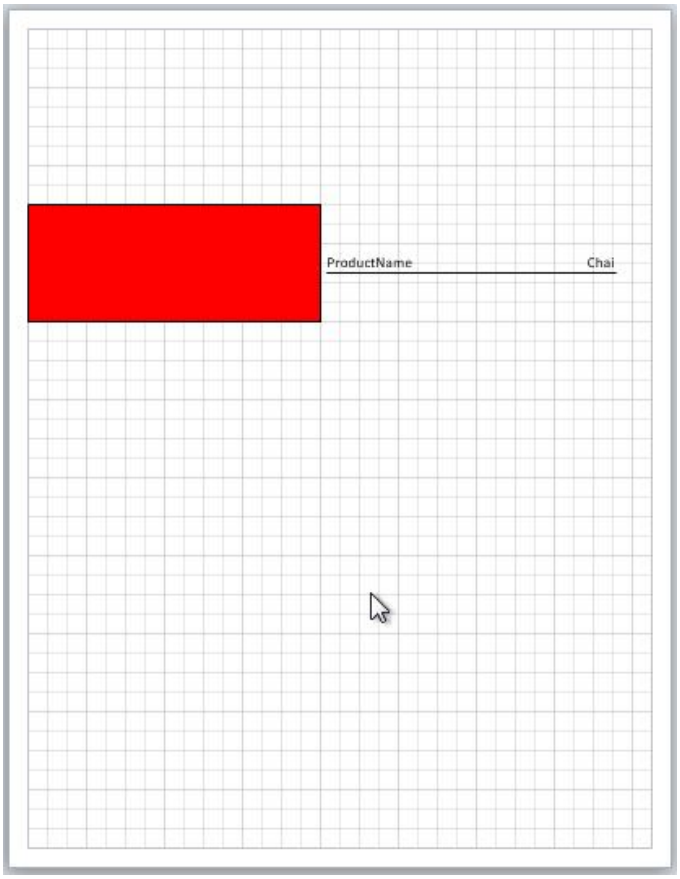

Figure 3: Document with Recalculated Visual Properties

3.2.1 DataBinding XML Part

```
<BindingConnections>
    <BindingConnection RecordsetID="2">
        <Bindings>
            <Binding BindingID="1">
                <PrimaryKeys>
                    <PrimaryKey Value="1" />
                </PrimaryKeys>
            </Binding>
        </Bindings>
    </BindingConnection>
</BindingConnections>
```

The following table provides more information about attribute values in the preceding sample DataBinding XML.

Attribute Name	Value	Notes
RecordsetID	2	The identifier of the recordset that has **rows** of data bound to shapes is 2.
BindingID	1	The CT_Binding identifier is 1.
Value	1	The value of the **field** that is a component of the **primary key** (ProductID) is 1.

3.2.2 DataConnection XML Part

```
<Connections>
    <DataConnections>
        <DataConnection ID="1" ConnectionString="Provider=SQLOLEDB.1;Integrated
Security=SSPI;Persist Security Info=True;Initial Catalog=Northwind;Data
Source=SQLServerDataSource;Use Procedure for Prepare=1;Auto Translate=True;Packet
Size=4096;Workstation ID=MACHINENAME;Use Encryption for Data=False;Tag with column collation
when possible=False" />
    </DataConnections>
    <DataRecordsets>
        <DataRecordset ID="2" ConnectionID="1" Command="select * from
"Northwind"."dbo"."Current Product List"" Pages="1">
            <PrimaryKeys>
                <PrimaryKey Key="ProductID" />
            </PrimaryKeys>
            <DataColumns>
                <DataColumn Name="ProductID" DisplayName="ProductID" Type="3" />
                <DataColumn Name="ProductName" DisplayName="ProductName" Type="202" />
            </DataColumns>
        </DataRecordset>
    </DataRecordsets>
</Connections>
```

The following table provides more information about attribute values in the preceding sample DataConnection XML.

Attribute Name	Value	Notes
ID	1	The identifier of the data connection is 1.
ConnectionString	Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=True;Initial Catalog=Northwind;Data Source=SQLServerDataSource;Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation ID=MACHINENAME;Use Encryption for Data=False;Tag with column collation when possible=False	The **connection string** to the **data source** is "Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist Security Info=True;Initial Catalog=Northwind;Data Source=SQLServerDataSource;Use Procedure for Prepare=1;Auto Translate=True;Packet Size=4096;Workstation ID=MACHINENAME;Use Encryption for Data=False;Tag with column collation when possible=False".
ID	2	The identifier of the recordset is 2.
ConnectionID	1	The identifier of the data connection corresponding to the recordset is 1.

Attribute Name	Value	Notes
Command	select * from "Northwind"."dbo"."Current Product List"	The **query** for the recordset is "select * from "Northwind"."dbo"."Current Product List"".
Pages	1	The drawing page with index 1 is the only drawing page that contains shapes bound to data in the recordset.
Key	ProductID	A **field** named "ProductID" is a component of the **primary key** of the recordset.
Name	ProductID	The recordset has a field named "ProductID".
DisplayName	ProductID	The display name of the field is "ProductID".
Type	3	The **ADO data type** of the field is integer.
Name	ProductName	The recordset has a field named "ProductName".
DisplayName	ProductName	The display name of the field is "ProductName".
Type	202	The ADO data type of the field is wide variable character.

3.2.3 DataGraphic XML Part

```
<DataGraphics ColorTable="0,0,0 255,255,255 255,0,0 0,255,0 0,0,255 255,255,0 255,0,255
0,255,255 128,0,0 0,128,0 0,0,128 128,128,0 128,0,128 0,128,128 192,192,192 230,230,230
205,205,205 179,179,179 154,154,154 128,128,128 102,102,102 77,77,77 51,51,51 26,26,26
0,112,192 150,175,207 191,206,225 0,145,251 240,240,240 0,0,0 166,210,74 200,227,142
31,71,125 69,91,23 157,187,97 255,255,255 163,184,204 54,73,89 204,0,68 69,94,115 255,0,0
255,60,60 ">
    <Page
        PageName="Page-1"
        ViewTransform="96.000000 0.000000 0.000000 -96.000000 0.000000 1056.000000">

        <Shape
            ID="{020D279F-6024-4139-8CDF-A7637F3217E7}"
            ShapeName="_1"
            DataRefs="1,ProductName">

                <Formulas>
                    <Sheet
                        Name=" 1"
                        FillBackground="1.242.131.0:2 1.5:32 <>:14 173519920:160 1.1.3.0:2
191:98 206:98 225:98 RGB():128 150:98 175:98 207:98 RGB():128 LUMDIFF():128 TINT():128
173519989:163 173519920:164 :147 FALSE:97 CELLISTHEMED():128 173519972:160 191:98 206:98
225:98 RGB():128 173519988:163 173519972:164 :147 255:98 60:98 60:98 RGB():128 173519988:164
173519989:164 THEMEGUARD():128"
                        FillForeground="1.242.131.0:2 1.5:32 <>:14 165284145:160
1.242.131.0:2 165284212:163 165284145:164 :147 FALSE:97 CELLISTHEMED():128 165284195:160
150:98 175:98 207:98 RGB():128 165284211:163 165284195:164 :147 0:98 112:98 192:98 RGB():128
165284211:164 165284212:164 THEMEGUARD():128" />
```

```
                </Formulas>
                <FormulaReferences
                    Ref-1.1.3.0="1.242.131.0:2 1.5:32 <>:14 165284145:160 1.242.131.0:2
165284212:163 165284145:164 :147 FALSE:97 CELLISTHEMED():128 165284195:160 150:98 175:98
207:98 RGB():128 165284211:163 165284195:164 :147 0:98 112:98 192:98 RGB():128 165284211:164
165284212:164 THEMEGUARD():128"
                    Ref-1.1.3.1="1.242.131.0:2 1.5:32 <>:14 173519920:160 1.1.3.0:2 191:98
206:98 225:98 RGB():128 150:98 175:98 207:98 RGB():128 LUMDIFF():128 TINT():128 173519989:163
173519920:164 :147 FALSE:97 CELLISTHEMED():128 173519972:160 191:98 206:98 225:98 RGB():128
173519988:163 173519972:164 :147 255:98 60:98 60:98 RGB():128 173519988:164 173519989:164
THEMEGUARD():128"
                    Ref-1.242.131.0=""Chai":96 _1,"ProductName",96,1:1 TRUE:97
3;STRSAME():129 173438756:160 255:98 0:98 0:98 RGB():128 173438833:163 173438756:164 :147
"Coffee":96 _1,"ProductName",96,1:1 TRUE:97 3;STRSAME():129 173438821:160 255:98 255:98 0:98
RGB():128 173438832:163 173438821:164 :147 1.5:32  173438832:164 173438833:164" />
        </Shape>
    </Page>
</DataGraphics>
```

The following table provides more information about attribute values in the preceding sample DataGraphic XML.

Attribute Name	Value	Notes
ColorTable	0,0,0 255,255,255 255,0,0 0,255,0 0,0,255 255,255,0 255,0,255 0,255,255 128,0,0 0,128,0 0,0,128 128,128,0 128,0,128 0,128,128 192,192,192 230,230,230 205,205,205 179,179,179 154,154,154 128,128,128 102,102,102 77,77,77 51,51,51 26,26,26 0,112,192 150,175,207 191,206,225 0,145,251 240,240,240 0,0,0 166,210,74 200,227,142 31,71,125 69,91,23 157,187,97 255,255,255 163,184,204 54,73,89 204,0,68 69,94,115 255,0,0 255,60,60	The color table used to render data graphics specified using the **RGB** color model are (0,0,0), (255,255,255), (255,0,0), (0,255,0), (0,0,255), (255,255,0), (255,0,255), (0,255,255), (128,0,0), (0,128,0), (0,0,128), (128,128,0), (128,0,128), (0,128,128), (192,192,192), (230,230,230), (205,205,205), (179,179,179), (154,154,154), (128,128,128), (102,102,102), (77,77,77), (51,51,51), (26,26,26), (0,112,192), (150,175,207), (191,206,225), (0,145,251), (240,240,240), (0,0,0), (166,210,74), (200,227,142), (31,71,125), (69,91,23), (157,187,97), (255,255,255), (163,184,204), (54,73,89), (204,0,68), (69,94,115), (255,0,0), (255,60,60).
PageName	Page-1	The name of the drawing page is "Page-1".
ViewTransform	96.000000 0.000000 0.000000 -96.000000 0.000000 1056.000000	96.000000 0.000000 0.000000 -96.000000 0.000000 1056.000000 represent a 3-by-3 affine transformation matrix as specified in CT_DataGraphics_Page.
ID	{020D279F-6024-4139-8CDF-A7637F3217E7}	The identifier of the data graphic is {020D279F-6024-4139-8CDF-A7637F3217E7}
ShapeName	_1	The identifier of the shape is "_1".
DataRefs	1,ProductName	The **BindingID** attribute of the CT_Binding element in DataBinding XML Part corresponding to the recordset external data **row** the shape is bound to is one. The **Name** attribute of the CT_DataColumn element in

Attribute Name	Value	Notes
		DataConnection XML Part corresponding to the **field** in the recordset the text run is bound to is "ProductName".
Name	_1	The name of the corresponding sheet element data graphic is "_1".
FillBackground	1.242.131.0:2 1.5:32 <>:14 173519920:160 1.1.3.0:2 191:98 206:98 225:98 RGB():128 150:98 175:98 207:98 RGB():128 LUMDIFF():128 TINT():128 173519989:163 173519920:164 :147 FALSE:97 CELLISTHEMED():128 173519972:160 191:98 206:98 225:98 RGB():128 173519988:163 173519972:164 :147 255:98 60:98 60:98 RGB():128 173519988:164 173519989:164 THEMEGUARD():128	The formula expression used to recalculate the background color which is a part of the fill property of the corresponding sheet element data graphic is "1.242.131.0:2 1.5:32 <>:14 173519920:160 1.1.3.0:2 191:98 206:98 225:98 RGB():128 150:98 175:98 207:98 RGB():128 LUMDIFF():128 TINT():128 173519989:163 173519920:164 :147 FALSE:97 CELLISTHEMED():128 173519972:160 191:98 206:98 225:98 RGB():128 173519988:163 173519972:164 :147 255:98 60:98 60:98 RGB():128 173519988:164 173519989:164 THEMEGUARD():128".
FillForeground	1.242.131.0:2 1.5:32 <>:14 165284145:160 1.242.131.0:2 165284212:163 165284145:164 :147 FALSE:97 CELLISTHEMED():128 165284195:160 150:98 175:98 207:98 RGB():128 165284211:163 165284195:164 :147 0:98 112:98 192:98 RGB():128 165284211:164 165284212:164 THEMEGUARD():128	The formula expression used to recalculate the foreground color which is a part of the fill property of the corresponding sheet element data graphic is "1.242.131.0:2 1.5:32 <>:14 165284145:160 1.242.131.0:2 165284212:163 165284145:164 :147 FALSE:97 CELLISTHEMED():128 165284195:160 150:98 175:98 207:98 RGB():128 165284211:163 165284195:164 :147 0:98 112:98 192:98 RGB():128 165284211:164 165284212:164 THEMEGUARD():128".
Ref-1.1.3.0	1.242.131.0:2 1.5:32 <>:14 165284145:160 1.242.131.0:2 165284212:163 165284145:164 :147 FALSE:97 CELLISTHEMED():128 165284195:160 150:98 175:98 207:98 RGB():128 165284211:163 165284195:164 :147 0:98 112:98 192:98 RGB():128 165284211:164 165284212:164 THEMEGUARD():128	"Ref-1.1.3.0" is the name of the formula reference for the formula expression "1.242.131.0:2 1.5:32 <>:14 165284145:160 1.242.131.0:2 165284212:163 165284145:164 :147 FALSE:97 CELLISTHEMED():128 165284195:160 150:98 175:98 207:98 RGB():128 165284211:163 165284195:164 :147 0:98 112:98 192:98 RGB():128 165284211:164 165284212:164 THEMEGUARD():128".
Ref-1.1.3.1	1.242.131.0:2 1.5:32 <>:14 173519920:160 1.1.3.0:2 191:98 206:98 225:98 RGB():128 150:98 175:98 207:98 RGB():128 LUMDIFF():128 TINT():128 173519989:163 173519920:164 :147 FALSE:97 CELLISTHEMED():128 173519972:160 191:98 206:98 225:98 RGB():128 173519988:163 173519972:164 :147 255:98 60:98 60:98 RGB():128 173519988:164 173519989:164 THEMEGUARD():128	"Ref-1.1.3.1" is the name of the formula reference for the formula expression "1.242.131.0:2 1.5:32 <>:14 173519920:160 1.1.3.0:2 191:98 206:98 225:98 RGB():128 150:98 175:98 207:98 RGB():128 LUMDIFF():128 TINT():128 173519989:163 173519920:164 :147 FALSE:97 CELLISTHEMED():128 173519972:160 191:98 206:98 225:98 RGB():128 173519988:163 173519972:164 :147 255:98 60:98 60:98 RGB():128 173519988:164 173519989:164

Attribute Name	Value	Notes
		THEMEGUARD():128".
Ref-1.242.131.0	"Chai":96 _1,"ProductName",96,1:1 TRUE:97 3;STRSAME():129 173438756:160 255:98 0:98 0:98 RGB():128 173438833:163 173438756:164 :147 "Coffee":96 _1,"ProductName",96,1:1 TRUE:97 3;STRSAME():129 173438821:160 255:98 255:98 0:98 RGB():128 173438832:163 173438821:164 :147 1.5:32 173438832:164 173438833:164	"Ref-1.242.131.0" is the name of the formula reference for the formula expression ""Chai":96 _1,"ProductName",96,1:1 TRUE:97 3;STRSAME():129 173438756:160 255:98 0:98 0:98 RGB():128 173438833:163 173438756:164 :147 "Coffee":96 _1,"ProductName",96,1:1 TRUE:97 3;STRSAME():129 173438821:160 255:98 255:98 0:98 RGB():128 173438832:163 173438821:164 :147 1.5:32 173438832:164 173438833:164".

3.2.4 ShapeInfo XML Part

```
<Page
    Name="Page-1"
    Zoom="0.500000"
    OffsetX="407"
    OffsetY="528"
    DefaultUnitsText="50"
    DefaultUnitsAngle="81"
    DefaultUnitsDuration="44"
    DefaultUnitsPage="73">
    <Pages>
        <PageInfo Name="Page-1" />
    </Pages>
    <ShapeInfo
        Name=" 1"
        DisplayName="Sheet.1"
        Layout="{"type":"2D","bounds":{"x":23,"y":239,"width":727,"height":146}}">
        <ShapeDataItems>
            <ShapeData
                Name="ProductID"
                FormattedValue="1"
                Value="1.000000"
                Format="{0:0.####} {1}"
                Type="2"
                LangID="1033"
                UnitLabel="0"
                Unit="32"
                BindingID="1" />
            <ShapeData
                Name="ProductName"
                FormattedValue="Chai"
                Value="Chai"
                Format="{0}"
                Type="0"
                LangID="1033"
                UnitLabel="0"
                Unit="96"
                BindingID="1" />
        </ShapeDataItems>
    </ShapeInfo>
</Page>
```

The following table provides more information about attribute values in the preceding sample ShapeInfo XML.

Attribute Name	Value	Notes
Name	Page-1	Name of the drawing page is "Page-1".
Zoom	0.500000	The **zoom level** at which the drawing page is initially displayed is 50%.
OffsetX	407	The x-coordinate in pixels of the point on the drawing page that is centered in the **view** when the drawing page is initially displayed is 407. The value is relative to the top, left corner of the drawing page and is expressed in the coordinate system of the view that displays the drawing page.
OffsetY	528	The y-coordinate in pixels of the point on the drawing page that is centered in the view when the drawing page is initially displayed is 528. The value is relative to the top, left corner of the drawing page and is expressed in the coordinate system of the view that displays the drawing page.
DefaultUnitsText	50	"Points" is used as the default typographic measurement unit in formula evaluations for this drawing page.
DefaultUnitsAngle	81	"Decimal degrees" is used as the default angle measurement unit in formula evaluations for this drawing page.
DefaultUnitsDuration	44	"Days" is used as the default duration measurement unit in formula evaluations for this drawing page.
DefaultUnitsPage	73	"Inches (fractional)" is used as the default unit for length measurement in formula evaluations for this drawing page, if none is explicitly specified in the formula expression.
Name	_1	The shape identifier is "_1".
DisplayName	Sheet.1	The shape name used for display purposes is "Sheet.1".
Layout	{"type":"2D","bounds":{"x":23," y":239, "width":727, "height":146}}	23 and 239 are the upper left x-coordinate and y-coordinate, respectively, of the **bounding rectangle** of the shape in **pixels**. The coordinates are relative to the top, left corner of the drawing page that contains the shape. 727 and 146 are the width and height, respectively, of the bounding rectangle of the shape in pixels.
Name	ProductID	The label of the shape data item is "ProductID".
FormattedValue	1	The formatted value of the shape data item is 1.
Value	1.000000	The value of the shape data item is 1.000000.
Format	{0:0.####} {1} {0}	"0.####" is the format string used to format the value in the first ShapeData element. A unit string will be displayed after the value if the value has a unit associated with it. "{0}" is the format string used to format the value in the second ShapeData element. No formatting is required. The formatted value is the same as the

Attribute Name	Value	Notes
		value.
Type	2	The value is a numerical value.
LangID	1033	The **LCID** used to format the value is 1033, which maps to US English.
UnitLabel	0	No **rules** with respect to format (abbreviated or full) and casing are applied to the unit label.
Unit	32	The vUnitType of the value is ptgNum.
BindingID	1	The CT_Binding identifier for the shape data item is 1.

3.2.5 ShapeTextBinding XML Part

```
<Page Name="">
    <ShapeText Name="_3">
        <TextRun
            Name="txt_3_fld0"
            Formula="_1,&#34;ProductName&#34;,96,1:1 &#34;{0:0.####}@0@0@11@0@1033@0&#34;:96
:65 :252 1033:98 0:32 6;FORMATEX():129    "
            Format="{0:0.####}@0@0@11@0@1033@0"
            DataRefs="1,ProductName"/>
    </ShapeText>
</Page>
```

The following table provides more information about attribute values in the preceding sample ShapeTextBinding XML.

Attribute Name	Value	Notes
Name	""	Value is ignored
Name	_3	The identifier of the corresponding text element data graphic is "_3".
Name	txt_3_fld0	The identifier of the corresponding text element data graphic is "txt_3_fld0".
Formula	_1,"ProductName",96,1:1 "{0:0.####}@0@0@11@0@1033@0":96 :65 :252 1033:98 0:32 6;FORMATEX():129	The formula expression used to recalculate the content of the text run in the text element data graphic upon **refresh** is "_1,"ProductName",96,1:1 "{0:0.####}@0@0@11@0@1033@0":96 :65 :252 1033:98 0:32 6;FORMATEX():129".
Format	{0:0.####}@0@0@11@0@1033@0	The format string used to format the result of the formula evaluation is "{0:0.####}". No custom formatting needs to be applied to string. No **rules** with respect to form (abbreviated or full) and casing are applied to the unit label.

Attribute Name	Value	Notes
		The **LCID** used to format the value is 1033, which maps to US English.
DataRefs	1,ProductName	The BindingID attribute of the CT_Binding element in DataBinding XML Part corresponding to the external data **row** the text run is bound to is one. The Name attribute of the CT_DataColumn element in DataConnection XML Part corresponding to the **field** in the recordset the text run is bound to is "ProductName".

3.2.6 XAML

```
<Canvas
    Height="1056"
    Name="Workspace" Width="816"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation">

    <Canvas Name="D">

        <Canvas.Background>
            <SolidColorBrush Color="#FFFFFFFF"/>
        </Canvas.Background>

        <Canvas
            Tag="Shape"
            Name="_1"
            RenderTransform="1, 0, 0, 1, 24, -672">

            <Rectangle
                Canvas.Left="-0.5"
                Canvas.Top="911.5"
                Height="145"
                Name="_1_G0_P0"
                Width="361"
                StrokeThickness="1"
                StrokeDashCap="Round"
                StrokeEndLineCap="Round"
                StrokeLineJoin="Round"
                StrokeStartLineCap="Round">

                <Rectangle.Stroke>
                    <SolidColorBrush Color="#FF000000"/>
                </Rectangle.Stroke>

                <Rectangle.Fill>
                    <SolidColorBrush Color="#FFFF0000"/>
                </Rectangle.Fill>

            </Rectangle>

            <Canvas Name="_2">

                <Canvas.RenderTransform>
                    <MatrixTransform Matrix="1, 0, 0, 1, 366, -57.544364"/>
                </Canvas.RenderTransform>


                <Path
                    Name="Path1"
```

```
                    Data="M1.92 1054.08 L358.08 1054.08 L358.08 1029.01 L1.92 1029.01 L1.92
1054.08 Z"/>

                <Path
                    Name="Path2"
                    Data="M1.92 1054.08 L358.08 1054.08"/>

                <Path
                    Name="Path3"
                    StrokeThickness="0.5"
                    StrokeDashCap="Round"
                    StrokeEndLineCap="Round"
                    StrokeLineJoin="Round"
                    StrokeStartLineCap="Round"
                    Data="M1.92 1054.08 L358.08 1054.08 L358.08 1029.01 L1.92 1029.01 L1.92
1054.08">
                    <Path.Stroke>
                        <SolidColorBrush Opacity="0" Color="#FF000000"/>
                    </Path.Stroke>

                </Path>

                <Path
                    Name="Path4"
                    StrokeThickness="0.5"
                    StrokeDashCap="Round"
                    StrokeEndLineCap="Round"
                    StrokeLineJoin="Round"
                    StrokeStartLineCap="Round"
                    Data="M1.92 1054.08 L358.08 1054.08">

                    <Path.Stroke>
                        <SolidColorBrush Color="#FF000000"/>
                    </Path.Stroke>

                </Path>

                <Canvas>
                    <Glyphs
                        FontRenderingEmSize="18.667"
                        FontUri="CommonFonts/4C1AABE5-BA71-4F2C-8263-66731F995C4B.odttf"
                        OriginX="3.2534"
                        OriginY="1047.1"
                        UnicodeString="ProductName">

                        <Glyphs.Fill>
                            <SolidColorBrush Color="#FF000000"/>
                        </Glyphs.Fill>

                    </Glyphs>
                </Canvas>

                <Canvas Name="_3">
                    <TextBlock
Tag="Right;Center;;1.2;0.013888888888889;0.013888888888889;0.013888888888889;0.01388888888888
9"
                        Height="0.26116"
                        Visibility="Collapsed"
                        Width="2.545"
                        RenderTransform="96, 0, 0, 96, 113.756384, 1029.008729"
                        TextWrapping="Wrap">

                        <Run
                            Name="txt_3_fld0"
                            FontFamily="Calibri"
                            FontSize="0.19444"
                            Foreground="#FF000000">

                                Chai
```

```
                        </Run>

                    </TextBlock>

                    <Canvas>
                        <Glyphs
                            FontRenderingEmSize="18.667"
                            FontUri="CommonFonts/4C1AABE5-BA71-4F2C-8263-66731F995C4B.odttf"
                            OriginX="323.76"
                            OriginY="1047.1"
                            UnicodeString="Chai">

                            <Glyphs.Fill>
                                <SolidColorBrush Color="#FF000000"/>
                            </Glyphs.Fill>
                        </Glyphs>
                    </Canvas>
                </Canvas>
            </Canvas>
        </Canvas>
    </Canvas>
</Canvas>
```

The following table provides more information about elements in the preceding sample XAML.

Node	Notes
Topmost canvas node with name "Workspace"	Canvas for the container of the whole drawing page. 816 and 1056 are the width and height, respectively, at 100% zoom in **pixels**.
Canvas node with name "D"	Canvas for the drawing page.
Canvas node with name "_1"	Canvas for the rectangle shape. Identifier of the shape is "_1".
Rectangle node	Graphical representation of the rectangle shape.
Rectangle.Stroke node	The line color of the shape is black.
Rectangle.Fill node	The fill color of the shape is red.
Canvas node with name "_2"	Canvas for text callout shape. The identifier of the shape is "_2".
Glyphs node	Representation of the text in the shape.
FontRenderingEmSize attribute	Pixel size of text is 18.667.
FontUri attribute	**Font** of the text is Calibri.
UnicodeString attribute	The text in the shape is "Sample Text".
Glyphs.Fill node	The color of the text is black.
Canvas node with name "_3"	Canvas for shape with identifier "_3".
TextBlock Node within Canvas node with name "_3"	Contains high level information to be able to compose text after it has been refreshed.
Run node with name "txt_3_fld0"	Maps to corresponding TextRun node with name "txt_3_fld0" in the ShapeTextBinding XML Part.

3.3 Formula Evaluation

This example describes how a formula expression is parsed from the file and evaluated. This formula expression returns a PtgColorRGB based on the value obtained for the data bound shape data item named "ProductName". If the value is "Chai" the color returned is 0xFFFF0000 (red). If the value is "Coffee" the color returned is 0xFFFFFF00 (yellow).

The formula expression is persisted in the DataGraphic XML Part as follows.

```
"Chai":96 _1,"ProductName",96,1:1 TRUE:97 3;STRSAME():129  173438756:160 255:98 0:98 0:98
RGB():128 173438833:163 173438756:164 :147 "Coffee":96 _1,"ProductName",96,1:1 TRUE:97
3;STRSAME():129 173438821:160 255:98 255:98 0:98 RGB():128 173438832:163 173438821:164 :147
1.5:32 173438832:164 173438833:164
```

The following table describes the parse tokens in the formula expression and the values of the tokens on the evaluation stack during formula evaluation.

File Representation of Parse Token	Token Parsing Operation	Evaluation Stack State
"Chai":96	This is a PtgStr1, an operand token. The value of the token is pushed onto the stack.	"Chai"
_1,"ProductName",96,1:1	This is a PtgDataBinding, an operand token. The value of the cell in the recordset referenced by the data binding is a PtgStr1. It is pushed onto the stack.	"Coffee" "Chai"
TRUE:97	This is a PtgBool, an operand token. A **Boolean** value of TRUE is pushed onto the stack.	TRUE "Coffee" "Chai"
3;STRSAME():129	This is a PtgFuncVar, a function token for a function that takes three arguments. The top three tokens are popped off the stack and evaluated by the StrSame function. The resulting value of FALSE is pushed onto the stack.	FALSE
173438756:160	This is a PtgJmpF, a control token. It compares the value of the token at the top of the stack to FALSE. The comparison succeeds and all subsequent tokens are parsed and ignored until a PtgJmpLabel token with a value of 173438756 is reached.	FALSE
255:98	This is a PtgUnsWord that is parsed but ignored.	FALSE
0:98	This is a PtgUnsWord that is parsed but ignored.	FALSE
0:98	This is a PtgUnsWord that is parsed but ignored.	FALSE
RGB():128	This is a PtgFunc that is parsed but ignored.	FALSE
173438833:163	This is a PtgJmp that is parsed but ignored.	FALSE
173438756:164	This is a PtgJmpLabel, a control token. Evaluation resumes at this position. Nothing is pushed on to the stack.	FALSE
:147	This is a PtgPop, a control token that pops the top token off the stack.	

File Representation of Parse Token	Token Parsing Operation	Evaluation Stack State
"Coffee":96	This is a PtgStr1, an operand token. The value of the token is pushed onto the stack.	"Coffee"
_1,"ProductName",96,1:1	This is a PtgDataBinding, an operand token. The value of the cell in the recordset referenced by the data binding is pushed onto the stack.	"Coffee" "Coffee"
TRUE:97	This is a PtgBool, an operand token. A Boolean value of TRUE is pushed onto the stack.	TRUE "Coffee" "Coffee"
3;STRSAME():129	This is a PtgFuncVar, a function token for a function that takes three arguments. The top three tokens are popped off the stack and evaluated by the StrSame function. The resulting value of TRUE is pushed onto the stack.	TRUE
173438821:160	This is a PtgJmpF, a control token. It compares the value of the token at the top of the stack to FALSE. The comparison fails and the top token is popped off the stack.	
255:98	This is a PtgUnsWord, an operand token. The numeric value 255 is pushed onto the stack.	255
255:98	This is a PtgUnsWord, an operand token. The numeric value 255 is pushed onto the stack.	255 255
0:98	This is a PtgUnsWord, an operand token. The numeric value 0 is pushed onto the stack.	0 255 255
RGB():128	This is a PtgFunc, a function token for a function that takes three arguments. The top three tokens are popped off the stack and evaluated by the RGB function. The function returns a PtgColorRGB with a value of 0xFFFFFF00. The corresponding decimal value of -256 is pushed onto the stack.	-256
173438832:163	This is a PtgJmp, a control token. All subsequent tokens are parsed and ignored until a PtgJmpLabel token with a value of 173438832 is reached.	-256
173438821:164	This is a PtgJmpLabel that is parsed but ignored.	-256
:147	This is a PtgPop that is parsed but ignored.	-256
1.5:32	This is a PtgNum that is parsed but ignored.	-256
173438832:164	This is a PtgJmpLabel, a control token. Evaluation resumes at this position. Nothing is pushed on to the stack.	-256
173438833:164	This is a PtgJmpLabel, a control token. It is parsed but ignored because another control token is not being evaluated.	-256

At this point all the parse tokens in the formula expression have been parsed. The single token that is left on the stack is a PtgColorRGB with a value of -256, or 0xFFFFFF00 (yellow). This token is popped off the evaluation stack and returned as the result of the formula evaluation. The formula evaluation is now complete.

4 Security

4.1 Security Considerations for Implementers

Implementers or consumers of this file format need to take into consideration that these files contain sensitive information. Implementers are encouraged to treat these files as sensitive resources and protect them appropriately. These files often contain the following:

- Internal information such as server names, table names, and **list** names can be embedded in the **connection strings** and **query** strings of the data connection elements in the file. However, usernames and passwords are never stored in a VDW file.
- The location of **Office data connection (ODC) files**, which contain additional sensitive information.

Data from external **data sources** are linked to shapes and stored in shape data elements. The data is queried with the credentials of the user making the request and can return sensitive data that is specific to the user. This data is stored in the VDW file when the file is created.

4.2 Index of Security Fields

None.

5 Appendix A: Full XML Schemas

For ease of implementation, this section provides the full XML schemas for the portions of the document described in XML Part Structure.

5.1 app Schema

```
<xsd:schema id="app"
targetNamespace="http://schemas.openxmlformats.org/officeDocument/2006/extended-properties"
attributeFormDefault="qualified" elementFormDefault="qualified">
  <xsd:complexType name="CT_PageMetaData">
    <xsd:attribute name="ID" type="xsd:unsignedLong" use="required" form="unqualified"/>
    <xsd:attribute name="Name" type="xsd:string" use="required" form="unqualified"/>
  </xsd:complexType>
  <xsd:complexType name="CT_PagesMetaData">
    <xsd:sequence>
      <xsd:element name="PageMetaData" minOccurs="1" maxOccurs="unbounded"
type="CT_PageMetaData"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_Properties">
    <xsd:sequence>
      <xsd:element name="Application" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="Company" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="AppVersion" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="DocumentVersion" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="DocumentLanguage" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="DocumentMeasurementSystem" type="xsd:string" minOccurs="1"
maxOccurs="1"/>
      <xsd:element name="DocumentCurrencyID" type="xsd:integer" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="PagesMetaData" type="CT_PagesMetaData" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="Properties" type="CT_Properties"/>
</xsd:schema>
```

5.2 core Schema

```
<xsd:schema id="core"
targetNamespace="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
xmlns="http://schemas.openxmlformats.org/package/2006/metadata/core-properties"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-
msdata" attributeFormDefault="qualified" elementFormDefault="qualified"
>
  <xsd:complexType name="CT_coreProperties">
    <xsd:sequence>
      <xsd:element name="description" msdata:Prefix="dc" type="xsd:string" minOccurs="0"
maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="coreProperties" msdata:Prefix="cp"
type="CT_coreProperties"></xsd:element>
</xsd:schema>
```

5.3 DataBinding Schema

```
<xsd:schema id="DataBinding">
  <xsd:complexType name="CT_PrimaryKeyValue">
    <xsd:attribute name="Value" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="CT_PrimaryKeyValues">
    <xsd:sequence>
```

```
      <xsd:element name="PrimaryKey" minOccurs="1" maxOccurs="unbounded"
type="CT_PrimaryKeyValue"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_Binding">
    <xsd:sequence>
      <xsd:element name="PrimaryKeys" minOccurs="0" maxOccurs="1"
type="CT_PrimaryKeyValues"/>
    </xsd:sequence>
    <xsd:attribute name="BindingID" type="xsd:integer"/>
    <xsd:attribute name="Row" type="xsd:integer" use="optional"/>
  </xsd:complexType>
  <xsd:complexType name="CT_Bindings">
    <xsd:sequence>
      <xsd:element name="Binding" minOccurs="1" maxOccurs="unbounded" type="CT_Binding"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_BindingConnection">
    <xsd:sequence>
      <xsd:element name="Bindings" minOccurs="1" maxOccurs="1" type="CT_Bindings"/>
    </xsd:sequence>
    <xsd:attribute name="RecordsetID" type="xsd:unsignedLong"/>
  </xsd:complexType>
  <xsd:complexType name="CT_BindingConnections">
    <xsd:sequence>
      <xsd:element name="BindingConnection" minOccurs="0" maxOccurs="unbounded"
type="CT_BindingConnection"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="BindingConnections" type="CT_BindingConnections"/>
</xsd:schema>
```

5.4 DataConnection Schema

```
<xsd:schema id="DataConnection">
  <xsd:complexType name="CT DataConnection">
    <xsd:attribute name="ID" type="xsd:unsignedLong" use="required"/>
    <xsd:attribute name="ConnectionString" type="xsd:string" use="optional"/>
    <xsd:attribute name="Filename" type="xsd:string" use="optional"/>
  </xsd:complexType>
  <xsd:complexType name="CT_DataConnections">
    <xsd:sequence>
      <xsd:element name="DataConnection" minOccurs="0" maxOccurs="unbounded"
type="CT DataConnection"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_PrimaryKey">
    <xsd:attribute name="Key" type="xsd:string" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="CT PrimaryKeys">
    <xsd:sequence>
      <xsd:element name="PrimaryKey" minOccurs="1" maxOccurs="unbounded"
type="CT_PrimaryKey"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_DataColumn">
    <xsd:attribute name="Name" type="xsd:string" use="required"/>
    <xsd:attribute name="DisplayName" type="xsd:string" use="required"/>
    <xsd:attribute name="Type" type="xsd:integer" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="CT DataColumns">
    <xsd:sequence>
      <xsd:element name="DataColumn" minOccurs="1" maxOccurs="unbounded"
type="CT_DataColumn"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_DataRecordset">
```

```
    <xsd:sequence>
      <xsd:element name="PrimaryKeys" minOccurs="0" maxOccurs="1" type="CT_PrimaryKeys"/>
      <xsd:element name="DataColumns" minOccurs="1" maxOccurs="1" type="CT_DataColumns"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:unsignedLong" use="required"/>
    <xsd:attribute name="ConnectionID" type="xsd:unsignedLong" use="required"/>
    <xsd:attribute name="Command" type="xsd:string" use="required"/>
    <xsd:attribute name="Pages" type="xsd:string" use="required"/>
    <xsd:attribute name="RowOrder" type="xsd:integer" use="optional"/>
  </xsd:complexType>
  <xsd:complexType name="CT DataRecordsets">
    <xsd:sequence>
      <xsd:element name="DataRecordset" minOccurs="0" maxOccurs="unbounded"
type="CT_DataRecordset"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT Connections">
    <xsd:sequence>
      <xsd:element name="DataConnections" minOccurs="1" maxOccurs="1"
type="CT_DataConnections"/>
      <xsd:element name="DataRecordsets" minOccurs="1" maxOccurs="1"
type="CT_DataRecordsets"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="Connections" type="CT_Connections"/>
</xsd:schema>
```

5.5 DataGraphicDefinition Schema

```
<xsd:schema id="DataGraphicDefinition">
  <xsd:complexType name="CT_IconSetRule">
    <xsd:attribute name="Name" type="xsd:string" use="required"/>
    <xsd:attribute name="Formula" type="xsd:string" use="required"/>
    <xsd:attribute name="Image" type="xsd:string" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="CT IconSetDef">
    <xsd:sequence>
      <xsd:element name="IconSetRule" minOccurs="1" maxOccurs="unbounded"
type="CT_IconSetRule"/>
    </xsd:sequence>
    <xsd:attribute name="DefID" type="xsd:string" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="CT_DataGraphicDefs">
    <xsd:sequence>
      <xsd:element name="IconSetDef" minOccurs="0" maxOccurs="unbounded"
type="CT_IconSetDef"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="DataGraphicDefs" type="CT DataGraphicDefs"/>
</xsd:schema>
```

5.6 DataGraphic Schema

```
<xsd:schema id="DataGraphic">
  <xsd:complexType name="CT_DataGraphicDef">
    <xsd:attribute name="DefID" type="xsd:string" use="required"/>
    <xsd:attribute name="DefShapeID" type="xsd:string" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="CT_DataGraphicDefinitions">
    <xsd:sequence>
      <xsd:element name="DataGraphicDef" minOccurs="1" maxOccurs="unbounded"
type="CT_DataGraphicDef"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_Geometry">
```

```
    <xsd:attribute name="Index" type="xsd:integer" use="required"/>
    <xsd:attribute name="BoundPts" type="xsd:string" use="optional"/>
    <xsd:attribute name="Attribute" type="xsd:string" use="required"/>
    <xsd:attribute name="Formula" type="xsd:string" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="CT Sheet">
    <xsd:sequence>
      <xsd:element name="Geometry" minOccurs="0" maxOccurs="unbounded" type="CT_Geometry"/>
    </xsd:sequence>
    <xsd:attribute name="Name" type="xsd:string"/>
    <xsd:attribute name="Angle" type="xsd:string" use="optional"/>
    <xsd:attribute name="FlipX" type="xsd:string" use="optional"/>
    <xsd:attribute name="FlipY" type="xsd:string" use="optional"/>
    <xsd:attribute name="LocPinX" type="xsd:string" use="optional"/>
    <xsd:attribute name="LocPinY" type="xsd:string" use="optional"/>
    <xsd:attribute name="PinX" type="xsd:string" use="optional"/>
    <xsd:attribute name="PinY" type="xsd:string" use="optional"/>
    <xsd:attribute name="HideText" type="xsd:string" use="optional"/>
    <xsd:attribute name="FillBackground" type="xsd:string" use="optional"/>
    <xsd:attribute name="FillBackgroundTrans" type="xsd:string" use="optional"/>
    <xsd:attribute name="FillForeground" type="xsd:string" use="optional"/>
    <xsd:attribute name="FillForegroundTrans" type="xsd:string" use="optional"/>
    <xsd:attribute name="FillPattern" type="xsd:string" use="optional"/>
    <xsd:attribute name="LineColor" type="xsd:string" use="optional"/>
    <xsd:attribute name="LineColorTrans" type="xsd:string" use="optional"/>
    <xsd:attribute name="LineWeight" type="xsd:string" use="optional"/>
    <xsd:attribute name="TextWidth" type="xsd:string" use="optional"/>
    <xsd:attribute name="TextHeight" type="xsd:string" use="optional"/>
    <xsd:attribute name="TextAngle" type="xsd:string" use="optional"/>
    <xsd:attribute name="TextLocPinX" type="xsd:string" use="optional"/>
    <xsd:attribute name="TextLocPinY" type="xsd:string" use="optional"/>
    <xsd:attribute name="TextPinX" type="xsd:string" use="optional"/>
    <xsd:attribute name="TextPinY" type="xsd:string" use="optional"/>
    <xsd:attribute name="Rounding" type="xsd:string" use="optional"/>
  </xsd:complexType>
  <xsd:complexType name="CT Formulas">
    <xsd:sequence>
      <xsd:element name="Sheet" minOccurs="1" maxOccurs="unbounded" type="CT_Sheet"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_FormulaReferences">
    <xsd:anyAttribute namespace="##local" processContents="skip"/>
  </xsd:complexType>
  <xsd:complexType name="CT_DataGraphics_Shape">
    <xsd:sequence>
      <xsd:element name="DataGraphicDefinitions" minOccurs="0" maxOccurs="1"
type="CT_DataGraphicDefinitions"/>
      <xsd:element name="Formulas" minOccurs="0" maxOccurs="1" type="CT Formulas"/>
      <xsd:element name="FormulaReferences" minOccurs="0" maxOccurs="1"
type="CT_FormulaReferences"/>
    </xsd:sequence>
    <xsd:attribute name="ID" type="xsd:string" use="required"/>
    <xsd:attribute name="ShapeName" type="xsd:string" use="required"/>
    <xsd:attribute name="DataRefs" type="xsd:string" use="optional"/>
  </xsd:complexType>
  <xsd:complexType name="CT_DataGraphics_Page">
    <xsd:sequence>
      <xsd:element name="Shape" minOccurs="0" maxOccurs="unbounded"
type="CT DataGraphics Shape"/>
    </xsd:sequence>
    <xsd:attribute name="PageName" type="xsd:string"/>
    <xsd:attribute name="ViewTransform" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="CT_DataGraphics">
    <xsd:sequence>
      <xsd:element name="Page" minOccurs="1" maxOccurs="unbounded"
type="CT_DataGraphics_Page"/>
    </xsd:sequence>
    <xsd:attribute name="ColorTable" type="xsd:string"/>
```

```
  </xsd:complexType>
  <xsd:element name="DataGraphics" type="CT_DataGraphics"/>
</xsd:schema>
```

5.7 ShapeInfo Schema

```
<xsd:schema id="ShapeInfo">
  <xsd:complexType name="CT_PageInfo">
    <xsd:attribute name="Name" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="CT_Pages">
    <xsd:sequence>
      <xsd:element name="PageInfo" minOccurs="1" maxOccurs="unbounded" type="CT PageInfo"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_ShapeData">
    <xsd:attribute name="Name" type="xsd:string" use="required"/>
    <xsd:attribute name="FormattedValue" type="xsd:string" use="required"/>
    <xsd:attribute name="Value" type="xsd:string" use="required"/>
    <xsd:attribute name="Format" type="xsd:string" use="required"/>
    <xsd:attribute name="Type" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:enumeration value="0"/>
          <xsd:enumeration value="1"/>
          <xsd:enumeration value="2"/>
          <xsd:enumeration value="3"/>
          <xsd:enumeration value="4"/>
          <xsd:enumeration value="5"/>
          <xsd:enumeration value="6"/>
          <xsd:enumeration value="7"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="LangID" type="xsd:integer" use="required"/>
    <xsd:attribute name="UnitLabel" type="xsd:integer" use="required"/>
    <xsd:attribute name="CalendarID" type="xsd:integer" use="optional"/>
    <xsd:attribute name="CurrencyID" type="xsd:integer" use="optional"/>
    <xsd:attribute name="ServerAction" type="xsd:integer" use="optional"/>
    <xsd:attribute name="Unit" type="xsd:integer" use="required"/>
    <xsd:attribute name="DisplayUnit" type="xsd:integer" use="optional"/>
    <xsd:attribute name="BindingID" type="xsd:integer" use="optional"/>
    <xsd:attribute name="HyperlinkID" type="xsd:string" use="optional"/>
  </xsd:complexType>
  <xsd:complexType name="CT_ShapeDataItems">
    <xsd:sequence>
      <xsd:element name="ShapeData" minOccurs="1" maxOccurs="unbounded" type="CT_ShapeData"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT Hyperlink">
    <xsd:attribute name="Name" type="xsd:string" use="required"/>
    <xsd:attribute name="Value" type="xsd:string" use="optional"/>
    <xsd:attribute name="Description" type="xsd:string" use="optional"/>
    <xsd:attribute name="SubAddress" type="xsd:string" use="optional"/>
    <xsd:attribute name="SubAddressShape" type="xsd:string" use="optional"/>
    <xsd:attribute name="Zoom" use="optional">
      <xsd:simpleType>
        <xsd:union>
          <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
              <xsd:minExclusive value="0"/>
            </xsd:restriction>
          </xsd:simpleType>
          <xsd:simpleType>
            <xsd:restriction base="xsd:integer">
              <xsd:enumeration value="-2"/>
              <xsd:enumeration value="-1"/>
```

```
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:union>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="Default" type="xsd:boolean" use="optional"/>
  </xsd:complexType>
  <xsd:complexType name="CT_Hyperlinks">
    <xsd:sequence>
      <xsd:element name="Hyperlink" minOccurs="1" maxOccurs="unbounded" type="CT_Hyperlink"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_ShapeInfo">
    <xsd:sequence>
      <xsd:element name="ShapeDataItems" minOccurs="0" maxOccurs="1"
type="CT_ShapeDataItems"/>
      <xsd:element name="Hyperlinks" minOccurs="0" maxOccurs="1" type="CT Hyperlinks"/>
    </xsd:sequence>
    <xsd:attribute name="Name" type="xsd:string" use="required"/>
    <xsd:attribute name="DisplayName" type="xsd:string"/>
    <xsd:attribute name="Guid" type="xsd:string" use="optional"/>
    <xsd:attribute name="Layout" type="xsd:string" use="required"/>
  </xsd:complexType>
  <xsd:complexType name="CT_Page">
    <xsd:sequence>
      <xsd:element name="Pages" minOccurs="1" maxOccurs="1" type="CT_Pages"/>
      <xsd:element name="ShapeInfo" minOccurs="0" maxOccurs="unbounded" type="CT_ShapeInfo"/>
    </xsd:sequence>
    <xsd:attribute name="Name" type="xsd:string" use="required"/>
    <xsd:attribute name="Zoom" use="required">
      <xsd:simpleType>
        <xsd:union>
          <xsd:simpleType>
            <xsd:restriction base="xsd:double">
              <xsd:enumeration value="-2"/>
              <xsd:enumeration value="-1"/>
            </xsd:restriction>
          </xsd:simpleType>
          <xsd:simpleType>
            <xsd:restriction base="xsd:double">
              <xsd:minInclusive value="0"/>
            </xsd:restriction>
          </xsd:simpleType>
        </xsd:union>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="OffsetX" type="xsd:integer" use="required"/>
    <xsd:attribute name="OffsetY" type="xsd:integer" use="required"/>
    <xsd:attribute name="DefaultUnitsText" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:enumeration value="48"/>
          <xsd:enumeration value="49"/>
          <xsd:enumeration value="50"/>
          <xsd:enumeration value="51"/>
          <xsd:enumeration value="52"/>
          <xsd:enumeration value="53"/>
          <xsd:enumeration value="54"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
    <xsd:attribute name="DefaultUnitsAngle" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:integer">
          <xsd:enumeration value="80"/>
          <xsd:enumeration value="81"/>
          <xsd:enumeration value="82"/>
          <xsd:enumeration value="83"/>
          <xsd:enumeration value="84"/>
```

```
            <xsd:enumeration value="85"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="DefaultUnitsDuration" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:integer">
            <xsd:enumeration value="42"/>
            <xsd:enumeration value="43"/>
            <xsd:enumeration value="44"/>
            <xsd:enumeration value="45"/>
            <xsd:enumeration value="46"/>
            <xsd:enumeration value="47"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
      <xsd:attribute name="DefaultUnitsPage" use="required">
        <xsd:simpleType>
          <xsd:restriction base="xsd:integer">
            <xsd:enumeration value="43"/>
            <xsd:enumeration value="44"/>
            <xsd:enumeration value="45"/>
            <xsd:enumeration value="46"/>
            <xsd:enumeration value="47"/>
            <xsd:enumeration value="50"/>
            <xsd:enumeration value="51"/>
            <xsd:enumeration value="53"/>
            <xsd:enumeration value="54"/>
            <xsd:enumeration value="63"/>
            <xsd:enumeration value="64"/>
            <xsd:enumeration value="65"/>
            <xsd:enumeration value="66"/>
            <xsd:enumeration value="67"/>
            <xsd:enumeration value="68"/>
            <xsd:enumeration value="69"/>
            <xsd:enumeration value="70"/>
            <xsd:enumeration value="71"/>
            <xsd:enumeration value="72"/>
            <xsd:enumeration value="73"/>
            <xsd:enumeration value="74"/>
            <xsd:enumeration value="75"/>
            <xsd:enumeration value="76"/>
          </xsd:restriction>
        </xsd:simpleType>
      </xsd:attribute>
    </xsd:complexType>
    <xsd:element name="Page" type="CT Page"/>
  </xsd:schema>
```

5.8 ShapeTextBinding Schema

```
  <xsd:schema id="ShapeTextBinding">
    <xsd:complexType name="CT_TextRun">
      <xsd:attribute name="Name" type="xsd:string" use="required"/>
      <xsd:attribute name="Formula" type="xsd:string" use="optional"/>
      <xsd:attribute name="Format" type="xsd:string" use="optional"/>
      <xsd:attribute name="Color" type="xsd:string" use="optional"/>
      <xsd:attribute name="DataRefs" type="xsd:string" use="optional"/>
      <xsd:attribute name="DataRefsColor" type="xsd:string" use="optional"/>
    </xsd:complexType>
    <xsd:complexType name="CT ShapeText">
      <xsd:sequence>
        <xsd:element name="TextRun" minOccurs="1" maxOccurs="1" type="CT_TextRun"/>
      </xsd:sequence>
      <xsd:attribute name="Name" type="xsd:string" use="required"/>
    </xsd:complexType>
    <xsd:complexType name="CT_ShapeTextBinding_Page">
```

```
    <xsd:sequence>
      <xsd:element name="ShapeText" minOccurs="0" maxOccurs="unbounded" type="CT_ShapeText"/>
    </xsd:sequence>
    <xsd:attribute name="Name" type="xsd:string"/>
  </xsd:complexType>
  <xsd:element name="Page" type="CT ShapeTextBinding Page"/>
</xsd:schema>
```

5.9 ShapeOutline Schema

```
<xsd:schema id="ShapeOutline">
  <xsd:complexType name="CT_Path">
    <xsd:attribute name="Shape" type="xsd:string" fixed="poly"/>
    <xsd:attribute name="Data" use="required">
      <xsd:simpleType>
        <xsd:restriction base="xsd:string">
          <xsd:pattern value="\d*(,\d*){3,}"/>
        </xsd:restriction>
      </xsd:simpleType>
    </xsd:attribute>
  </xsd:complexType>
  <xsd:complexType name="CT_Shape">
    <xsd:sequence>
      <xsd:element name="Path" minOccurs="1" maxOccurs="unbounded" type="CT_Path"/>
    </xsd:sequence>
    <xsd:attribute name="Name" type="xsd:string" use="required"/>
    <xsd:attribute name="Layout" type="xsd:string"/>
  </xsd:complexType>
  <xsd:complexType name="CT_Shapes">
    <xsd:sequence>
      <xsd:element name="Shape" minOccurs="0" maxOccurs="unbounded" type="CT_Shape"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="CT_ShapeOutline_Page">
    <xsd:sequence>
      <xsd:element name="Shapes" type="CT Shapes"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:element name="Page" type="CT_ShapeOutline_Page"/>
</xsd:schema>
```

6 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

- Microsoft SharePoint Server 2010
- Microsoft Visio 2010
- Microsoft SharePoint Server 2013
- Microsoft Visio 2013
- Microsoft SharePoint Server 2016
- Microsoft Visio 2016
- Microsoft SharePoint Server 2019
- Microsoft Visio 2019
- Microsoft Visio 2021
- Microsoft SharePoint Server Subscription Edition
- Microsoft Visio 2024 Preview

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

7 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
6 Appendix B: Product Behavior	Updated list of supported products.	Major

8 Index

F

R

S

T

V

W

X