[MS-OSHARED]:

Office Common Data Types and Objects Structures

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
6/27/2008	1.0	New	Initial Availability
1/16/2009	1.01	Editorial	Revised and edited the technical content
7/13/2009	1.02	Major	Changes made for template compliance
8/28/2009	1.03	Editorial	Revised and edited the technical content
11/6/2009	1.04	Editorial	Revised and edited the technical content
2/19/2010	2.0	Editorial	Revised and edited the technical content
3/31/2010	2.01	Editorial	Revised and edited the technical content
4/30/2010	2.02	Editorial	Revised and edited the technical content
6/7/2010	2.03	Editorial	Revised and edited the technical content
6/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
7/23/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.04	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	2.04	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	2.04	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	2.5	Minor	Clarified the meaning of the technical content.
4/11/2012	2.5	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	2.6	Minor	Clarified the meaning of the technical content.
10/8/2012	2.6	None	No changes to the meaning, language, or formatting of the technical content.
2/11/2013	2.6	None	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	2.6	None	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	2.6	None	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	2.6	None	No changes to the meaning, language, or formatting of the technical content.

Date	Revision History	Revision Class	Comments
4/30/2014	2.7	Minor	Clarified the meaning of the technical content.
7/31/2014	2.8	Minor	Clarified the meaning of the technical content.
10/30/2014	2.9	Minor	Clarified the meaning of the technical content.
3/16/2015	3.0	Major	Significantly changed the technical content.
9/4/2015	4.0	Major	Significantly changed 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.
12/15/2016	4.1	Minor	Clarified the meaning of the technical content.
6/20/2017	4.1	None	No changes to the meaning, language, or formatting of the technical content.
9/19/2017	4.2	Minor	Clarified the meaning of the technical content.
12/12/2017	4.3	Minor	Clarified the meaning of the technical content.
4/27/2018	5.0	Major	Significantly changed the technical content.
8/28/2018	6.0	Major	Significantly changed the technical content.
12/11/2018	7.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The Office Common Data Types and Objects Structures provide a collection of common data types, objects, and algorithms used by various Office application binary file formats. While the structure of the persistence of the data types and objects is specified in this document, their location in the binary formats is determined by the host application.

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:

absolute path: A string that identifies the location of a file and that begins with a drive identifier and root directory or network share and ends with the complete file name. Examples are C:\Documents\Work\example.txt and \\netshare\Documents\Work\example.txt.

accelerator key: Any combination of keys that are pressed simultaneously to run a command.

ActiveX control: A reusable software control, such as a check box or button, that uses ActiveX technology and provides options to users or runs macros or scripts that automate a task. See also ActiveX object.

adaptive menu: A type of menu that displays the most recently used commands at the top of that menu.

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.

anti-moniker: A Component Object Model (COM) object that is the inverse of a simple **moniker** and has no internal structure; it is the inverse of a COM implementation of a file, an item, or a pointer moniker. An anti-moniker that is composed to the right of a file moniker, item moniker, or pointer moniker composes to nothing.

assembly: A collection of one or more files that is versioned and deployed as a unit. An assembly is the primary building block of a .NET Framework application. All managed types and resources are contained within an assembly and are marked either as accessible only within the assembly or as accessible from code in other assemblies. Assemblies also play a key role in security. The code access security system uses information about an assembly to determine the set of permissions that is granted to code in the assembly.

base URL: A URL that is specified for a web resource to convert all relative URLs in that resource to absolute URLs. A base URL ends with either a file name, such as http://www.example.com/sample.htm, or a slash, such as http://www.example.com/subdir/. See also absolute URL.

basic toolbar: A toolbar that consists of a row, column, or block of buttons, each of which perform an action when activated. Unlike a **menu toolbar**, which displays only text labels, a basic toolbar can display both text and icons.

big-endian: Multiple-byte values that are byte-ordered with the most significant byte stored in the memory location with the lowest address.

ButtonPopup control: A type of Button control that displays a menu of related commands when activated.

cell: A box that is formed by the intersection of a row and a column in a worksheet or a table. A cell can contain numbers, strings, and formulas, and various formats can be applied to that data.

class identifier (CLSID): A **GUID** that identifies a software component; for instance, a DCOM object class or a COM class.

clipboard format: An unsigned integer that uniquely identifies the format of a data packet that is stored in a binary large object (BLOB) and can be shared between processes through the operating system clipboard or other means.

code page: An ordered set of characters of a specific script in which a numerical index (code-point value) is associated with each character. Code pages are a means of providing support for character sets and keyboard layouts used in different countries. Devices such as the display and keyboard can be configured to use a specific code page and to switch from one code page (such as the United States) to another (such as Portugal) at the user's request.

Component Object Model (COM): An object-oriented programming model that defines how objects interact within a single process or between processes. In **COM**, clients have access to an object through interfaces implemented on the object. For more information, see [MS-DCOM].

composite moniker: A Component Object Model (COM) object that joins two or more moniker objects and that can determine the relation between the parts. There are two types of composite monikers: generic, which can connect any two monikers regardless of class; and, nongeneric, which can connect monikers only of the same class.

context menu: A menu that is related to the active window, selection, or object. Also referred to as shortcut menu.

Coordinated Universal Time (UTC): A high-precision atomic time standard that approximately tracks Universal Time (UT). It is the basis for legal, civil time all over the Earth. Time zones around the world are expressed as positive and negative offsets from UTC. In this role, it is also referred to as Zulu time (Z) and Greenwich Mean Time (GMT). In these specifications, all references to UTC refer to the time at UTC-0 (or GMT).

custom toolbar: A type of toolbar that contains a user-defined set of controls and is not included in an application by default. A custom toolbar has a toolbar identifier value of "1".

custom toolbar control: A user-defined control that can be added to a toolbar. A custom toolbar control has a **toolbar control identifier (TCID)** value of "1" and can be one of the following types of controls: ActiveX, Button, ComboBox, DropDown, Edit, or Popup.

cyclic redundancy check (CRC): An algorithm used to produce a checksum (a small, fixed number of bits) against a block of data, such as a packet of network traffic or a block of a computer file. The CRC is a broad class of functions used to detect errors after transmission or

storage. A CRC is designed to catch random errors, as opposed to intentional errors. If errors might be introduced by a motivated and intelligent adversary, a cryptographic hash function should be used instead.

datasheet: A worksheet window that contains the source data for a Microsoft Graph chart object.

digital certificate: See the "digital certificate definition standard," as described in [X509].

digital certificate store: A database that stores a variety of **digital certificates** and information about those certificates, including attributes and constraints.

digital signature: A value that is generated by using a digital signature algorithm, taking as input a private key and an arbitrary-length string, such that a specific verification algorithm is satisfied by the value, the input string, and the public key corresponding to the input private key.

docked: A condition where a toolbar is attached to the **docking area** of an application window.

docked location: A specific position of a toolbar within the **docking area** of an application window.

docking area: An area that is adjacent to the edge of an application window. A toolbar can be moved and attached to a docking area.

document workspace: A document repository that enables users to collaborate on one or more documents.

ExpandingGrid control: A type of **ButtonPopup control** that displays and sets a value from a continuous range of possible values when the user drags across the menu area.

file moniker: A Component Object Model (COM) object that stores the path name that is assigned to a file by the native file system.

Gauge control: A type of control that displays and sets a value from a continuous range of possible values that the user selects by dragging a slider. An example is the Zoom control in an application window.

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).

GraphicCombo control: A type of ComboBox control that can display both text and graphics as a list of options.

GraphicDropDown control: A type of DropDown control that can display custom graphics in a list of options.

Grid control: A type of menu control that can be activated by clicking a cell in a grid. An example is the Font Color control.

GUID_NULL: A **GUID** that has the value "{00000000-0000-0000-0000-000000000000}".

hash: A fixed-size result that is obtained by applying a one-way mathematical function, which is sometimes referred to as a hash algorithm, to an arbitrary amount of data. If the input data changes, the hash also changes. The hash can be used in many operations, including authentication and digital signing.

hyperlink: A relationship between two anchors, as described in [RFC1866].

hyperlink location: A portion of a hyperlink that specifies the location of a specific item, such as a bookmark, within a document, object, or other type of resource; for example "#bookmark" in the hyperlink location C:\Documents\Document.docx#bookmark.

hyperlink target: A portion of a hyperlink that specifies the document, object, or other resource; for example "C:\Documents\Document.docx" in the hyperlink location C:\Documents\Document.docx#bookmark.

item moniker: A Component Object Model (COM) object that identifies an object that is contained in another object, such as an OLE object that is embedded in a document.

Joint Photographic Experts Group (JPEG): A raster graphics file format for displaying high-resolution color graphics. JPEG graphics apply a user-specified compression scheme that can significantly reduce the file sizes of photo-realistic color graphics. A higher level of compression results in lower quality, whereas a lower level of compression results in higher quality. JPEG-format files have a .jpg or .jpeg file name extension.

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.

linked object: An object that is inserted into a document and continues to exist in a separate source file. If the object in the source file changes, the object in the document is updated automatically to reflect those changes.

little-endian: Multiple-byte values that are byte-ordered with the least significant byte stored in the memory location with the lowest address.

locale: A collection of rules and data that are specific to a language and a geographical area. A locale can include information about sorting rules, date and time formatting, numeric and monetary conventions, and character classification.

locale settings: A collection of rules and data that are specific to a language and a geographic area. Locale settings include information about sorting rules, date and time formatting, numeric and monetary conventions, and character classification.

major version: An iteration of a software component, document, or list item that is ready for a larger group to see, or has changed significantly from the previous major version. For an item on a SharePoint site, the **minor version** is always "0" (zero) for a major version.

managed code: Code that is executed by the common language runtime (CLR) environment rather than directly by the operating system. Managed code applications gain CLR services, such as automatic garbage collection, runtime type checking, and security support. These services provide uniform behavior that is independent of platform and language.

manifest: A file that stores metadata about an expansion pack, such as the name of the expansion pack, the files and resources that are included in the expansion pack, and the dependencies that it has on other files and components.

menu toolbar: A type of toolbar that is displayed in an application window, typically at the top, and provides a set of menu controls from which the user can select. Activating a control on the toolbar displays a list of commands in that menu, and the menu remains open until the user closes it or chooses a menu command.

minor version: An iteration of a software component, document, or list item that is in progress or has changed only slightly from the previous version. For an item on a SharePoint site, the minor version number is never "0" (zero) and is incremented for each new version of an item, unless a **major version** is explicitly published. When minor versioning is disabled on a SharePoint site, only major version numbers are incremented, and the minor version is always "0" (zero).

moniker: An object that stores information that uniquely identifies a Component Object Model (COM) object and allows that object to be located and activated.

MS-DOS path compatibility mode: A mode that converts an MS-DOS path into a corresponding path by mapping MS-DOS device names to MS-DOS devices and drive letters.

object: In **COM**, a software entity that implements the IUnknown interface and zero or more additional interfaces that may be obtained from each other using the IUnknown interface. A **COM object** can be exposed to remote clients via the DCOM protocol, in which case it is also a DCOM object.

Object Linking and Embedding (OLE): A technology for transferring and sharing information between applications by inserting a file or part of a file into a compound document. The inserted file can be either embedded or linked. See also embedded object and **linked object**.

OCXDropDown control: A type of DropDown control that displays a list of the ActiveX controls that are available within that application.

Pane control: A type of toolbar control that hosts a window within itself. The hosted window is not constrained by the layout and control type options of a **basic toolbar** or a **menu toolbar**.

parent directory indicator: In a hierarchical filing system, two periods followed by a backslash (..\) that specify a working directory, relative to the current working directory.

Popup control: A built-in or custom control on a menu bar or toolbar that displays a menu of related commands when clicked.

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.

relative path: A path that is implied by the active working directory or is calculated based on a specified directory. If users enter a command that refers to a file and the full path is not entered, the active working directory is the relative path of the referenced file.

smart document: A file that is programmed to assist the user as the user creates or updates the document. Several types of files, such as forms and templates, can also function as smart documents.

smart tag: A feature that adds the ability to recognize and label specific data types, such as people's names, within a document and displays an action button that enables users to perform common tasks for that data type.

smart tag recognizer: An **add-in** that can interpret a specific type of smart tag, such as an address or a financial symbol, in a document and display an action button that enables users to perform common tasks for that data type.

SplitButtonMRUPopup control: A type of **SplitButtonPopup control** whose icon changes to reflect the command that the user most recently selected from the menu that is displayed by that button.

SplitButtonPopup control: A type of Button control that performs an action when clicked, and can also display a menu of related commands when the user clicks a drop-down arrow that appears on the button.

SplitDropDown control: A type of Button control that performs a default action when clicked, and can also expand to display a list of other possible actions when the user clicks a drop-down arrow that appears on the button.

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.

target frame: The name of a frame that is in an HTML-based frames page and displays the destination of a hyperlink.

thumbnail: A miniature version of an image that is typically used to browse multiple images quickly.

time stamp authority: A service acknowledging that a datum existed before a specific time. The service is typically a trusted third party.

timestamp: A condition of a **digital signature** that indicates whether the signature was created with a valid certificate that has expired or was created with a certificate that had expired already. If the certificate expired after the signature was created, the signature can be trusted. If it expired before the signature was created, it cannot be trusted.

toolbar: A row, column, or block of controls that represent tasks or commands within an application. A toolbar can be either a menu toolbar, which provides access to menu commands, or a **basic toolbar**, which contains buttons that provide shortcuts to tasks that are frequently accessed from menus.

toolbar control: An object that appears on a toolbar and enables user interaction or input, typically to initiate an action, display information, or set values.

toolbar control identifier (TCID): An integer that identifies a specific control on a toolbar.

toolbar control separator: A line that separates groups of controls on a user interface surface, such as controls on a toolbar or commands on a menu.

toolbar control type: A set of pre-defined behaviors and user interface elements for an object that can be implemented on a toolbar. Types include: ActiveX; Button; ButtonPopup; ComboBox; DropDown; Edit; ExpandingGrid; Gauge; GraphicCombo; GraphicDropDown; Grid; Label; OCXDropDown; Popup; SplitButtonPopup; SplitButtonMRUPopup; and SplitDropDown.

toolbar delta: A file component that stores a modification that a user made to a built-in toolbar. Stored modifications include adding, changing, or removing a control from a built-in toolbar.

ToolTip: A small pop-up window that provides brief context-sensitive help when users point to an item. Also referred to as ScreenTip.

top-level toolbar: A **basic toolbar** that is not contained by another **toolbar**.

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).

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].

Universal Naming Convention (UNC): A string format that specifies the location of a resource. For more information, see [MS-DTYP] section 2.2.57.

URI fragment: The portion of a **Uniform Resource Identifier (URI)** that allows indirect identification of a secondary resource by reference to a primary resource and additional

identifying information, as described in [RFC3986]. A fragment component is indicated by a number sign (#) and is terminated by the end of the URI.

URI query: The portion of a **Uniform Resource Identifier (URI)** that, in conjunction with the data in the path component, identifies a resource within the scope of a URI's scheme and naming authority, if any, as described in [RFC3986]. A query component is indicated by the first question mark (?) character and is terminated by a number sign (#) or the end of the URI.

URI scheme: The portion of a **Uniform Resource Identifier (URI)** that refers to a specification for assigning identifiers within the URI, as described in [RFC3986].

URL moniker: A Component Object Model (COM) object that stores a **Uniform Resource Locator (URL)** as a string, based on either a full URL or the combination of a **base URL** and a partial URL string.

Visual Basic for Applications (VBA): A macro-based programming language that derives from Microsoft Visual Basic and can be used to customize and extend an application. Unlike Visual Basic, Microsoft Visual Basic for Applications (VBA) code and macros can be run only from within a host application that supports VBA.

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.

XML expansion pack: A collection of files, managed by a manifest.xml file, that adds functionality to a document by specifying custom displays or actions.

XML namespace: A collection of names that is used to identify elements, types, and attributes in XML documents identified in a URI reference [RFC3986]. A combination of XML namespace and local name allows XML documents to use elements, types, and attributes that have the same names but come from different sources. For more information, see [XMLNS-2ED].

XML schema: A description of a type of XML document that is typically expressed in terms of constraints on the structure and content of documents of that type, in addition to the basic syntax constraints that are imposed by XML itself. An XML schema provides a view of a document type at a relatively high level of abstraction.

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.

[ECMA-376] ECMA International, "Office Open XML File Formats", 1st Edition, ECMA-376, December 2006, http://www.ecma-international.org/publications/standards/Ecma-376.htm

[ITUX680-1994] ITU-T, "Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation", ITU-T Recommendation X.680, July 1994, http://www.itu.int/rec/T-REC-X.680-199407-S/en

[MS-CTDOC] Microsoft Corporation, "Word Custom Toolbar Binary File Format".

[MS-CTXLS] Microsoft Corporation, "Excel Custom Toolbar Binary File Format".

[MS-DOC] Microsoft Corporation, "Word (.doc) Binary File Format".

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-EMF] Microsoft Corporation, "Enhanced Metafile Format".

[MS-IPFF] Microsoft Corporation, "InfoPath Form Template Format".

[MS-LCID] Microsoft Corporation, "Windows Language Code Identifier (LCID) Reference".

[MS-LISTSWS] Microsoft Corporation, "Lists Web Service Protocol".

[MS-ODRAW] Microsoft Corporation, "Office Drawing Binary File Format".

[MS-OFFCRYPTO] Microsoft Corporation, "Office Document Cryptography Structure".

[MS-OFORMS] Microsoft Corporation, "Office Forms Binary File Formats".

[MS-OLEPS] Microsoft Corporation, "Object Linking and Embedding (OLE) Property Set Data Structures".

[MS-OVBA] Microsoft Corporation, "Office VBA File Format Structure".

[MS-OXCDATA] Microsoft Corporation, "Data Structures".

[MS-OXPROPS] Microsoft Corporation, "Exchange Server Protocols Master Property List".

[MS-WMF] Microsoft Corporation, "Windows Metafile Format".

[MS-WSSCAML] Microsoft Corporation, "Collaborative Application Markup Language (CAML) Structure".

[MS-XLS] Microsoft Corporation, "Excel Binary File Format (.xls) Structure".

[PKCS7] RSA Laboratories, "PKCS #7: Cryptographic Message Syntax Standard", PKCS #7, version 1.5, November 1993, http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-7-cryptographic-message-syntax-standar.htm

[PKCS9] RSA Laboratories, "PKCS #9: Selected Attribute Types", PKCS #9, version 1.1, November 1993, http://www.emc.com/emc-plus/rsa-labs/standards-initiatives/pkcs-9-selected-attribute-types.htm

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, http://www.ietf.org/rfc/rfc1321.txt

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

[RFC3280] Housley, R., Polk, W., Ford, W., and Solo, D., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002, http://www.ietf.org/rfc/rfc3280.txt

[RFC3986] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005, http://www.rfc-editor.org/rfc/rfc3986.txt

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.rfc-editor.org/rfc/rfc5234.txt

[W3C-XSD] World Wide Web Consortium, "XML Schema Part 2: Datatypes Second Edition", 28 October 2004, http://www.w3.org/TR/2004/REC-xmlschema-2-20041028

1.2.2 Informative References

[MS-PPT] Microsoft Corporation, "PowerPoint (.ppt) Binary File Format".

[MSDN-AVSTOS] Microsoft Corporation, "Architecture of Visual Studio Tools for Office Solutions", http://msdn.microsoft.com/en-us/library/zkhw8h59(VS.80).aspx

[MSDN-CreateUri] Microsoft Corporation, "CreateUri Function", http://msdn.microsoft.com/en-us/library/ms775098(VS.85).aspx

[MSDN-FM] Microsoft Corporation, "File Monikers", http://msdn.microsoft.com/en-us/library/ms688670(VS.85).aspx

[MSDN-IMAMI] Microsoft Corporation, "IMoniker -- Anti-Moniker Implementation", http://msdn.microsoft.com/en-us/library/ms682346(VS.85).aspx

[MSDN-IMCOM] Microsoft Corporation, "Item Monikers (COM)", http://msdn.microsoft.com/en-us/library/ms693722(VS.85).aspx

[MSDN-IMGCMI] Microsoft Corporation, "IMoniker -- Generic Composite Moniker Implementation", http://msdn.microsoft.com/en-us/library/ms688541(VS.85).aspx

[MSDN-SDO] Microsoft Corporation, "Smart Documents Overview", Office 2003 Smart Document Software Development Kit, http://msdn.microsoft.com/en-us/library/aa193956(office.11).aspx

[MSDN-URLM] Microsoft Corporation, "URL Monikers", http://msdn.microsoft.com/en-us/library/ms688580(VS.85).aspx

1.3 Structure Overview (Synopsis)

The following sections provide an overview of the structure of this protocol.

1.3.1 Toolbar Customization

Toolbar customization structures provide a run-time solution for the creation and deployment of toolbar customizations that are specific to the content of a file.

The following diagram illustrates how toolbar customization structures are presented within the XCB binary **stream**, as described in [MS-XLS] section 2.1.7.10 with its toolbar customization structures.

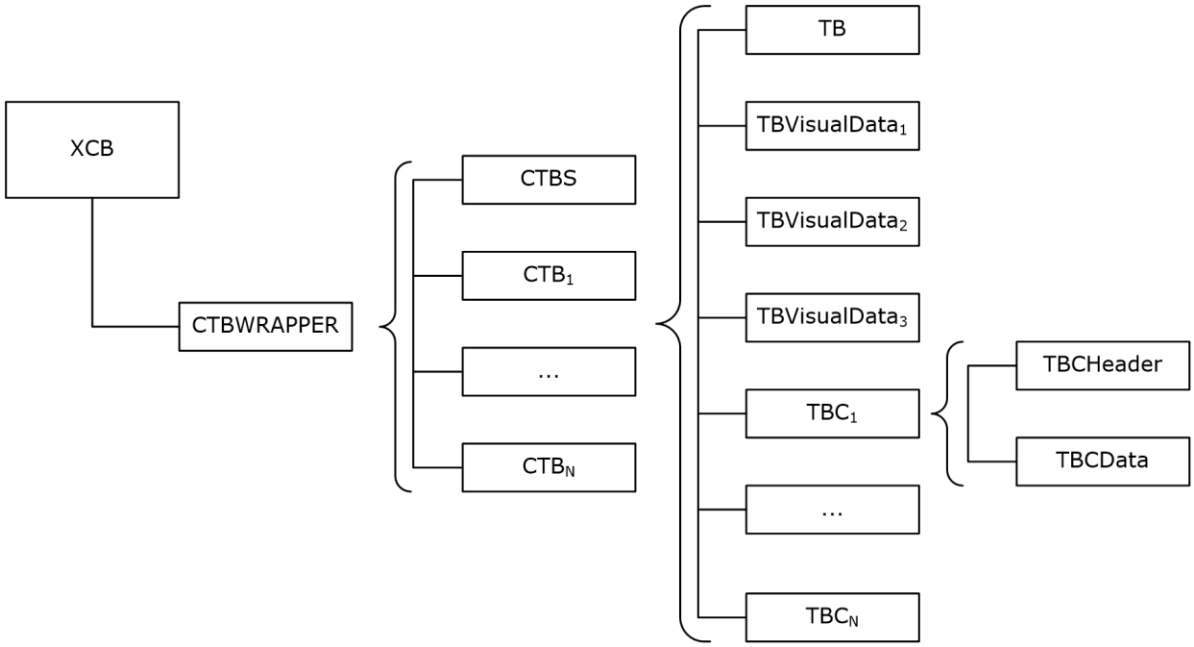


Figure 1: Toolbar customization structures in an XCB binary stream

The following diagram illustrates how toolbar customization structures are presented within the Table Stream, as described in [MS-DOC] section 2.1.2.

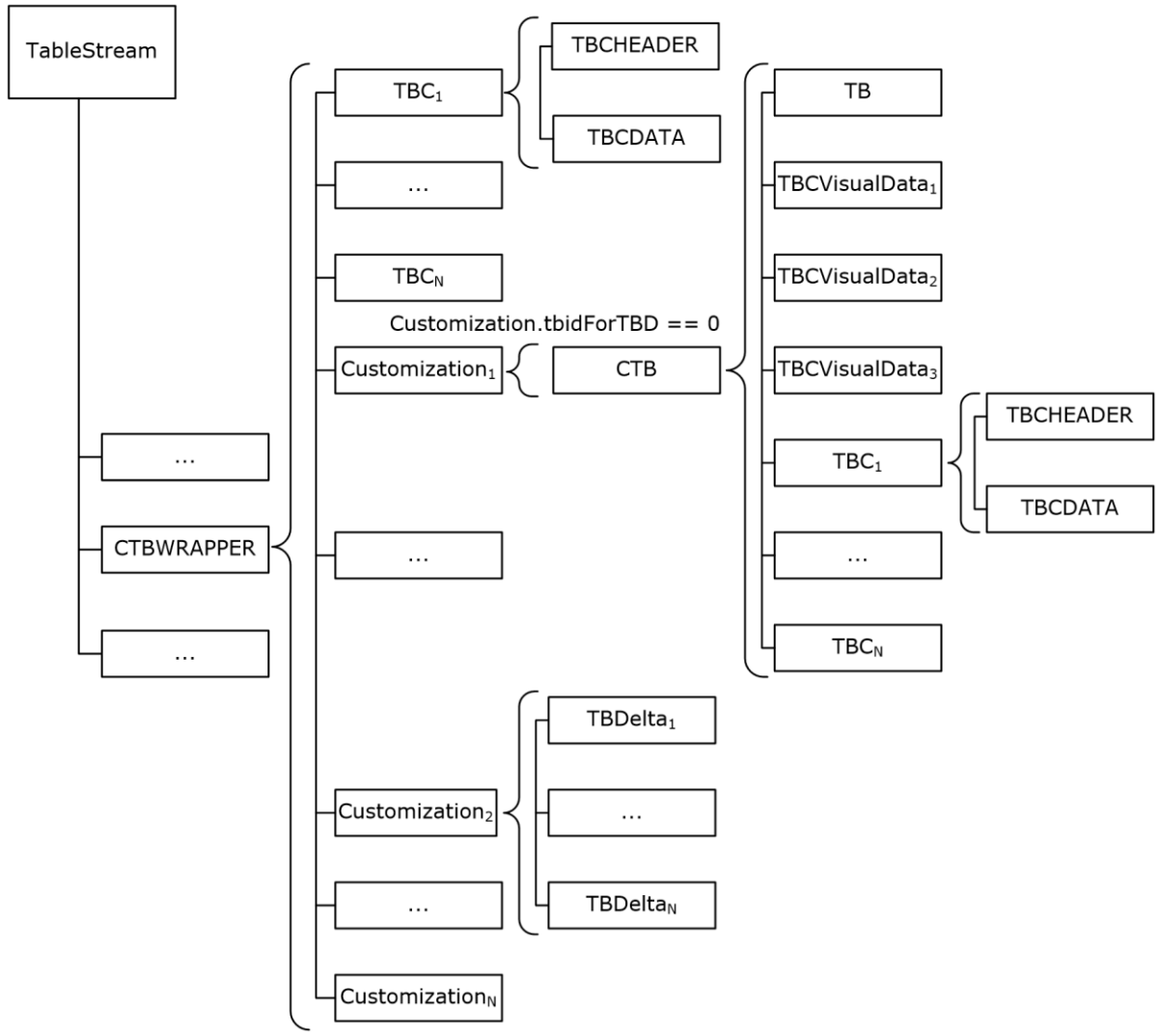


Figure 2: Toolbar customization structures in a Table Stream

1.3.2 Property Set Storage

OLE Property Sets (section 2.3.3.2) define a common way to store metadata about a file. The **Summary Information Property Set** (section 2.3.3.2.1) and **Document Summary Information Property Set** (section 2.3.3.2.2) contain a predefined set of properties, storing information such as title, subject, keywords, author, and line count. The User Defined Property Set (section 2.3.3.2.3) contains a set of properties that an application or user can extend.

The following two diagrams illustrate the structure of these property sets as persisted to a binary file. These property sets are implementations of **OLE** Property Sets. For a structural overview of OLE Property Sets, see [MS-OLEPS] section 1.3.

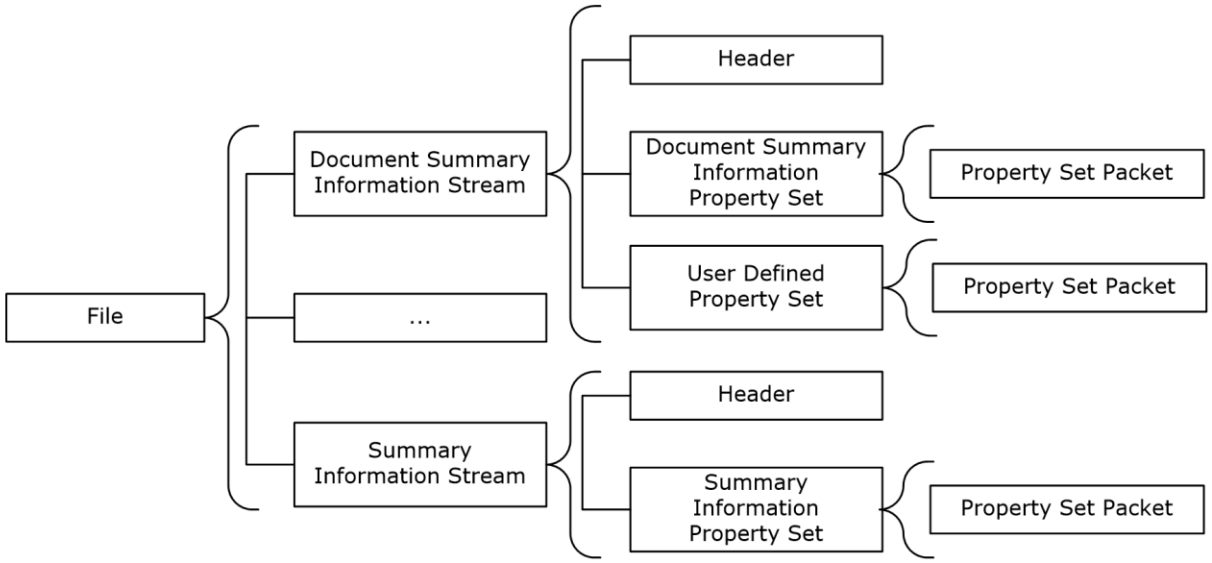


Figure 3: Property set storage structure

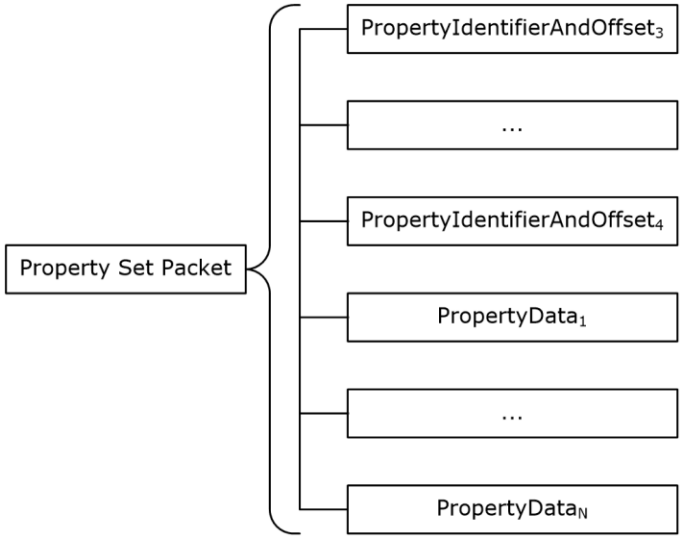


Figure 4: Property set packet structure

1.3.3 RefEdit Control

A **RefEdit** control (section 2.3.5) displays a **range** value that references **cells** in a **datasheet**. A user can enter the value into the control directly or click a drop-down box to select it. The format of the range value depends on the datasheet implementation.

1.3.4 Visual Basic for Applications Digital Signature

The **Visual Basic for Applications (VBA)** **digital signature structure** (section 2.3.2) is contained in a wrapping structure that is specific to the mechanism the file format uses to store the signature.

In the formats described in [MS-XLS] and [MS-PPT], the signature structure is stored in the **Document Summary Information OLE property set** (section 2.3.3.2.2.1) as a VT_BLOB

TypedPropertyValue property. The following diagram illustrates the relative structures for this storage mechanism.

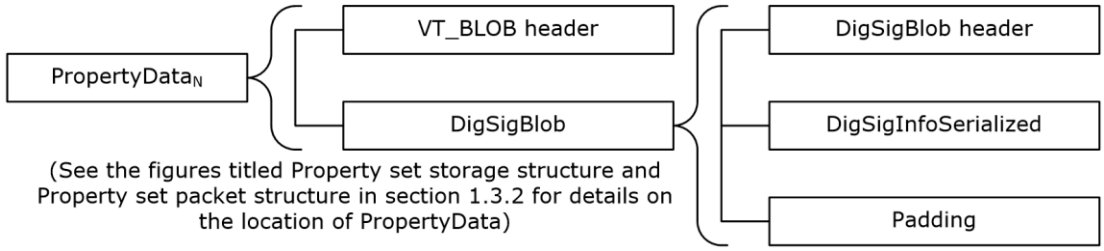


Figure 5: VBA digital signature storage as OLE property set property

In the format described in [MS-DOC], the signature structure is stored in the StwUser string table ([MS-DOC] section 2.9.298) as the value of **Unicode** string variables named "Sign" or "SigAgile". The following diagram illustrates the relative structures for this storage mechanism.

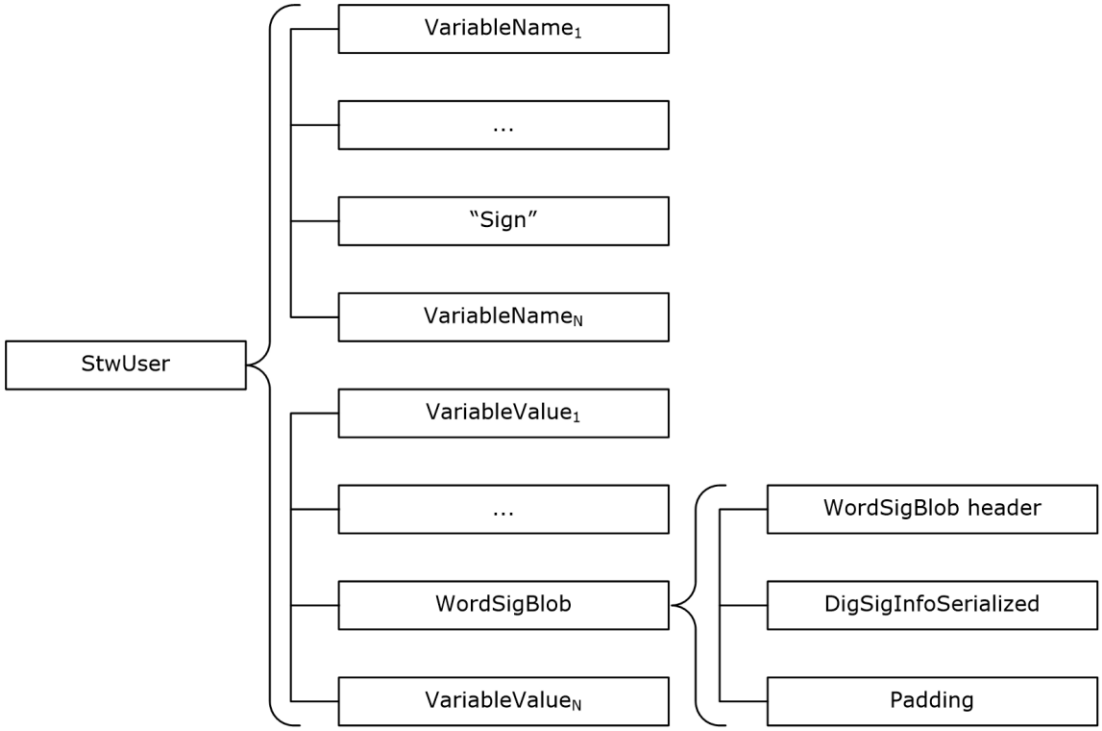


Figure 6: VBA digital signature storage as [MS-DOC] string table variable

The signature structure consists of information about the **digital signature**, as well as the signature (section 2.3.2.4) itself, and a serialized certificate store (section 2.3.2.5.5). The following diagram illustrates this structure.

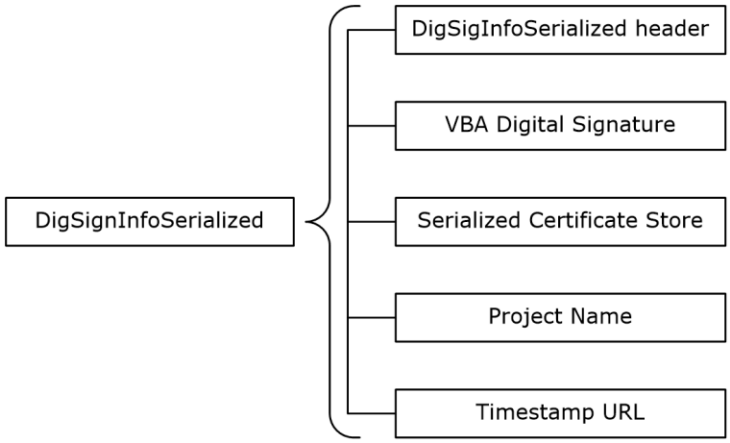


Figure 7: DigSigInfoSerialized structure

1.3.5 Byte Ordering

Data in the Office Common Data Types and Objects Structure is stored in **little-endian** format.

Some computer architectures number bytes in a binary word from left to right, which is referred to as **big-endian**. The packet diagrams for this documentation are big-endian. Other architectures number the bytes in a binary word from right to left, which is referred to as little-endian. The underlying file format enumerations, objects, and records are little-endian.

Using big-endian and little-endian methods, the number 0x12345678 would be stored as shown in the following table.

Byte order	Byte 0	Byte 1	Byte 2	Byte 3
Big-endian	0x12	0x34	0x56	0x78
Little-endian	0x78	0x56	0x34	0x12

1.4 Relationship to Protocols and Other Structures

1.4.1 Toolbar Customization

The **CTB** and **TBC** structures described in [MS-XLS] section 2.6, and the **CTB** and **TBC** structures described in [MS-DOC] section 2.9 depend on the toolbar customization structures specified in this document.
The **TBCHeader** (section 2.3.1.10), **TBCSFlags** (section 2.3.1.12), and **TBCMenuSpecific** (section 2.3.1.21) structures specified in this document are dependent on the toolbar customization data described in the following references:

- [MS-CTDOC] for the persistence format for word processing documents.
- [MS-CTXLS] for the persistence format for spreadsheet **toolbar** customization structures defined in this document.

1.4.2 Property Set Storage

The **OLE Property Sets** (section 2.3.3.2) specified in this document are implementations of **Object Linking and Embedding (OLE)** Property Set Data Structures (OLEPS) as described in [MS-OLEPS]. This document describes where [MS-DOC], [MS-PPT], and [MS-XLS] file formats extend [MS-OLEPS] and where they deviate from [MS-OLEPS].

These property sets are contained in persistence formats described in [MS-XLS], [MS-DOC], and [MS-PPT]. Some data in the property sets is generated according to algorithms specified in [MS-ODRAW], [MS-DOC], and [MS-XLS].

1.4.3 Visual Basic for Applications Digital Signature

The **digital signature** for a **VBA** project is a specific implementation of the PKCS #7 Cryptographic Message Syntax Standard as described in [PKCS7]. Additional implementation information for attributes of the signature is described in [PKCS9]. The file formats [MS-XLS] and [MS-PPT] store their VBA digital signature in their **OLE Property Sets** (section 2.3.3.2). The file format [MS-DOC] stores the VBA digital signature in a string table within the document. The data that the VBA digital signature encrypts is described in the [MS-OVBA] specification.

1.5 Applicability Statement

The structures in this document are intended to function as persistence formats for portions of a document, and are not intended for stand-alone use. These persistence formats provide interoperability with applications that create or read documents conforming to this structure<1>.

1.5.1 Toolbar Customization

Toolbar customization structures represent a common model for storing **custom toolbars** or toolbar alterations within the [MS-DOC] and [MS-XLS] file formats. Toolbar customization structures can be used whenever it is desirable to have **custom toolbar controls** (for example, custom toolbar controls that run custom macros) travel with a file from one system to another.

1.5.2 Property Set Storage

OLE Property Sets (section 2.3.3.2) represent a common way to store metadata about a file within the [MS-DOC], [MS-PPT], and [MS-XLS] file formats in a way that is discoverable to other software.

1.5.3 Visual Basic for Applications Digital Signature

The **digital signature** for a **VBA** project represents a way to securely identify the author of the executable VBA code that is associated with the document by means of a public **digital certificate**. By securely identifying the author, a future user of the document can make a decision to trust or reject the executable document content based on the signer's identity. A digital signature also allows a future user to determine whether the content of the document's executable code is no longer the same as the code that was originally signed, which means that the code can no longer be trusted to have come from the signer identified by the accompanying digital certificate.

1.6 Versioning and Localization

1.6.1 Toolbar Customization

This document covers versioning issues in the following areas:

- **Structure Versions:** There is only one version of the **toolbar** customization structures (section 2.3.1).
- **Localization:** The toolbar customization structures define no **locale**-specific processes or data.

1.6.2 Property Set Storage

This document covers versioning issues in the following areas:

- **Structure Versions:** There is only one version of the property sets defined in **Property Set Storage** (section 2.3.3).
- **Localization:** The encoding of strings contained in the **Lpstr** (section 2.3.3.1.4) data type and the **UnalignedLpstr** (section 2.3.3.1.5) data type depend on the property set's **CodePage** property (described in [MS-OLEPS] section 2.18.2).

1.6.3 Visual Basic for Applications Digital Signature

This document covers versioning issues in the following areas:

- **Structure Versions:** There is only one version of the **VBA digital signature** defined in Visual Basic for Applications Digital Signature Storage (section 2.3.2).
- **Localization:** The VBA digital signature structures define no **locale**-specific processes or data.

1.7 Vendor-Extensible Fields

The following sections describe the vendor-extensible fields used by this protocol.

1.7.1 Toolbar Customization

None.

1.7.2 Property Set Storage

None.

1.7.3 Visual Basic for Applications Digital Signature

Extensibility is as described in the [PKCS7] specification, except the constraints in, VBA Digital Signature (section 2.3.2.4).

2 Structures

2.1 Common ABNF Definitions

The following are common ABNF ([RFC5234]) definitions for the standard ASCII and **Unicode** characters.

```
ASCII-SPACE = %x20
ASCII-EXCLAMATION-MARK = %x21
ASCII-QUOTATION-MARK = %x22
ASCII-NUMBER-SIGN = %x23
ASCII-DOLLAR-SIGN = %x24
ASCII-PERCENT-SIGN = %x25
ASCII-AMPERSAND = %x26
ASCII-APOSTROPHE = %x27
ASCII-LEFT-PARENTHESIS = %x28
ASCII-RIGHT-PARENTHESIS = %x29
ASCII-ASTERISK = %x2A
ASCII-PLUS-SIGN = %x2B
ASCII-COMMA = %x2C
ASCII-HYPHEN-MINUS = %x2D
ASCII-FULL-STOP = %x2E
ASCII-SOLIDUS = %x2F
ASCII-DIGIT-ZERO = %x30
ASCII-DIGIT-ONE = %x31
ASCII-DIGIT-TWO = %x32
ASCII-DIGIT-THREE = %x33
ASCII-DIGIT-FOUR = %x34
ASCII-DIGIT-FIVE = %x35
ASCII-DIGIT-SIX = %x36
ASCII-DIGIT-SEVEN = %x37
ASCII-DIGIT-EIGHT = %x38
ASCII-DIGIT-NINE = %x39
ASCII-COLON = %x3A
ASCII-SEMICOLON = %x3B
ASCII-LESS-THAN-SIGN = %x3C
ASCII-EQUALS-SIGN = %x3D
ASCII-GREATER-THAN-SIGN = %x3E
ASCII-QUESTION-MARK = %x3F
ASCII-COMMERCIAL-AT = %x40
ASCII-CAPITAL-LETTER-A = %x41
ASCII-CAPITAL-LETTER-B = %x42
ASCII-CAPITAL-LETTER-C = %x43
ASCII-CAPITAL-LETTER-D = %x44
ASCII-CAPITAL-LETTER-E = %x45
ASCII-CAPITAL-LETTER-F = %x46
ASCII-CAPITAL-LETTER-G = %x47
ASCII-CAPITAL-LETTER-H = %x48
ASCII-CAPITAL-LETTER-I = %x49
ASCII-CAPITAL-LETTER-J = %x4A
ASCII-CAPITAL-LETTER-K = %x4B
ASCII-CAPITAL-LETTER-L = %x4C
ASCII-CAPITAL-LETTER-M = %x4D
ASCII-CAPITAL-LETTER-N = %x4E
ASCII-CAPITAL-LETTER-O = %x4F
ASCII-CAPITAL-LETTER-P = %x50
ASCII-CAPITAL-LETTER-Q = %x51
ASCII-CAPITAL-LETTER-R = %x52
ASCII-CAPITAL-LETTER-S = %x53
ASCII-CAPITAL-LETTER-T = %x54
ASCII-CAPITAL-LETTER-U = %x55
ASCII-CAPITAL-LETTER-V = %x56
ASCII-CAPITAL-LETTER-W = %x57
ASCII-CAPITAL-LETTER-X = %x58
ASCII-CAPITAL-LETTER-Y = %x59
ASCII-CAPITAL-LETTER-Z = %x5A
```

```
ASCII-LEFT-SQUARE-BRACKET = %x5B
ASCII-REVERSE-SOLIDUS = %x5C
ASCII-RIGHT-SQUARE-BRACKET = %x5D
ASCII-CIRCUMFLEX-ACCENT = %x5E
ASCII-LOW-LINE = %x5F
ASCII-GRAVE-ACCENT = %x60
ASCII-SMALL-LETTER-A = %x61
ASCII-SMALL-LETTER-B = %x62
ASCII-SMALL-LETTER-C = %x63
ASCII-SMALL-LETTER-D = %x64
ASCII-SMALL-LETTER-E = %x65
ASCII-SMALL-LETTER-F = %x66
ASCII-SMALL-LETTER-G = %x67
ASCII-SMALL-LETTER-H = %x68
ASCII-SMALL-LETTER-I = %x69
ASCII-SMALL-LETTER-J = %x6A
ASCII-SMALL-LETTER-K = %x6B
ASCII-SMALL-LETTER-L = %x6C
ASCII-SMALL-LETTER-M = %x6D
ASCII-SMALL-LETTER-N = %x6E
ASCII-SMALL-LETTER-O = %x6F
ASCII-SMALL-LETTER-P = %x70
ASCII-SMALL-LETTER-Q = %x71
ASCII-SMALL-LETTER-R = %x72
ASCII-SMALL-LETTER-S = %x73
ASCII-SMALL-LETTER-T = %x74
ASCII-SMALL-LETTER-U = %x75
ASCII-SMALL-LETTER-V = %x76
ASCII-SMALL-LETTER-W = %x77
ASCII-SMALL-LETTER-X = %x78
ASCII-SMALL-LETTER-Y = %x79
ASCII-SMALL-LETTER-Z = %x7A
ASCII-LEFT-CURLY-BRACKET = %x7B
ASCII-VERTICAL-LINE = %x7C
ASCII-RIGHT-CURLY-BRACKET = %x7D
ASCII-TILDE = %x7E
ASCII-DELETE = %x7F
ASCII-CRLF = %x0d.0a

ASCII-DIGIT = ASCII-DIGIT-ZERO /
              ASCII-DIGIT-ONE /
              ASCII-DIGIT-TWO /
              ASCII-DIGIT-THREE /
              ASCII-DIGIT-FOUR /
              ASCII-DIGIT-FIVE /
              ASCII-DIGIT-SIX /
              ASCII-DIGIT-SEVEN /
              ASCII-DIGIT-EIGHT /
              ASCII-DIGIT-NINE

ASCII-DIGIT-HEXADECIMAL = ASCII-DIGIT /
                          ASCII-SMALL-LETTER-A /
                          ASCII-SMALL-LETTER-B /
                          ASCII-SMALL-LETTER-C /
                          ASCII-SMALL-LETTER-D /
                          ASCII-SMALL-LETTER-E /
                          ASCII-SMALL-LETTER-F /
                          ASCII-CAPITAL-LETTER-A /
                          ASCII-CAPITAL-LETTER-B /
                          ASCII-CAPITAL-LETTER-C /
                          ASCII-CAPITAL-LETTER-D /
                          ASCII-CAPITAL-LETTER-E /
                          ASCII-CAPITAL-LETTER-F

ASCII-ALL = %x20-7F

UTF16-CHAR-NON-ASCII = %x0080-FFFF
```

```
UTF16-ANY = ASCII-ALL / UTF16-CHAR-NON-ASCII
```

2.2 Data Types

2.2.1 Common Data Types

2.2.1.1 ObjectUpdateEnum

Specifies how the container updates the **linked object** embedded in an application.

Name	Value	Meaning
OU_Always	0x00000001	The container updates the linked object whenever there is an update associated with it.
OU_OnCall	0x00000003	The container updates the linked object only when the container calls its update method.

2.2.1.2 DataViewAspectEnum

Specifies the desired data or view aspect of the **object** when drawing or obtaining data.

Name	Value	Meaning
OR_Content	0x00000001	Specifies that the object is displayed as an embedded object inside of a container.
OR_Thumbnail	0x00000002	Specifies that the object is displayed as a **thumbnail** image.
OR_Icon	0x00000004	Specifies that the object is displayed as an icon.
OR_DocPrint	0x00000008	Specifies that the object is displayed on the screen as though it were printed to a printer.

2.2.1.3 MSONFC

This specifies the list of numbering formats that can be used for a group of automatically numbered objects. The numbering format values are mapped to the ST_NumberFormat ([ECMA-376] partition IV section 2.18.66) enumeration equivalents as described in the following table.

Name	Value	Meaning
msonfcArabic	0x00	decimal
msonfcUCRoman	0x01	upperRoman
msonfcLCRoman	0x02	lowerRoman
msonfcUCLetter	0x03	upperLetter
msonfcLCLetter	0x04	lowerLetter
msonfcOrdinal	0x05	ordinal
msonfcCardtext	0x06	cardinalText

Name	Value	Meaning
msonfcOrdtext	0x07	ordinalText
msonfcHex	0x08	hex
msonfcChiManSty	0x09	chicago
msonfcDbNum1	0x0A	ideographDigital
msonfcDbNum2	0x0B	japaneseCounting
msonfcAiueo	0x0C	Aiueo
msonfcIroha	0x0D	Iroha
msonfcDbChar	0x0E	decimalFullWidth
msonfcSbChar	0x0F	decimalHalfWidth
msonfcDbNum3	0x10	japaneseLegal
msonfcDbNum4	0x11	japaneseDigitalTenThousand
msonfcCirclenum	0x12	decimalEnclosedCircle
msonfcDArabic	0x13	decimalFullWidth2
msonfcDAiueo	0x14	aiueoFullWidth
msonfcDIroha	0x15	irohaFullWidth
msonfcArabicLZ	0x16	decimalZero
msonfcBullet	0x17	bullet
msonfcGanada	0x18	ganada
msonfcChosung	0x19	chosung
msonfcGB1	0x1A	decimalEnclosedFullstop
msonfcGB2	0x1B	decimalEnclosedParen
msonfcGB3	0x1C	decimalEnclosedCircleChinese
msonfcGB4	0x1D	ideographEnclosedCircle
msonfcZodiac1	0x1E	ideographTraditional
msonfcZodiac2	0x1F	ideographZodiac
msonfcZodiac3	0x20	ideographZodiacTraditional
msonfcTpeDbNum1	0x21	taiwaneseCounting
msonfcTpeDbNum2	0x22	ideographLegalTraditional
msonfcTpeDbNum3	0x23	taiwaneseCountingThousand
msonfcTpeDbNum4	0x24	taiwaneseDigital
msonfcChnDbNum1	0x25	chineseCounting
msonfcChnDbNum2	0x26	chineseLegalSimplified

Name	Value	Meaning
msonfcChnDbNum3	0x27	chineseCountingThousand
msonfcChnDbNum4	0x28	decimal
msonfcKorDbNum1	0x29	koreanDigital
msonfcKorDbNum2	0x2A	koreanCounting
msonfcKorDbNum3	0x2B	koreanLegal
msonfcKorDbNum4	0x2C	koreanDigital2
msonfcHebrew1	0x2D	hebrew1
msonfcArabic1	0x2E	arabicAlpha
msonfcHebrew2	0x2F	hebrew2
msonfcArabic2	0x30	arabicAbjad
msonfcHindi1	0x31	hindiVowels
msonfcHindi2	0x32	hindiConsonants
msonfcHindi3	0x33	hindiNumbers
msonfcHindi4	0x34	hindiCounting
msonfcThai1	0x35	thaiLetters
msonfcThai2	0x36	thaiNumbers
msonfcThai3	0x37	thaiCounting
msonfcViet1	0x38	vietnameseCounting
msonfcNumInDash	0x39	numberInDash
msonfcLCRus	0x3A	russianLower
msonfcUCRus	0x3B	russianUpper
msonfcNone	0xFF	Specifies that the sequence will not display any numbering

2.2.1.4 WebScreenSizeEnum

An enumeration that specifies the screen resolution for the target monitor on which the Web page is to be displayed. Values are described in the following table.

Name	Value	Meaning
MSOWOPTScreenSize544x376	0x00	544 by 376 pixels
MSOWOPTScreenSizeWebTV	0x00	544 by 376 pixels
MSOWOPTScreenSize640x480	0x01	640 by 480 pixels
MSOWOPTScreenSize720x512	0x02	720 by 512 pixels
MSOWOPTScreenSize800x600	0x03	800 by 600 pixels

Name	Value	Meaning
MSOWOPTScreenSize1024x768	0x04	1024 by 768 pixels
MSOWOPTScreenSize1152x882	0x05	1152 by 882 pixels
MSOWOPTScreenSize1152x900	0x06	1152 by 900 pixels
MSOWOPTScreenSize1280x1024	0x07	1280 by 1024 pixels
MSOWOPTScreenSize1600x1200	0x08	1600 by 1200 pixels
MSOWOPTScreenSize1800x1440	0x09	1800 by 1440 pixels
MSOWOPTScreenSize1920x1200	0x0A	1920 by 1200 pixels

2.2.1.5 FilePointer

Specifies the offset into the **stream** or file that is to be read from or written to.

offset (4 bytes): Unsigned integer that specifies the offset into a stream or file.

2.2.1.6 FixedPoint

Specifies an approximation of a real number, where the approximation has a fixed number of digits after the radix point.

```
Value of the real number = Integral + (Fractional / 65536.0)
```

Integral (2 bytes): A signed integer that specifies the integral part of the real number.

Fractional (2 bytes): An unsigned integer that specifies the fractional part of the real number.

2.3 Common Objects

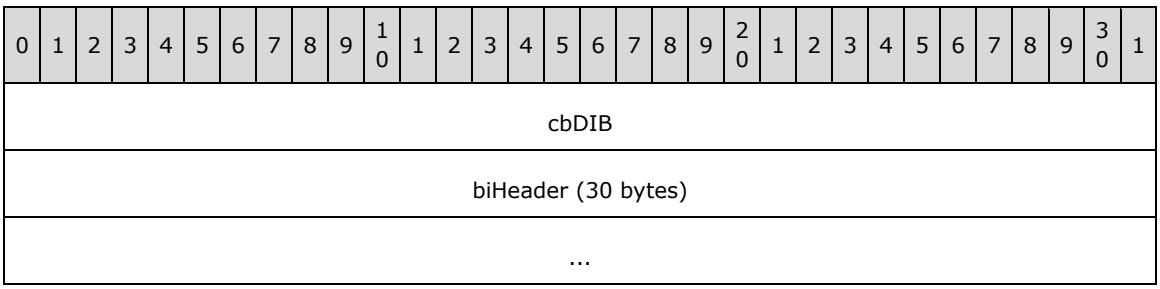
2.3.1 Toolbar Customization

This section specifies structures used by the **CTB**, and **TBC** structures specified in [MS-XLS] section 2.6, and by the **CTB**, and **TBC** structures specified in [MS-DOC] section 2.9.

2.3.1.1 TBCBitmap

Referenced by: TBCBSpecific

Specifies a bitmap used to store a custom icon used by a **toolbar control**.



...	
...	colors (variable)
...	
bitmapData (variable)	
...	

cbDIB (4 bytes): Signed integer that specifies the count of total bytes, excluding this field, in the **TBCBitmap** structure plus 10. The value is given by the following formula:

```
cbDIB = sizeOf(biHeader) + sizeOf(colors) + sizeOf(bitmapData) + 10
```

The value MUST be greater than or equal to 40, and MUST be less than or equal to 65576 (which is a bitmap that is 128 pixels high, 128 pixels wide at 32 bits of color per pixel plus 40 ((128 * 128 * 32 / 8) + 40)).

biHeader (30 bytes): A **BITMAPINFOHEADER** structure (section 2.3.1.2) that contains information about this bitmap.

colors (variable): Zero-based array of **RGBQuad** structures (section 2.3.1.3). MUST exist only if **biHeader.biBitCount** is less than or equal to 0x08. The number of elements in this array MUST be equal to `2^biHeader.biBitCount`.

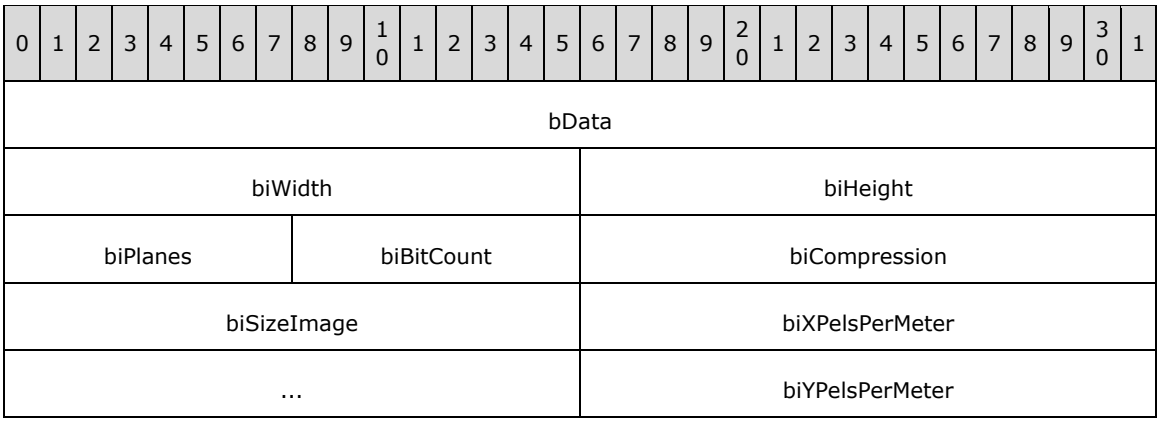
bitmapData (variable): An array of bytes as specified by the **aData** field of a **DeviceIndependentBitmap** object specified in [MS-WMF] section 2.2.2.9. The number of bytes in this array MUST be equal to `cbDIB – sizeOf(colors) – sizeOf(biHeader) –10`. The number of bytes in this array is also calculated with the following formula:

```
Number of bytes in bitmapData = ((biHeader.biWidth * biHeader.biBitCount + 31) & ~31) / 8 *
biHeader.biHeight
```

2.3.1.2 BITMAPINFOHEADER

Referenced by: TBCBitmap

Bitmap header. Contains information about a bitmap.



...	biClrUsed
...	biClrImportant
...	

bData (4 bytes): Unsigned integer. MUST be 0x00000028.

biWidth (2 bytes): Signed integer that specifies the width of the bitmap in pixels. MUST be positive and less than or equal to 0x0080.

biHeight (2 bytes): Signed integer that specifies the height of the bitmap in pixels. MUST be positive and less than or equal to 0x0080.

biPlanes (1 byte): Unsigned integer that MUST be 0x01.

biBitCount (1 byte): Unsigned integer that specifies the number of bits per pixel. MUST be equal to one of the values in the following table.

Value of biBitCount
0x01
0x04
0x08
0x10
0x18
0x20

biCompression (2 bytes): Unsigned integer that specifies the bitmap compression format. A value of 0x0000 means that the bitmap is uncompressed. The value MUST be 0x0000.

biSizeImage (2 bytes): Unsigned integer that specifies the size, in bytes, of the image. The value SHOULD specify the size of the **bitmapData** array of the **TBCBitmap** structure (section 2.3.1.1) that contains this structure and is given by the following formula. For uncompressed bitmaps, this value is equal to 0x0000 and does not specify the size of the image.

```
biSizeImage = ((biHeader.biWidth * biHeader.biBitCount + 31) & ~31) / 8 * biHeader.biHeight
```

biXPelsPerMeter (4 bytes): MUST be 0x00000000 and MUST be ignored.

biYPelsPerMeter (4 bytes): MUST be 0x00000000 and MUST be ignored.

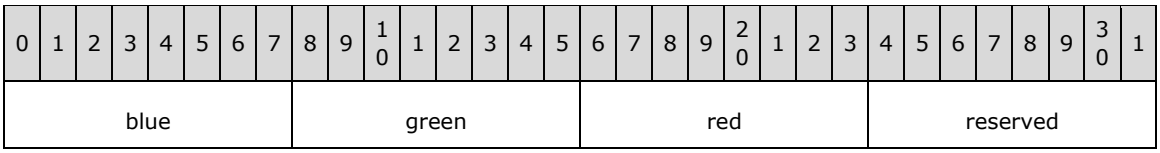
biClrUsed (4 bytes): Unsigned integer as specified by the **ColorUsed** field of the **BitmapInfoHeader** object specified in [MS-WMF] section 2.2.2.3. MUST be 0x00000000.

biClrImportant (4 bytes): Unsigned integer as specified by the **ColorImportant** field of the **BitmapInfoHeader** object specified in [MS-WMF] section 2.2.2.3. MUST be 0x00000000.

2.3.1.3 RGBQuad

Referenced by: TBCBitmap

Specifies the pixel color values in a **TBCBitmap** (section 2.3.1.1).



blue (1 byte): An unsigned integer that specifies the relative intensity of blue.

green (1 byte): An unsigned integer that specifies the relative intensity of green.

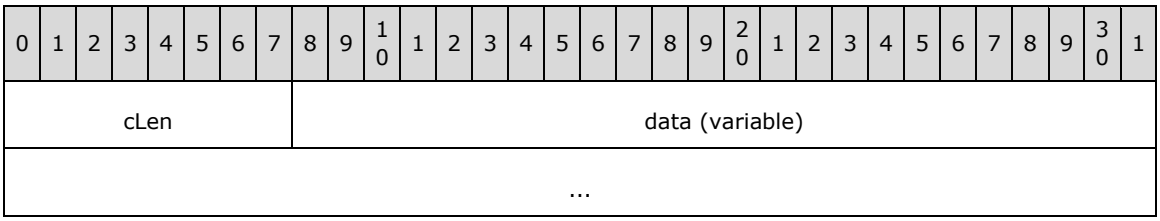
red (1 byte): An unsigned integer that specifies the relative intensity of red.

reserved (1 byte): Undefined and MUST be ignored.

2.3.1.4 WString

Referenced by: TB, TBCBSpecific, TBCCDData, TBCExtraInfo, TBCGeneralInfo, TBCMenuSpecific

String structure that **toolbar** customization structures use. This structure specifies a non-null-terminated **Unicode** string.



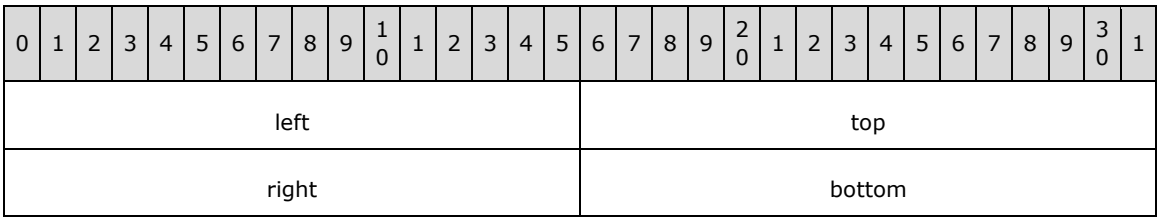
cLen (1 byte): Unsigned integer that specifies the count of characters in this string.

data (variable): Array of Unicode characters. The number of characters in the array MUST be equal to the value of the **cLen** field. Because this is a non-null-terminated Unicode string all of the elements of this string MUST NOT have a value of 0x0000.

2.3.1.5 SRECT

Referenced by: TBVisualData

Specifies a rectangle structure.



left (2 bytes): Signed integer that specifies the position in pixels of the left side of the rectangle.

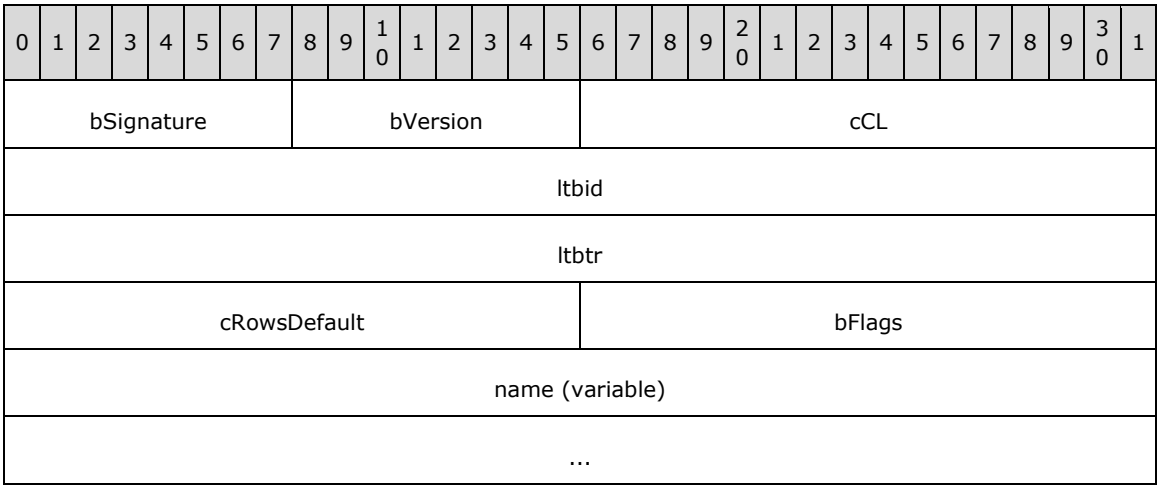
top (2 bytes): Signed integer that specifies the position in pixels of the top side of the rectangle.

right (2 bytes): Signed integer that specifies the position in pixels of the right side of the rectangle.

bottom (2 bytes): Signed integer that specifies the position in pixels of the bottom side of the rectangle.

2.3.1.6 TB

Structure that contains **toolbar** information.



bSignature (1 byte): Signed integer that specifies the toolbar signature number. MUST be 0x02.

bVersion (1 byte): Signed integer that specifies the toolbar version number. MUST be 0x01.

cCL (2 bytes): Signed integer that SHOULD<2> specify the number of **toolbar controls** contained in this toolbar.

ltbid (4 bytes): Signed integer that specifies the toolbar ID. MUST be 0x00000001 (**custom toolbar** ID).

ltbtr (4 bytes): Unsigned integer of type **TBTRFlags** (section 2.3.1.7) that specifies the toolbar type and toolbar restrictions.

cRowsDefault (2 bytes): Unsigned integer that specifies the number of preferred rows for the toolbar when the toolbar is not **docked**. MUST be less than or equal to 255.

bFlags (2 bytes): Unsigned integer of type **TBFlags** (section 2.3.1.8).

name (variable): Structure of type **WString** (section 2.3.1.4) that specifies the toolbar name.

2.3.1.7 TBTRFlags

Referenced by: TB

Toolbar type and restrictions flags. The bit description begins from the least significant bit.

This structure MUST specify the toolbar type which means that the 8 most significant bits MUST have one of the values in the following table.

8 most significant bits of the TBTRFlags structure (section 2.3.1.7)	Meaning
00000000	Specifies that the toolbar is a **basic toolbar**.
00000010	Specifies that the toolbar is a **menu toolbar**.

The rest of the bits in this structure specify toolbar restrictions. A toolbar restriction limits the end user's ability to change and manipulate the toolbar.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	a	reserved17				

A - NoAddDelCtl (1 bit): A bit that specifies whether **toolbar controls** can be added to or removed from this toolbar. A value of 1 specifies that no toolbar controls can be added or removed from this toolbar.

B - NoResize (1 bit): A bit that specifies whether this toolbar can be resized when not **docked**. A value of 1 specifies that this toolbar cannot be resized when not docked.

C - NoMove (1 bit): A bit that specifies whether this toolbar can or cannot be moved. A value of 1 specifies that this toolbar cannot be moved.

D - NoChangeVisible (1 bit): A bit that specifies whether the visibility of this toolbar can be changed. A value of 1 specifies that the visibility of this toolbar cannot be changed.

E - NoChangeDock (1 bit): A bit that specifies whether the **docked location** of this toolbar can or cannot be changed. A value of 1 specifies that the end user cannot change the docked location of this toolbar.

F - NoVerticalDock (1 bit): A bit that specifies whether this toolbar can be vertically docked. A value of 1 specifies that the toolbar cannot be vertically docked.

G - NoHorizontalDock (1 bit): A bit that specifies whether this toolbar can be horizontally docked. A value of 1 specifies that the toolbar cannot be horizontally docked.

H - NoBorder (1 bit): A bit that specifies whether the toolbar has a title bar and a border when not docked. A value of 1 specifies that the toolbar does not have a title bar and a border when not docked. MUST be 1 when the **TBTPopupMenu** field equals 1; MUST be 0 when the **TBTPopupMenu** equals 0.

I - NoTbContextMenu (1 bit): A bit that specifies whether the toolbar shows a **context menu** when right-clicked. A value of 1 specifies that the toolbar will not show a context menu when right-clicked. MUST be 1 when **TBTPopupMenu** equals 1; MUST be 0 when **TBTPopupMenu** equals 0.

J - reserved1 (1 bit): Reserved bit. MUST be 0.

K - reserved2 (1 bit): Reserved bit. MUST be 0.

L - NotTopLevel (1 bit): A bit that specifies whether the toolbar can be a **top-level toolbar**. A value of 1 specifies that this toolbar is a child of another toolbar, and can never be a top-level toolbar. A value of 0 specifies that this toolbar can be a top-level toolbar. MUST be 1 when the **TBTPopupMenu** field equals 1; MUST be 0 when the **TBTPopupMenu** field equals 0.

M - reserved3 (1 bit): Reserved bit. MUST be 0.

N - reserved4 (1 bit): Reserved bit. MUST be 0.

O - reserved5 (1 bit): Reserved bit. MUST be 0.

P - reserved6 (1 bit): Reserved bit. MUST be 0.

Q - reserved7 (1 bit): Reserved bit. MUST be 0.

R - reserved8 (1 bit): Reserved bit. MUST be 0.

S - reserved9 (1 bit): Reserved bit. MUST be 0.

T - reserved10 (1 bit): Reserved bit. MUST be 0.

U - reserved11 (1 bit): Reserved bit. MUST be 0.

V - reserved12 (1 bit): Reserved bit. MUST be 0.

W - reserved13 (1 bit): Reserved bit. MUST be 0.

X - reserved14 (1 bit): Reserved bit. MUST be 0.

Y - reserved15 (1 bit): Reserved bit. MUST be 0.

Z - TBTPopupMenu (1 bit): A bit that specifies whether this toolbar is of type menu toolbar. A value of 1 specifies that this is a menu toolbar. If the value equals 1, the fields **NoResize**, **NoMove**, **NoChangeDock**, **NoBorder**, **NoTbContextMenu**, and **NotTopLevel** MUST equal 1.

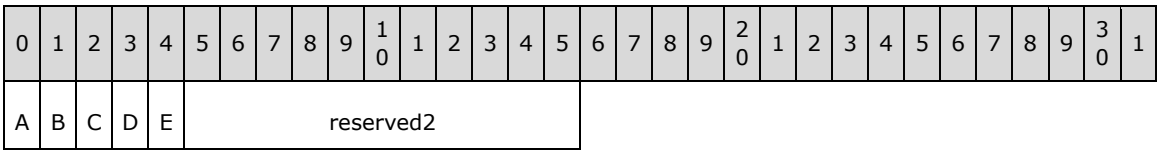
a - reserved16 (1 bit): Reserved bit. MUST be 0.

reserved17 (5 bits): Reserved bits. MUST be 0.

2.3.1.8 TBFlags

Referenced by: TB

Toolbar flags. The bit description begins from the least significant bit.



A - fDisabled (1 bit): A bit that specifies whether this toolbar is disabled. A value of 1 specifies that this toolbar is disabled.

B - reserved1 (1 bit): Undefined and MUST be ignored.

C - fCtlModified (1 bit): A bit that specifies whether the **toolbar controls** of this toolbar have to be saved to the file. A value of 1 specifies that the toolbar controls of this toolbar have to be saved to the file. MAY be 0 and the toolbar MAY still save its toolbar controls to the file.<3>

D - fNoAdaptiveMenus (1 bit): A bit that specifies whether **adaptive menus** are disabled for this toolbar. A value of 1 specifies that adaptive menus are disabled for this toolbar.

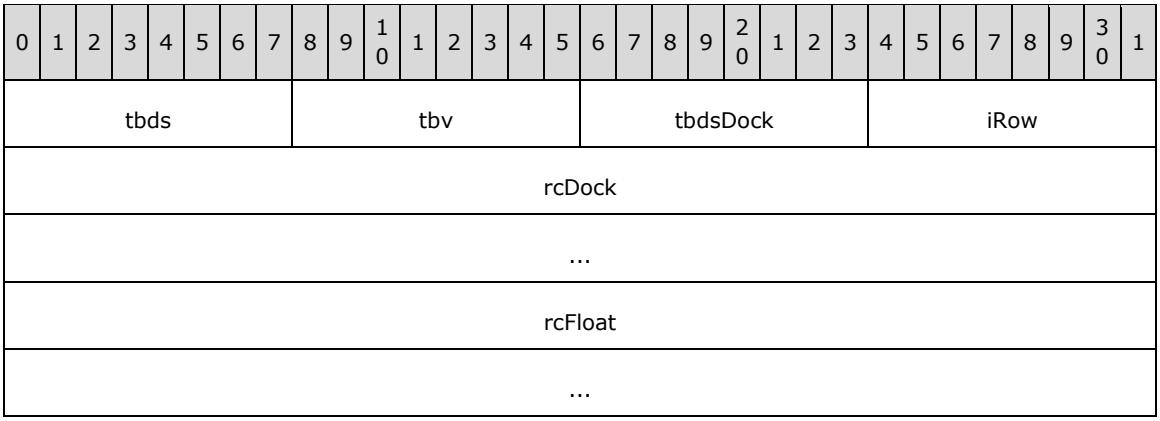
E - fNeedsPositioning (1 bit): A bit that specifies whether the toolbar's non-**docked** position needs to be updated for display on multiple monitors. A value of 1 specifies that the toolbar's non-docked position needs to be updated for display on multiple monitors.

reserved2 (11 bits): Reserved bits. MUST be 0.

2.3.1.9 TBVisualData

Contains visual information about a **toolbar**. The values of some of the fields in this structure are restricted by the toolbar type and restrictions of this toolbar (the value of the **ltbtr** field of the **TB** structure (section 2.3.1.6) that contains the structure that contains this structure). The restrictions are shown in the following table.

Value of toolbar types and restrictions flags (value of the ltbtr field of the TB structure that contains the structure that contains this structure)	TBVisualData restrictions
If the **NotTopLevel** bit of the **ltbtr** field of the **TB** structure (section 2.3.1.6) that contains toolbar information for this toolbar equals 1.	**tbv** MUST equal 0x00 or **tbds** MUST equal 0x04.
If the **TBTPopupMenu** bit of the **ltbtr** field of the **TB** structure (section 2.3.1.6) that contains toolbar information for this toolbar equals 1.	**tbv** MUST equal 0x00 or **tbds** MUST equal 0x04.
If the **NoVerticalDock** bit of the **ltbtr** field of the **TB** structure (section 2.3.1.6) that contains toolbar information for this toolbar equals 1.	**tbds** MUST NOT equal 0x00 or 0x02.
If the **NoHorizontalDock** bit of the **ltbtr** field of the **TB** structure (section 2.3.1.6) that contains toolbar information for this toolbar equals 1.	**tbds** MUST NOT equal 0x01 or 0x03.



tbds (1 byte): Signed integer that specifies the current toolbar **docked** state. The value MUST be in the following table.

Value	Meaning
0x00	Toolbar is docked on application frame at the left.
0x01	Toolbar is docked on application frame at the top.
0x02	Toolbar is docked on application frame at the right.
0x03	Toolbar is docked on application frame at the bottom.
0x04	Toolbar is not docked.

tbv (1 byte): Signed integer that specifies the toolbar visibility. The value MUST be in the following table.

Value	Meaning
0x00	Toolbar is not visible.
0x01	Toolbar is visible.
0x02	Toolbar is visible.

tbdsDock (1 byte): Signed integer that SHOULD specify the toolbar docked state. The value MUST be in the following table.

Value	Meaning
0x00	Toolbar is docked on application frame at the left.
0x01	Toolbar is docked on application frame at the top.
0x02	Toolbar is docked on application frame at the right.
0x03	Toolbar is docked on application frame at the bottom.

If the value of the **tbds** field is less than or equal to 0x03, this value MUST be equal to the value of the **tbds** field. When tbds is greater than 0x03, this field specifies the most recent toolbar dock state.

iRow (1 byte): Signed integer that specifies, when the toolbar is docked, the index of the **docked location** of the toolbar in the **docking area** in relation to other toolbars in the docking area. MUST be in the range 0-127 or in the following table.

Value	Meaning
0xFE (-2)	Row append. Toolbar is placed on the last row of the docking area.
0xFD (-3)	Row prepend. Toolbar is placed on the first row of the docking area.

rcDock (8 bytes): Structure of type **SRECT** (section 2.3.1.5) that specifies the preferred docked location of the toolbar. Refer to the following table for the meaning of the values of each field of this structure.

Field of rcDock structure	Meaning
rcDock.left	Signed integer that specifies the distance in pixels from the left border of the docking area to the left border of the toolbar.
rcDock.top	Signed integer that specifies the distance in pixels from the top border of the docking area to the top border of the toolbar.
rcDock.right	Signed integer that specifies the distance in pixels from the left border of the docking area to the right border of the toolbar.
rcDock.bottom	Signed integer that specifies the distance in pixels from the top border of the docking area to the bottom border of the toolbar.

rcFloat (8 bytes): Structure of type **SRECT** (section 2.3.1.5) that specifies the preferred toolbar location when the toolbar is not docked. Refer to the following table for the meaning of the values of each field of this structure.

Field of rcFloat structure	Meaning
rcFloat.left	Signed integer that specifies the distance in pixels from the left border of the screen to the left border of the toolbar.
rcFloat.top	Signed integer that specifies the distance in pixels from the top border of the screen to the top border of the toolbar.
rcFloat.right	Signed integer that specifies the distance in pixels from the left border of the screen to the right border of the toolbar.
rdFloat.bottom	Signed integer that specifies the distance in pixels from the top border of the screen to the bottom border of the toolbar.

2.3.1.10 TBCHeader

Toolbar control header information.

bSignature (1 byte): Signed integer that specifies the toolbar control signature number. MUST be 0x03.

bVersion (1 byte): Signed integer that specifies the toolbar control version number. MUST be 0x01.

bFlagsTCR (1 byte): Structure of type **TBCFlags** (section 2.3.1.11) that specifies toolbar control flags.

tct (1 byte): Unsigned integer that specifies the **toolbar control type**. The value MUST be in the following table:

Value	Toolbar control type
0x01	Button control
0x02	Edit control
0x03	DropDown control
0x04	ComboBox control
0x06	**SplitDropDown control**
0x07	**OCXDropDown control**
0x09	**GraphicDropDown control**
0x0A	**Popup control**
0x0C	**ButtonPopup control**
0x0D	**SplitButtonPopup control**
0x0E	**SplitButtonMRUPopup control**
0x0F	Label control
0x10	**ExpandingGrid control**
0x12	**Grid control**
0x13	**Gauge control**
0x14	**GraphicCombo control**
0x15	**Pane control**
0x16	**ActiveX control**

tcid (2 bytes): Unsigned integer that specifies the **toolbar control identifier (TCID)** for this toolbar control. MUST be 0x0001 when the toolbar control is a **custom toolbar control** or MUST be equal to one of the values listed in [MS-CTDOC] section 2.2 or in [MS-CTXLS] section 2.2 when the toolbar control is not a custom toolbar control.

tbct (4 bytes): Structure of type **TBCSFlags** (section 2.3.1.12) that specifies toolbar control flags.

bPriority (1 byte): Unsigned integer that specifies the toolbar control priority for dropping and wrapping purposes. The value MUST be in the range 0x00 to 0x07. If the value equals 0x00, it is considered the default state. If it equals 0x01 the toolbar control will never be dropped from the **toolbar** and will be wrapped when needed. Otherwise, the higher the number the sooner the toolbar control will be dropped.

width (2 bytes): Unsigned integer that specifies the width, in pixels, of the toolbar control. MUST only exist if **bFlagsTCR.fSaveDxy** equals 1.

height (2 bytes): Unsigned integer that specifies the height, in pixels, of the toolbar control. MUST only exist if **bFlagsTCR.fSaveDxy** equals 1.

2.3.1.11 TBCFlags

Referenced by: TBCHeader

Toolbar control flags. The bit description begins from the least significant bit.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
A	B	C	D	E	F	G	H																								

A - fHidden (1 bit): A bit that specifies whether this toolbar control is visible. A value of 1 specifies that the toolbar control is not visible.

B - fBeginGroup (1 bit): A bit that specifies whether a **toolbar control separator** appears before this toolbar control. A value of 1 specifies that a toolbar control separator appears before the toolbar control.

C - fOwnLine (1 bit): A bit that specifies whether the toolbar control requires its own row. A value of 1 specifies that the toolbar control requires its own row in the **toolbar**.

D - fNoCustomize (1 bit): A bit that specifies whether this toolbar control can be altered by customization. A value of 1 specifies that the toolbar control cannot be altered by customization.

E - fSaveDxy (1 bit): A bit that specifies whether the width and height of the toolbar control have been saved to the file. A value of 1 specifies that the **width** and **height** fields of the **TBCHeader** structure (section 2.3.1.10) that contains this structure MUST exist.

F - reserved1 (1 bit): Reserved bit. Undefined and MUST be ignored.

G - fBeginLine (1 bit): A bit that specifies whether the toolbar control begins a new row in the toolbar. A value of 1 specifies that the toolbar control begins a new row in the toolbar.

H - reserved2 (1 bit): Reserved bit. MUST be 0.

2.3.1.12 TBCSFlags

Referenced by: TBCHeader

Toolbar control flags that manage specific properties of a toolbar control. The bit description begins from the least significant bit.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
A		B	C	D	E	F	G	unused1								H	I	J	K	L	M	N	O	P	Q	R	unused3				S

A - textIcon (2 bits): Unsigned integer that specifies the visibility of the label and icon of this toolbar control. The value MUST be in the following table:

Value (value of bits in parenthesis)	Meaning - When toolbar control is on a basic toolbar	Meaning - When toolbar control is on a menu toolbar
0x00 (00)	Text is not visible and icon is visible	Text is visible and icon is visible.
0x01 (01)	Text is not visible and icon is visible	Text is visible and icon is not visible
0x02 (10)	Text is visible and icon is not visible.	Text is visible and icon is not visible.
0x03 (11)	Text is visible and icon is visible.	Text is visible and icon is visible.

B - fOwnerDraw (1 bit): A bit that specifies whether this toolbar control uses an internal rendering option. A value of 1 specifies that this toolbar control uses an internal rendering option. MUST be 0 if the **tcid** value of the **TBCHeader** structure (section 2.3.1.10) that contains this structure equals 0x0001. If the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001, this value MUST be equal to the value associated with the **tcid** listed in [MS-CTDOC] section 2.2 or in [MS-CTXLS] section 2.2.

C - fAllowResize (1 bit): A bit that specifies whether sizing is allowed for this toolbar control. A value of 1 specifies that toolbar control sizing is allowed. This flag is used when a toolbar control is being initialized and allows for the toolbar control to be smaller or bigger than normal. MUST be 0 if the **tcid** value of the **TBCHeader** structure that contains this structure equals 0x0001. If the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001, this value MUST be equal to the value associated with the **tcid** listed in [MS-CTDOC] section 2.2 or in [MS-CTXLS] section 2.2.

D - fOneState (1 bit): A bit that specifies whether this is a one-state toolbar control. This bit is only used by toolbar controls of type Button or ExpandingGrid. A value of 1 specifies that the toolbar control can have only one state (ButtonUp, the value of the **state** field of the **TBCBSFlags** structure (section 2.3.1.18) contained in the **TBCBSpecific** structure (section 2.3.1.17) contained in the **TBCData** structure (section 2.3.1.13) that contains toolbar control information for this toolbar control equals 0). MUST be 0 if the **tcid** value of the **TBCHeader** structure that contains this structure equals 0x0001. If the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001, this value MUST be equal to the value associated with the **tcid** listed in [MS-CTDOC] section 2.2 or in [MS-CTXLS] section 2.2.

E - fNoSetCursor (1 bit): A bit that specifies whether the toolbar control can change the mouse cursor when it is over the toolbar control area. A value of 1 specifies that the toolbar control can change the mouse cursor when this is over the toolbar control area. MUST be 0 if the **tcid** value of the **TBCHeader** structure that contains this structure equals 0x0001. SHOULD<4> be 0 if the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001.

F - fNoAccel (1 bit): A bit that specifies whether this toolbar control has **accelerator keys**. A value of 1 specifies that the toolbar control does not have accelerator keys. MUST be 0 if the **tcid** value of the **TBCHeader** structure that contains this structure equals 0x0001. SHOULD<5> be 0 if the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001.

G - fChgAccel (1 bit): A bit that specifies whether the accelerator keys for the toolbar control can change. A value of 1 specifies that the accelerator keys can be changed by the application. MUST be 0 if the **tcid** value of the **TBCHeader** structure that contains this structure equals 0x0001. If the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001, this value MUST be equal to the value associated with the **tcid** listed in [MS-CTDOC] section 2.2 or in [MS-CTXLS] section 2.2.

unused1 (8 bits): Undefined and MUST be ignored.

H - fAlwaysEnabled (1 bit): A bit that specifies whether this toolbar control is enabled by default. A value of 1 specifies that the toolbar control is enabled by default. MUST be 0 if the **tcid** value of the **TBCHeader** structure that contains this structure equals 0x0001. SHOULD<6> be 0 if the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001.

I - fAlwaysVisible (1 bit): A bit that specifies whether this toolbar control is visible by default. A value of 1 specifies that the toolbar control is visible by default. MUST be 0 if the **tcid** value of the **TBCHeader** structure that contains this structure equals 0x0001. If the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001, this value MUST be equal to the value associated with the **tcid** listed in [MS-CTDOC] section 2.2 or in [MS-CTXLS] section 2.2.

J - fNoChangeLabel (1 bit): A bit that specifies whether the label of the toolbar control can change. A value of 1 specifies that the toolbar control label does not be changed by the application. MUST be 0 if the **tcid** value of the **TBCHeader** structure (section 2.3.1.10) that contains this structure equals 0x0001. If the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001, this value MUST be equal to the value associated with the **tcid** listed in [MS-CTDOC] section 2.2 or in [MS-CTXLS] section 2.2.

K - fKeepLabel (1 bit): A bit that specifies whether the label of the toolbar control can change. A value of 1 specifies that the toolbar control label will not be changed by the application unless the toolbar control is reset.

L - fNoQueryTooltip (1 bit): A bit that specifies whether the toolbar control can use an internal string as a **ToolTip**. A value of 1 specifies that the toolbar control will not use an internal string as a ToolTip. If the toolbar control has a custom ToolTip, it will use it. MUST be 0 if the **tcid** value of the **TBCHeader** structure (section 2.3.1.10) that contains this structure equals 0x0001. If the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001, this value MUST be equal to the value associated with the **tcid** listed in [MS-CTDOC] section 2.2 or in [MS-CTXLS] section 2.2.

M - fSaveUIStrings (1 bit): A bit that specifies whether none, one, or more of a variety of strings are saved to the file. A value of 1 specifies that one or more of the following fields will be saved to the file: **customText**, **descriptionText**, and **tooltip** fields of the **TBCGeneralInfo** structure (section 2.3.1.14) contained by the **TBCData** structure (section 2.3.1.13) contained by the structure that contains the **TBCHeader** structure (section 2.3.1.10) that contains this structure and if this toolbar control is of type Button or ExpandingGrid, the **wstrAcc** field of the **TBCBSpecific** structure (section 2.3.1.17) contained by the **TBCData** structure contained by the structure that contains the **TBCHeader** structure that contains this structure. When the value of the **tcid** field of the **TBCHeader** structure that contains the **TBCSFlags** structure (section 2.3.1.12) that contains this field equals 1, **fSaveUIStrings** is equal to 1, even if no extra strings are saved to the file.

N - fExclusivePopup (1 bit): A bit that specifies whether the toolbar control is going to drop a unique **custom toolbar**. This bit is only used by toolbar controls that drop a **menu toolbar**. A value of 1 specifies that the toolbar control is going to drop a unique custom toolbar. SHOULD<7> be 0.

O - fDefaultBehavior (1 bit): A bit that specifies whether the toolbar control will have default behavior during **OLE** merging. A value of 1 specifies that the toolbar control will have default behavior during OLE merging. A value of 0 specifies that the application can change the behavior of the toolbar control during OLE merging. SHOULD<8> be 0.

P - unused2 (1 bit): Undefined and MUST be ignored.

Q - fWrapText (1 bit): A bit that specifies whether the toolbar control can wrap its label across multiple lines. A value of 1 specifies that the label of the toolbar control can wrap across multiple lines.

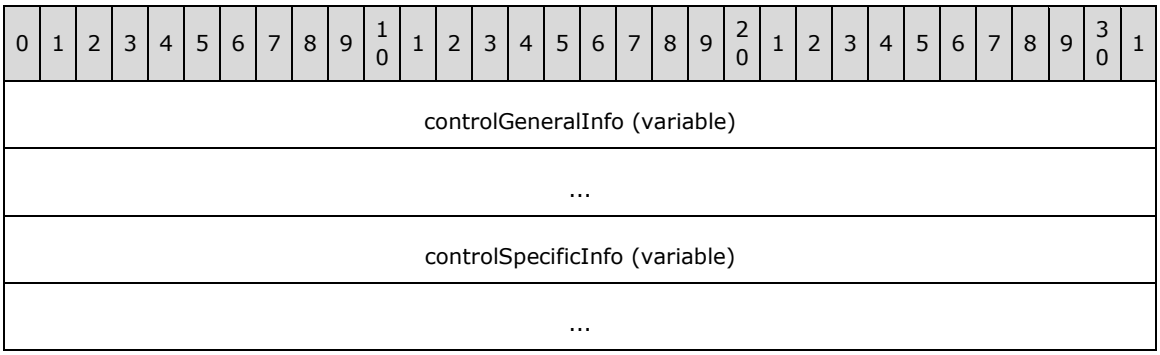
R - fTextBelow (1 bit): A bit that specifies that the label of the toolbar control will be displayed under the toolbar control icon. A value of 1 specifies that the label of the toolbar control will be displayed under the toolbar control icon, rather than beside it.

unused3 (4 bits): Undefined and MUST be ignored.

S - reserved1 (1 bit): Reserved bit. MUST be 0.

2.3.1.13 TBCData

Toolbar control information.



controlGeneralInfo (variable): Structure of type **TBCGeneralInfo** (section 2.3.1.14) that specifies toolbar control general information.

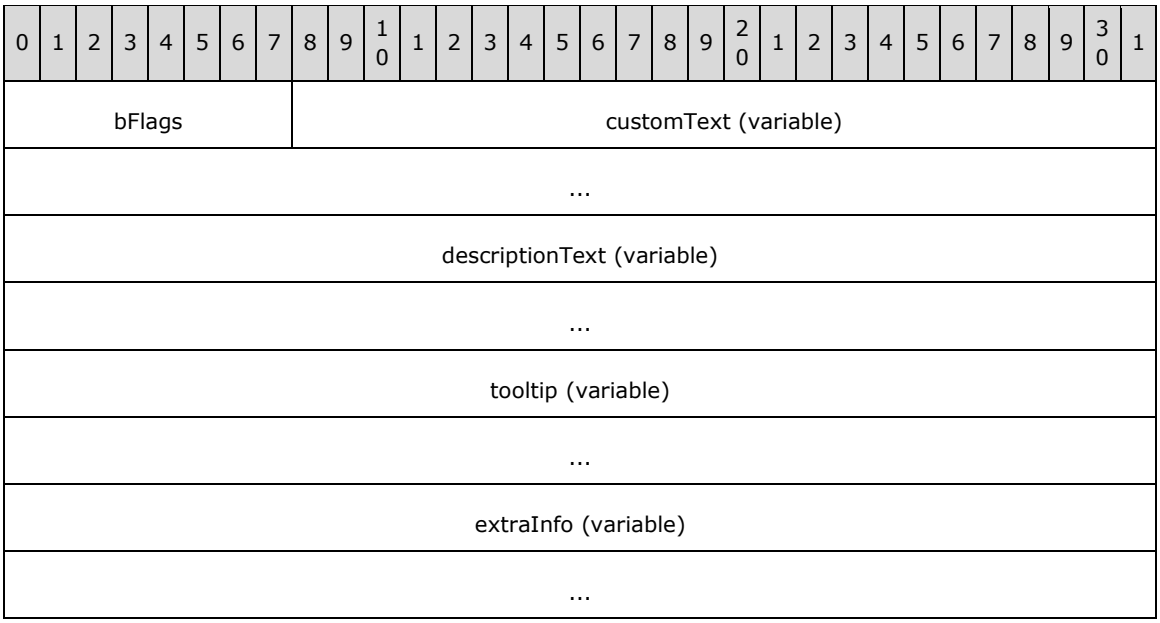
controlSpecificInfo (variable): Toolbar control specific information is saved depending on the type of the toolbar control which is specified by the value of the **tct** field of the TBCHeader structure (section 2.3.1.10) contained by the structure that contains this structure. The following table shows the type of structure that is saved according to the type of the toolbar control:

Value of the tct field	Type of the controlSpecificInfo field
0x01 (Button control)	**TBCBSpecific** (section 2.3.1.17)
0x10 (**ExpandingGrid control**)	**TBCBSpecific**
0x0A (**Popup control**)	**TBCMenuSpecific** (section 2.3.1.21)
0x0C (**ButtonPopup control**)	**TBCMenuSpecific**
0x0D (**SplitButtonPopup control**)	**TBCMenuSpecific**
0x0E (**SplitButtonMRUPopup control**)	**TBCMenuSpecific**
0x02 (Edit control)	**TBCComboDropdownSpecific** (section 2.3.1.19)
0x04 (ComboBox control)	**TBCComboDropdownSpecific**
0x14 (**GraphicCombo control**)	**TBCComboDropdownSpecific**
0x03 (DropDown control)	**TBCComboDropdownSpecific**
0x06 (**SplitDropDown control**)	**TBCComboDropdownSpecific**
0x09 (**GraphicDropDown control**)	**TBCComboDropdownSpecific**
0x07 (**OCXDropDown control**)	**controlSpecificInfo** MUST NOT exist
0x0F (Label control)	**controlSpecificInfo** MUST NOT exist
0x12 (**Grid control**)	**controlSpecificInfo** MUST NOT exist
0x13 (**Gauge control**)	**controlSpecificInfo** MUST NOT exist
0x16 (**ActiveX control**)	**controlSpecificInfo** MUST NOT exist

2.3.1.14 TBCGeneralInfo

Referenced by: TBCData

Toolbar control general information.



bFlags (1 byte): Structure of type **TBCGIFlags** (section 2.3.1.15) that specifies which of the fields of this structure have been saved to the file.

customText (variable): Structure of type **WString** (section 2.3.1.4) that specifies the custom label of the toolbar control. MUST exist if **bFlags.fSaveText** equals 1. MUST NOT exist if **bFlags.fSaveText** equals 0.

descriptionText (variable): Structure of type **WString** that specifies a description of this toolbar control. MUST exist if **bFlags.fSaveMiscUIStrings** equals 1. MUST NOT exist if **bflags.fSaveMiscUIString** equals 0.

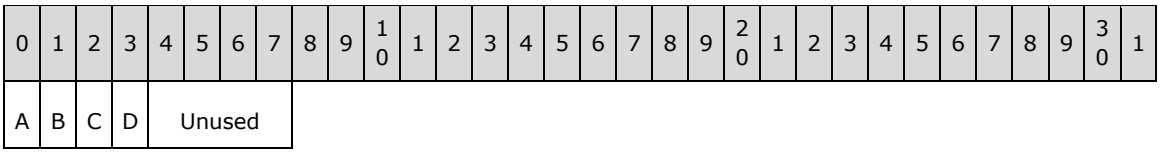
tooltip (variable): Structure of type **WString** that SHOULD specify the **ToolTip** of this toolbar control. MUST exist if **bFlags.fSaveMiscUIStrings** equals 1. MUST NOT exist if **bFlags.fSaveMiscUIStrings** equals 0. If the toolbar control is of type Button or ExpandingGrid, and the fHyperlinkType field of the TBCBSFlags structure (section 2.3.1.18) contained by the TBCBSpecific structure (section 2.3.1.17), contained by the TBCData structure (section 2.3.1.13) that contains the TBCGeneralInfo structure (section 2.3.1.14) that contains this structure does not equal 0, the value of tooltip specifies the hyperlink path for the toolbar control.

extraInfo (variable): Structure of type **TBCExtraInfo** (section 2.3.1.16) that specifies extra information saved for a toolbar control. MUST exist if **bFlags.fSaveMiscCustom** equals 1. MUST NOT exist if **bFlags.fSaveMiscCustom** equals 0.

2.3.1.15 TBCGIFlags

Referenced by: TBCGeneralInfo

Toolbar control general information flags that specify which fields in the **TBCGeneralInfo** structure (section 2.3.1.14) that contains this structure will be saved to the file. The bit description begins from the least significant bit.



A - fSaveText (1 bit): A bit that specifies whether the toolbar control will save its custom text. A value of 1 specifies that the **customText** field of the **TBCGeneralInfo** structure (section 2.3.1.14) that contains this structure MUST exist. If the value equals 1, the value of the **fSaveUIStrings** field of the **TBCSFlags** structure (section 2.3.1.12) contained by the **TBCHeader** structure (section 2.3.1.10) contained by the structure that contains the TBCData structure (section 2.3.1.13) that contains the **TBCGeneralInfo** structure (section 2.3.1.14) that contains this structure MUST be 1.

B - fSaveMiscUIStrings (1 bit): A bit that specifies whether the toolbar control will save its description and **ToolTip** strings. A value of 1 specifies that the **descriptionText** and **tooltip** fields of the **TBCGeneralInfo** structure that contains this structure MUST exist. If the value equals 1, the value of the **fSaveUIStrings** field of the **TBCSFlags** structure contained by the **TBCHeader** structure contained by the structure that contains the **TBCData** structure that contains the **TBCGeneralInfo** structure that contains this structure MUST be 1.

C - fSaveMiscCustom (1 bit): A bit that specifies whether the toolbar control will save toolbar control extra information. A value of 1 specifies that the **extraInfo** field of the **TBCGeneralInfo** structure that contains this structure MUST exist.

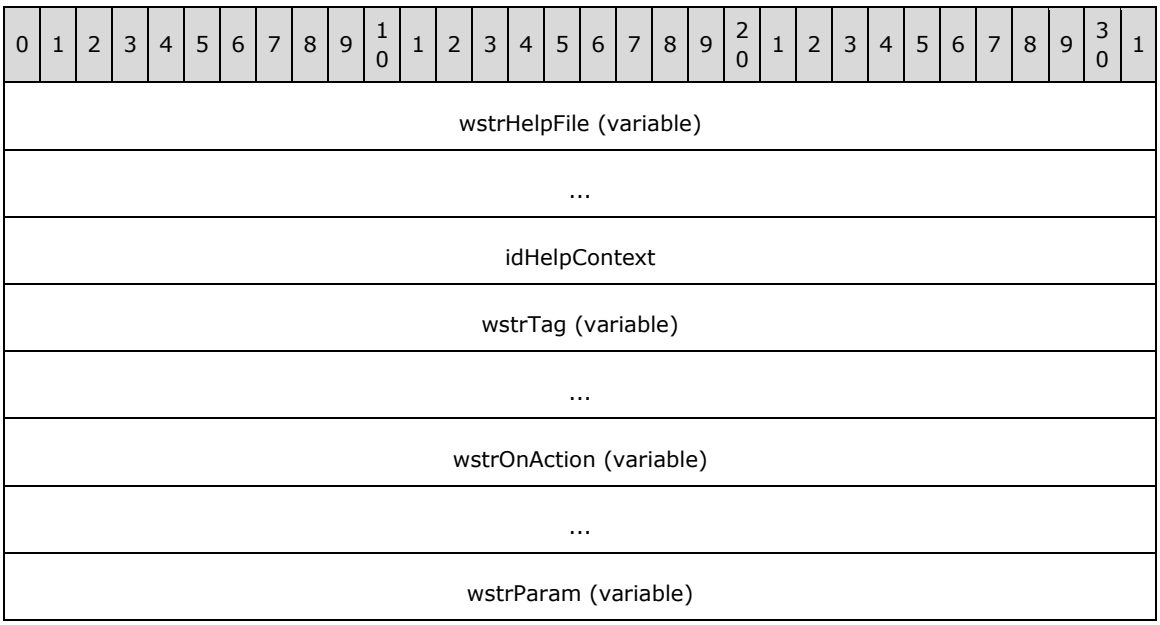
D - fDisabled (1 bit): A bit that specifies whether the toolbar control is disabled. A value of 1 specifies that the toolbar control is disabled.

Unused (4 bits): Undefined and MUST be ignored.

2.3.1.16 TBCExtraInfo

Referenced by: TBCGeneralInfo

Structure that specifies extra information saved for a **toolbar control**.



wstrHelpFile (variable): A structure of type **WString** (section 2.3.1.4) that specifies the full path to the help file used to provide the help topic of the toolbar control. For this field to be used **idHelpContext** MUST be greater than zero.

idHelpContext (4 bytes): Signed integer that specifies the help context id number for the help topic of the toolbar control. A help context id is a numeric identifier associated with a specific help topic. For this field to be used **wstrHelpFile** MUST specify a non-empty string.

wstrTag (variable): Structure of type **WString** that specifies a custom string used to store arbitrary information about the toolbar control.

wstrOnAction (variable): Structure of type **WString** that specifies the name of the macro associated with this toolbar control.

wstrParam (variable): Structure of type **WString** that specifies a custom string used to store arbitrary information about the toolbar control.

tbcu (1 byte): Signed integer that specifies how the toolbar control will be used during **OLE** merging. The value MUST be in the following table.

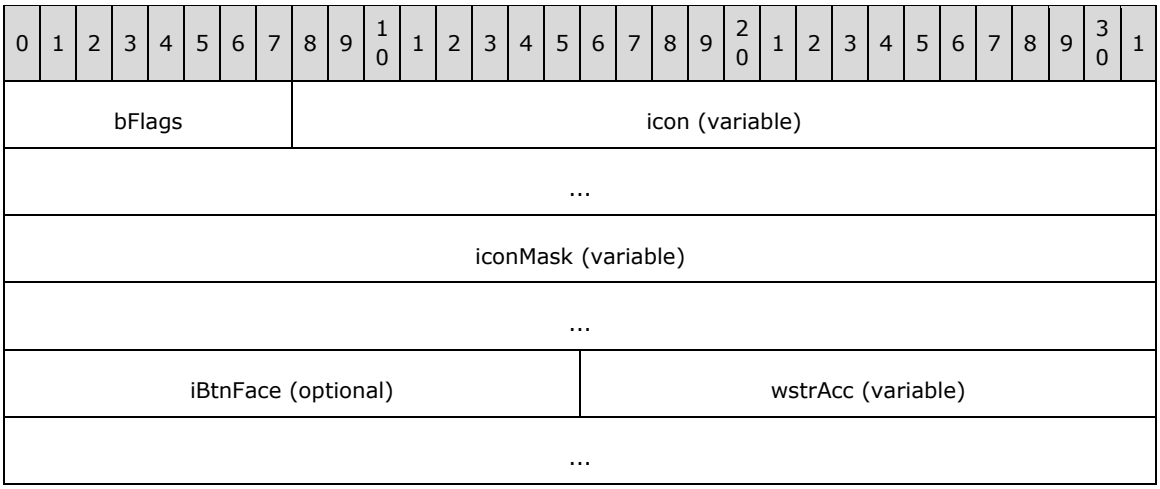
Value	Meaning
0xFF	A correct value was not found for this toolbar control. A value of 0x01 will be used when the value of this field is requested.
0x00	Neither. Toolbar control is not applicable when the application is in either OLE host mode or OLE server mode.
0x01	Server. Toolbar control is applicable when the application is in OLE server mode. (This is the default value used by **custom toolbar controls**.)
0x02	Host. Toolbar control is applicable when the application is in OLE host mode.
0x03	Both. Toolbar control is applicable when the application is in OLE server mode and OLE host mode.

tbmg (1 byte): Signed integer that specifies how the toolbar control will be used during OLE menu merging. This field is only used by toolbar controls of type Popup. The value MUST be in the following table.

Value	Meaning
0xFFFF	None. Toolbar control will not be placed in any OLE menu group.
0x0000	File. Toolbar control will be placed in the File OLE menu group.
0x0001	Edit. Toolbar control will be placed in the Edit OLE menu group.
0x0002	Container. Toolbar control will be placed in the Container OLE menu group.
0x0003	Object. Toolbar control will be placed in the Object OLE menu group.
0x0004	Window. Toolbar control will be placed in the Window OLE menu group.
0x0005	Help. Toolbar control will be placed in the Help OLE menu group.

2.3.1.17 TBCBSpecific

Contains information specific to button and **ExpandingGrid** type **toolbar controls**.



bFlags (1 byte): Structure of type **TBCBSFlags** section 2.3.1.18) that specifies which fields of this structure are saved to the file.

icon (variable): Structure of type **TBCBitmap** (section 2.3.1.1) that specifies the icon data for this toolbar control. MUST exist if **bFlags.fCustomBitmap** equals 1. MUST NOT exist if **bFlags.fCustomBitmap** equals 0.

iconMask (variable): Structure of type **TBCBitmap** (section 2.3.1.1) that specifies the icon data mask for this toolbar control. MUST exist if **bFlags.fCustomBitmap** equals 1. MUST NOT exist if **bFlags.fCustomBitmap** equals 0. The value of the **biBitCount** field of the **BITMAPINFOHEADER** (section 2.3.1.2) contained by this **TBCBitmap** MUST be 0x01. The **iconMask** is used to specify the transparency of the icon. The **iconMask** is white in all the areas in which the icon is displayed as transparent and is black in all other areas.

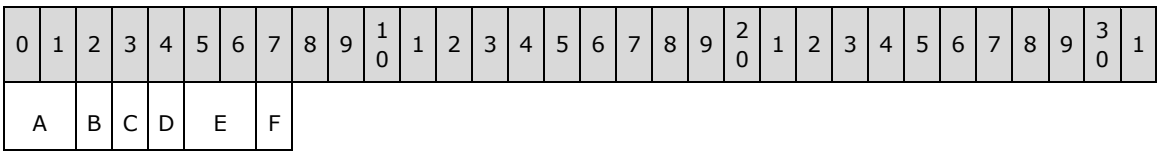
iBtnFace (2 bytes): Unsigned integer that specifies the icon of the toolbar control. When this value is set, the toolbar control will use the icon of the toolbar control whose **toolbar control identifier (TCID)** equals this value. MUST exist if **bFlags.fCustomBtnFace** equals 1. MUST NOT exist if **bFlags.fCustomBtnFace** equals 0.

wstrAcc (variable): Structure of type **WString** (section 2.3.1.4) that specifies the **accelerator keys** for this toolbar control. MUST exist if **bFlags.fAccelerator** equals 1. MUST NOT exist if **bFlags.fAccelerator** equals 0.

2.3.1.18 TBCBSFlags

Referenced by: TBCBSpecific

Contains flags for Button and ExpandingGrid type **toolbar controls**. The bit description begins from the least significant bit.



A - state (2 bits): Unsigned integer that specifies the toolbar control state. The value MUST be in the following table.

Value (value of bits in parenthesis)	Meaning
0x00 (00)	Button is up.
0x01 (01)	Button is down (pushed).
0x03 (11)	Button is in mixed state.

B - fAccelerator (1 bit): A bit that specifies whether the **wstrAcc** field of the **TBCBSpecific** structure (section 2.3.1.17) that contains this structure has been saved to the file. A value of 1 specifies that the **wstrAcc** field of the **TBCBSpecific** structure (section 2.3.1.17) that contains this structure MUST exist. If the value equals 1, the value of the **fSaveUIStrings** field of the **TBCSFlags** structure (section 2.3.1.12) contained by the **TBCHeader** structure (section 2.3.1.10) contained by the structure that contains the **TBCData** structure (section 2.3.1.13) that contains the **TBCBSpecific** structure (section 2.3.1.17) that contains this structure MUST be 1.

C - fCustomBitmap (1 bit): A bit that specifies whether the toolbar control has a custom icon. A value of 1 specifies that the **icon** and **iconMask** fields of the **TBCBSpecific** structure that contains this structure MUST exist.

D - fCustomBtnFace (1 bit): A bit that specifies whether the toolbar control is using an alternate icon as its own. A value of 1 specifies that the **iBtnFace** field of the **TBCBSpecific** structure that contains this structure MUST exist.

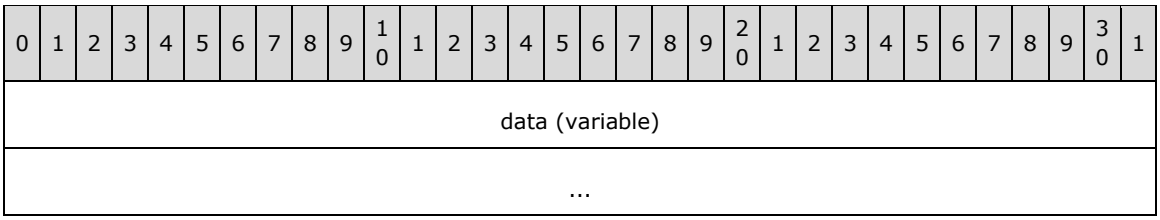
E - fHyperlinkType (2 bits): Unsigned integer that specifies the type of **hyperlink** associated with this toolbar control. The value MUST be in the following table.

Value (value of bits in parenthesis)	Hyperlink type
0x00 (00)	No hyperlink. This toolbar control does not have a hyperlink.
0x01 (01)	Open in browser. The hyperlink will be opened in the default browser.
0x02 (10)	Image link. The hyperlink links to an image file.

F - reserved1 (1 bit): Reserved bit. SHOULD<9> be 1.

2.3.1.19 TBCComboDropdownSpecific

Contains information specific to Edit, ComboBox, GraphicCombo, DropDown, SplitDropDown, and GraphicDropDown type **toolbar controls**.

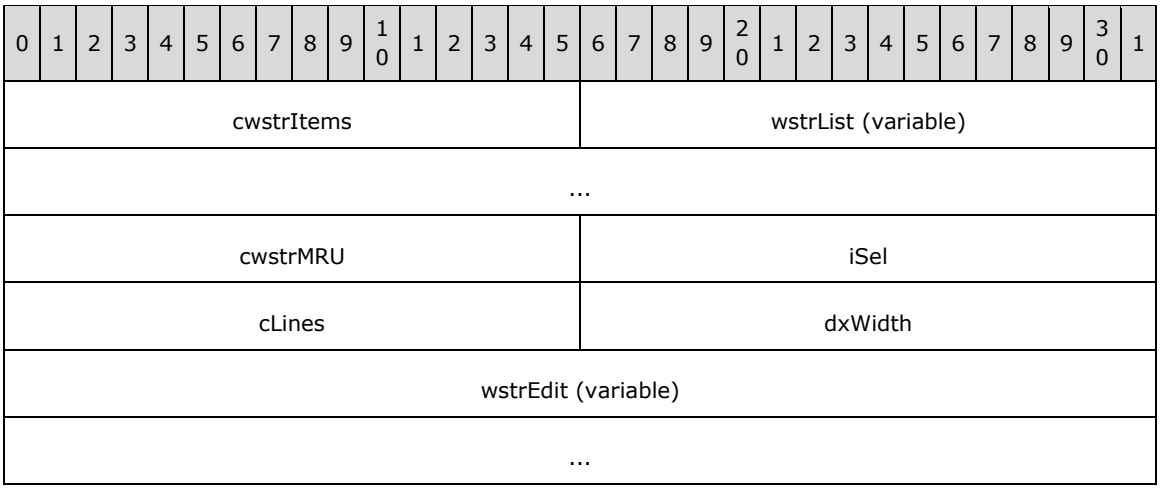


data (variable): Structure of type **TBCCDData** (section 2.3.1.20). MUST exist if the **toolbar control identifier (TCID)** (the **tcid** field of the **TBCHeader** structure (section 2.3.1.10) contained by the structure that contains the **TBCData** structure (section 2.3.1.13) that contains this structure) equals 0x0001. MUST NOT exist if the TCID does not equal 0x0001.

2.3.1.20 TBCCDData

Referenced by: TBCComboDropdownSpecific

Contains information specific to Edit, ComboBox, GraphicCombo, DropDown, SplitDropDown, and GraphicDropDown type **toolbar controls**.



cwstrItems (2 bytes): Signed integer that specifies the number of items in **wstrList**. MUST be positive.

wstrList (variable): Zero-based index array of **WString** structures (section 2.3.1.4). The number of elements MUST be equal to **cwstrItems**. Contains the list of strings that are dropped by this toolbar control.

cwstrMRU (2 bytes): Signed integer that specifies the number of most recently used strings. MUST be equal to 0xFFFF (-1) or greater than or equal to 0x0000. A value of 0xFFFF (-1) means that there are no most recently used strings.

iSel (2 bytes): Signed integer that specifies the zero-based index of the selected item in the **wstrList** field. MUST be equal to 0xFFFF (-1) or greater than or equal to 0x0000. A value of 0xFFFF (-1) means that there is no selected item. MUST be less than the **cwstrItems** value.

cLines (2 bytes): Signed integer that specifies the suggested number of lines that the toolbar control will display at any time when displaying the elements of **wstrList**. A value of 0x0000 means that the toolbar control will size itself according to the number of items. MUST equal to or greater than 0x0000.

dxWidth (2 bytes): Signed integer that specifies the width in pixels that the interior of the drop-down has. This excludes the width of the toolbar control border and scroll bar. MUST be equal to 0xFFFF (-1) or greater than or equal to 0x0000. A value of 0xFFFF (-1) will set the width to accommodate all the strings in **wstrList**.

wstrEdit (variable): Structure of type **WString** (section 2.3.1.4). Editable text for editable area of the ComboBox toolbar control.

2.3.1.21 TBCMenuSpecific

Contains information specific to **Popup**, **ButtonPopup**, **SplitButtonPopup**, and **SplitButtonMRUPopup** type **toolbar controls**.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
tbid																															
name (variable)																															
...																															

tbid (4 bytes): Signed integer that specifies the **toolbar** ID of the toolbar that the toolbar control drops. MUST be greater than or equal to 0x00000000. A value of 0x00000000 means that the toolbar control does not drop a toolbar. A value of 0x00000001 means that the toolbar control drops a **custom toolbar**. A value greater than 0x00000001 means that the toolbar control drops a built-in toolbar. See [MS-CTDOC] section 2.1 and [MS-CTXLS] section 2.1 for a list of toolbar identifiers associated with built-in toolbars.

name (variable): Structure of type **WString** (section 2.3.1.4). Name of the custom toolbar that the toolbar control drops. MUST exist if **tbid** equals 0x00000001. MUST NOT exist if **tbid** is not equal to 0x00000001.

2.3.2 Visual Basic for Applications Digital Signature Storage

Specifies the format of the **digital signature** for **Visual Basic for Applications (VBA)** projects.

2.3.2.1 DigSigInfoSerialized

Referenced by: DigSigBlob, WordSigBlob

Specifies the detailed data of a **VBA digital signature**.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
cbSignature																															
signatureOffset																															
cbSigningCertStore																															
certStoreOffset																															
cbProjectName																															
projectNameOffset																															
fTimestamp																															
cbTimestampUrl																															
timestampUrlOffset																															
pbSignatureBuffer (variable)																															

...
pbSigningCertStoreBuffer (variable)
...
rgchProjectNameBuffer (variable)
...
rgchTimestampBuffer (variable)
...

cbSignature (4 bytes): An unsigned integer that specifies the size of the **pbSignatureBuffer** field in bytes.

signatureOffset (4 bytes): An unsigned integer that specifies the offset of the **pbSignatureBuffer** field relative to the beginning of this structure's parent **DigSigBlob** (section 2.3.2.2); or if the parent is a **WordSigBlob** (section 2.3.2.3), the offset is relative to the beginning of the parent's **cbSigInfo** field.

cbSigningCertStore (4 bytes): An unsigned integer that specifies the size of the **pbSigningCertStoreBuffer** field in bytes.

certStoreOffset (4 bytes): An unsigned integer that specifies the offset of the **pbSigningCertStoreBuffer** field relative to the start of this structure's parent **DigSigBlob** (section 2.3.2.2); or if the parent is a **WordSigBlob** (section 2.3.2.3), the offset is relative to the start of the parent's **cbSigInfo** field.

cbProjectName (4 bytes): An unsigned integer that specifies the count in bytes of the **rgchProjectNameBuffer** field, not including the null-terminating character. MUST be 0x00000000.

projectNameOffset (4 bytes): An unsigned integer that specifies the offset of the **rgchProjectNameBuffer** field relative to the beginning of this structure's parent **DigSigBlob** (section 2.3.2.2); or if the parent is a **WordSigBlob** (section 2.3.2.3), the offset is relative to the beginning of the parent's **cbSigInfo** field.

fTimestamp (4 bytes): This field is reserved and MUST be 0x00000000. MUST ignore on reading.

cbTimestampUrl (4 bytes): An unsigned integer that specifies the count in bytes of the **rgchTimestampBuffer** field, not including the null-terminating character. MUST be 0x00000000.

timestampUrlOffset (4 bytes): An unsigned integer that specifies the offset of the **rgchTimestampBuffer** field relative to the beginning of this structure's parent **DigSigBlob** (section 2.3.2.2); or if the parent is a **WordSigBlob** (section 2.3.2.3), the offset is relative to the beginning of the parent's **cbSigInfo** field.

pbSignatureBuffer (variable): An array of bytes that specifies the **VBA Digital Signature** (section 2.3.2.4) of the VBA project.

pbSigningCertStoreBuffer (variable): A VBASigSerializedCertStore structure (section 2.3.2.5.5) containing the public **digital certificate** information of the certificate used to create the digital signature.

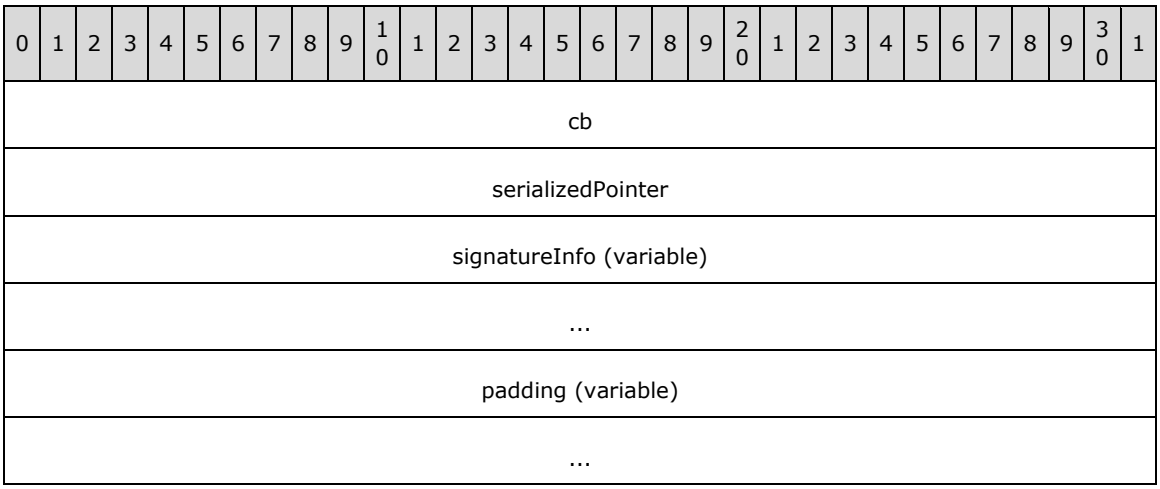
rgchProjectNameBuffer (variable): A null-terminated array of **Unicode** characters. The field is reserved and MUST be a single null Unicode character (0x0000).

rgchTimestampBuffer (variable): A null-terminated array of Unicode characters. The field is reserved and MUST be a single null Unicode character (0x0000).

2.3.2.2 DigSigBlob

Referenced by: VtDigSigValue

Specifies the layout of the **VBA digital signature** data.



cb (4 bytes): An unsigned integer that specifies the size of the **signatureInfo** and **padding** fields combined, in bytes.

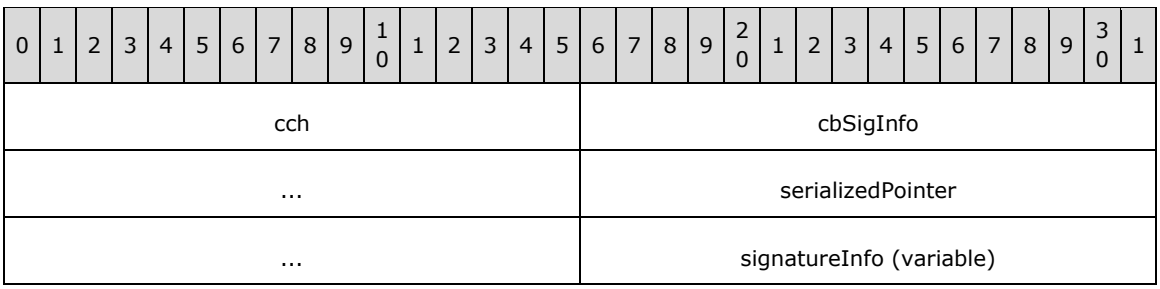
serializedPointer (4 bytes): An unsigned integer that specifies the offset of the **signatureInfo** field within this structure. MUST be 0x00000008.

signatureInfo (variable): A **DigSigInfoSerialized** structure (section 2.3.2.1) containing the data for the signature.

padding (variable): An array of bytes. The size of this array is the number of bytes necessary to pad the size of the **signatureInfo** field to a multiple of 4 bytes. The contents of this field are undefined and MUST be ignored.

2.3.2.3 WordSigBlob

Specifies the layout of the **VBA digital signature** data when it is wrapped as a length-prefixed **Unicode** character string.



...
padding (variable)
...

cch (2 bytes): An unsigned integer that specifies half the count of bytes of the remainder of the structure. MUST be the value given by the following formula.

cch = (**cbSigInfo** + (**cbSigInfo** mod 2) + 8) / 2

cbSigInfo (4 bytes): An unsigned integer that specifies the size of the **signatureInfo** field in bytes.

serializedPointer (4 bytes): An unsigned integer that specifies the offset of the **signatureInfo** field within this structure relative to the **cbSigInfo** field. MUST be 0x00000008.

signatureInfo (variable): A **DigSigInfoSerialized** structure (section 2.3.2.1) containing the data for the signature.

padding (variable): An array of bytes. The size of this array is the number of bytes necessary to pad the entire structure's size to a multiple of 2 bytes. The contents of this field are undefined and MUST be ignored.

2.3.2.4 VBA Digital Signature

The **VBA digital signature** MUST be a PKCS #7 **SignedData** ([PKCS7] section 9.1) structure, constrained as specified in the following sections. The following section uses ASN.1 notation ([ITUX680-1994]) to specify the data format where applicable.

2.3.2.4.1 SignedData Constraints

The **SignedData** structure ([PKCS7] section 9.1) specifying the **digital signature** MUST conform to the following constraints:

- The **digestAlgorithms** field MUST contain only one **digestAlgorithmIdentifier** ([PKCS7] section 6.3), and that **digestAlgorithmIdentifier** MUST specify the identifier of the algorithm used to create the digest of the signature.<10>
- The **contentInfo** field's **contentType** MUST be an **Object Identifier** ([ITUX680-1994] section 3.8.35) with the value "1.3.6.1.4.1.311.2.1.4". The **content** field of the **contentInfo** of this **SignedData** structure MUST be a **SpcIndirectDataContent** structure (section 2.3.2.4.3.1).
- The **certificates** field MUST contain certificates as specified by [PKCS7] section 9.1. This MUST include the signature verification certificate and can contain any intermediate certificates between that end entity and the root, including the root. If the **SignedData** contains a **Countersignature** ([PKCS9] section 6.6), the certificates associated with the **Countersignature** also MUST be contained in the **certificates** field.
- The **crls** field SHOULD be absent. If present, the **crls** field MUST be ignored.
- The **signerInfos** field MUST contain a single **SignerInfo** structure (section 2.3.2.4.2) ([PKCS7] section 9.2).

2.3.2.4.2 SignerInfo Constraints

The **SignerInfo** in the **SignedData** structure (section 2.3.2.4.1) MUST conform to the following constraints:

- The **authenticatedAttributes** (section 2.3.2.4.4) field MUST be present.
- This field MUST contain the following attributes:
  - A **content-type** attribute with its value set as specified in [PKCS7] section 9.2.
  - A **message-digest** attribute with its value set as specified in [PKCS7] section 9.2.
- This field can also contain:
  - One **SpcStatementType** attribute (section 2.3.2.4.4.1), but this attribute MUST be ignored.
  - One **SpcSpOpusInfo** attribute (section 2.3.2.4.4.2).
- If the signature has been timestamped, then the **unauthenticatedAttributes** field (section 2.3.2.4.5) MUST be present and MUST contain a single **Countersignature** attribute ([PKCS9] section 6.6). If the signature has not been timestamped, the **unauthenticatedAttributes** field MUST be absent.

2.3.2.4.3 SignedData contentInfo Structures

The **contentInfo** field of the **SignedData** structure (section 2.3.2.4.1) MUST be a **SpcIndirectDataContent** structure (section 2.3.2.4.3.1) or a **SpcIndirectDataContentV2** structure (section 2.3.2.4.3.2) formatted as specified in the corresponding section. The type of the **contentInfo** field MUST be specified by the consumer of the **DigSigBlob** (section 2.3.2.2) or **WordSigBlob** (section 2.3.2.3) structures.

2.3.2.4.3.1 SpcIndirectDataContent

This structure specifies data about the **digital signature** and contains the **hash** of the data that is to be signed. This structure is specified by the following ASN.1 ([ITUX680-1994]) notation.

```
SpcIndirectDataContent ::= SEQUENCE {
    data                SpcAttributeTypeAndOptionalValue,
    messageDigest       DigestInfo
}

SpcAttributeTypeAndOptionalValue ::= SEQUENCE {
    type                 OBJECT IDENTIFIER,
    value                [0] EXPLICIT ANY OPTIONAL
}

DigestInfo ::= SEQUENCE {
    digestAlgorithm    AlgorithmIdentifier,
    digest             OCTETSTRING
}

AlgorithmIdentifier ::= SEQUENCE {
    algorithm          OBJECT IDENTIFIER,
    parameters         [0] EXPLICIT ANY OPTIONAL
}
```

The **SpcIndirectDataContent** structure's fields MUST be constrained as follows:

- The **data** field MUST be a **SpcAttributeTypeAndOptionalValue** structure.
- The **messageDigest** field MUST be a **DigestInfo** structure.

The **SpcAttributeTypeAndOptionalValue** structure fields MUST be constrained as follows:

- The **type** field MUST be an **Object Identifier** ([ITUX680-1994] section 3.8.35) with the value 1.3.6.1.4.1.311.2.1.29.
- The **value** field SHOULD be set to a zero byte **OCTETSTRING** ([ITUX680-1994] section 20). If the field has any data associated with it, the data MUST be ignored.

The **DigestInfo** structure's fields MUST be constrained as follows:

- The **digestAlgorithm** field MUST be an **AlgorithmIdentifier** structure. The **algorithm** field of **digestAlgorithm** specifies the **Object Identifier** ([ITUX680-1994] section 3.8.35) of the digest algorithm that was used to hash the **VBA** project contents, producing the value for the **digest** field. This **Object Identifier** ([ITUX680-1994] section 3.8.35) value MUST be set to the same algorithm identifier as specified in the **digestAlgorithm** field of the **SignedData** structure (section 2.3.2.4.1). The **parameters** field of **digestAlgorithm** MUST be set to the **Null** type ([ITUX680-1994] section 6.2) with a length of zero.
- The **digest** field MUST be an **OCTETSTRING** ([ITUX680-1994] section 20). The value of the **OCTETSTRING** MUST be produced by means of the hash algorithm specified in [MS-OVBA] section 2.4.2.

2.3.2.4.3.2 SpcIndirectDataContentV2

This structure specifies data about the **digital signature** and contains the **hash** of the data that is to be signed. This structure is specified by the following ASN.1 ([ITUX680-1994]) notation.

```
SpcIndirectDataContentV2 ::= SEQUENCE {
    data               SpcAttributeTypeAndOptionalValue,
    messageDigest      DigestInfo
}

SpcAttributeTypeAndOptionalValue ::= SEQUENCE {
    type                OBJECT IDENTIFIER,
    value               [0] EXPLICIT ANY OPTIONAL
}

DigestInfo ::= SEQUENCE {
    digestAlgorithm    AlgorithmIdentifier,
    digest             OCTETSTRING
}

AlgorithmIdentifier ::= SEQUENCE {
    algorithm          OBJECT IDENTIFIER,
    parameters         [0] EXPLICIT ANY OPTIONAL
}

SigFormatDescriptorV1 ::= SEQUENCE {
    size               INTEGER,
    version            INTEGER,
    format             INTEGER
}

SigDataV1Serialized ::= SEQUENCE {
    algorithmIdSize    INTEGER,
    compiledHashSize   INTEGER,
    sourceHashSize     INTEGER,
    algorithmIdOffset  INTEGER,
    compiledHashOffset INTEGER,
    sourceHashOffset   INTEGER,
    algorithmId        OBJECT IDENTIFIER,
    compiledHash
    sourceHash         OCTETSTRING
}
```

The fields of the **SpcIndirectDataContentV2** structure MUST be constrained as follows:

- The **data** field MUST be a **SpcAttributeTypeAndOptionalValue** structure.
- The **messageDigest** field MUST be a **DigestInfo** structure.

The **SpcAttributeTypeAndOptionalValue** structure fields MUST be constrained as follows:

- The **type** field MUST be an **Object Identifier** ([ITUX680-1994] section 3.8.35) with the value 1.3.6.1.4.1.311.2.1.31.
- The **value** field MUST be an **OCTETSTRING** ([ITUX680-1994] section 20). The value MUST contain the DER encoding ASN.1 data of a **SigFormatDescriptorV1** structure.

The **SigFormatDescriptorV1** structure fields MUST be constrained as follows:

- The **size** field MUST be an **INTEGER** ([ITUX680-1994] section 3.8.29) and the value MUST be equal to the size of the structure.
- The **version** field MUST be an **INTEGER** ([ITUX680-1994] section 3.8.29) and the value MUST be equal to 1.
- The **format** field MUST be an **INTEGER** ([ITUX680-1994] section 3.8.29) and the value MUST be equal to 1.

The **DigestInfo** structure's fields MUST be constrained as follows:

- The **digestAlgorithm** field MUST be an **AlgorithmIdentifier** structure. The **algorithm** field of **digestAlgorithm** specifies the **Object Identifier** ([ITUX680-1994] section 3.8.35) of the digest algorithm that was used to hash the **VBA** project contents, producing the value for the **digest** field. This **Object Identifier** ([ITUX680-1994] section 3.8.35) value MUST be set to the same algorithm identifier as specified in the **digestAlgorithm** field of the **SignedData** structure (section 2.3.2.4.1). The **parameters** field of **digestAlgorithm** MUST be set to the **Null** type ([ITUX680-1994] section 6.2) with a length of zero.
- The **digest** field MUST be an **OCTETSTRING** ([ITUX680-1994] section 20). The value MUST contain the DER encoding ASN.1 data of a **SigDataV1Serialized** structure.

The **SigDataV1Serialized** structure fields MUST be constrained as follows:

- The **algorithmIdSize** field MUST be an **INTEGER** ([ITUX680-1994] section 3.8.29) and the value MUST be equal to the size of the **algorithmId** field.
- The **compiledHashSize** field SHOULD be zero.
- The **sourceHashSize** field MUST be an **INTEGER** ([ITUX680-1994] section 3.8.29) and the value MUST be equal to the size of the **sourceHash** field.
- The **algorithmIdOffset** field MUST be an **INTEGER** ([ITUX680-1994] section 3.8.29) and the value MUST be equal to the position in bytes of the **algorithmId** field from the beginning of the structure.
- The **compiledHashOffset** field SHOULD be zero.
- The **sourceHashOffset** field MUST be an **INTEGER** ([ITUX680-1994] section 3.8.29) and the value MUST be equal to the position in bytes of the **sourceHash** field from the beginning of the structure.
- The **algorithmId** field specifies the **Object Identifier** ([ITUX680-1994] section 3.8.35) of the digest algorithm that was used to hash the VBA project contents, producing the value for the

digest field. This **Object Identifier** ([ITUX680-1994] section 3.8.35) value MUST be set to the same algorithm identifier as specified in the **digestAlgorithm** field of the **SignedData** structure.

- The **compiledHash** field SHOULD be empty.
- The **sourceHash** field MUST be an **OCTETSTRING** ([ITUX680-1994] section 20). The value of the **OCTETSTRING** MUST be produced by means of the hash algorithm specified in [MS-OVBA] section 2.4.2.

2.3.2.4.4 SignerInfo authenticatedAttributes Structures

This section specifies the attribute data that is stored in the **authenticatedAttributes** field of the **SignerInfo** structure (section 2.3.2.4.2).

2.3.2.4.4.1 SpcStatementType

This structure is specified by the following ASN.1 ([ITUX680-1994]) notation.

```
SpcStatementType ::= SEQUENCE of OBJECT IDENTIFIER
```

The **SpcStatementType** MUST contain one **Object Identifier** ([ITUX680-1994] section 3.8.35) with either the value 1.3.6.1.4.1.311.2.1.21 or 1.3.6.1.4.1.311.2.1.22. The **SpcStatementType** MUST be ignored.

2.3.2.4.4.2 SpcSpOpusInfo

This structure is specified by the following ASN.1 ([ITUX680-1994]) notation.

```
SpcSpOpusInfo ::= SEQUENCE {
    programName         [0] EXPLICIT SpcString OPTIONAL,
    moreInfo            [1] EXPLICIT SpcLink OPTIONAL
}
```

The **programName** field MUST be set to a **SpcString** structure (section 2.3.2.4.4.3). The value of the string contained in this structure specifies the description of the project for which the signature was created. The **unicode** field within the **SpcString programName** SHOULD have a value that is a 0 length **BMPSTRING** ([ITUX680-1994] section 34.12). A value other than an empty string MAY<11> be provided.

The **moreInfo** field MUST be set to a **SpcLink** structure (section 2.3.2.4.4.4) if present. If present the value of the **url** field within the **SpcLink** structure specifies a **Uniform Resource Locator (URL)** to associate with the signature.<12>

2.3.2.4.4.3 SpcString

This structure is specified by the following ASN.1 ([ITUX680-1994]) notation.

```
SpcString ::= CHOICE {
    unicode         [0] IMPLICIT BMPSTRING
}
```

The **unicode** field MUST be of the **BMPSTRING** ([ITUX680-1994] section 34.12) type.

2.3.2.4.4.4 SpcLink

This structure is specified by the following ASN.1 ([ITUX680-1994]) notation.

```
SpcLink ::= CHOICE {
    url         [0] IMPLICIT IA5STRING
}
```

The **url** field MUST be of the **IA5STRING** ([ITUX680-1994] section 34.1) type. The structure specifies a **hyperlink** resource by using the **Uniform Resource Locator (URL)**.

2.3.2.4.5 SignerInfo unauthenticatedAttributes

A **Countersignature** located in the **SignerInfo** (section 2.3.2.4.2) **unauthenticatedAttributes** field SHOULD be generated by a trusted third-party **time stamp authority** (TSA). Its purpose is to assert that the **VBA** project signature existed prior to the time specified in the **SigningTime** attribute, specified in the following attributes.

The **Countersignature** used MUST consist of the following attributes in the **unauthenticatedAttributes** field:

- **ContentType** ([PKCS9] section 6.3): The attribute's value MUST be set to PKCS #7 **Data** ([PKCS7] section 8).
- **SigningTime** ([PKCS9] section 6.5): The value MUST be set as specified by [PKCS9] section 6.5.
- **messageDigest** ([PKCS9] section 6.4): The value MUST be set as specified by [PKCS9] section 6.6.

In addition, the certificate ([RFC3280]) whose private key signed the **Countersignature** MUST be added to the **SignedData** (section 2.3.2.4.1) **certificates** field. Intermediate and root certificates of the certificate chain, including the signing certificate, MAY also be added to the **SignedData** (section 2.3.2.4.1) **certificates** field.

2.3.2.4.6 VBA Digital Signature Verification

The **VBA** signature verification employs the steps specified in the following sections. The **SignedData** (section 2.3.2.4) integrity is verified as specified in PKCS #7 ([PKCS7] section 9.3 and [PKCS7] section 9.4).

2.3.2.4.6.1 Certificate Processing

The software publisher's signing certificate and certificate chain MUST be verified against the following criteria:

- The certificate chain MUST be validated to a trusted root certificate by using X.509 path validation rules as specified by [RFC3280] section 6.
- Either the signing certificate MUST contain the extended key usage (EKU) ([RFC3280] section 4.2.1.13) value for code signing (1.3.6.1.5.5.7.3.3) or there MUST be no EKU fields present in the signing certificate.
- The certificate chain MUST be within its validity period. If the certificate chain is not within its validity period, the signature MUST have a **timestamp**, and that timestamp MUST be validated according to the **Timestamp Processing** rules (section 2.3.2.4.6.2). If the signature has a timestamp, the certificate chain MUST be within its validity period at the timestamp time.

If any of these conditions are not met, the signature MUST be treated as invalid.

2.3.2.4.6.2 Timestamp Processing

A signature with a **timestamp** **Countersignature** MUST be considered valid if, and only if, it meets the following criteria:

- The certificate chain for the **Countersignature** MUST be built according to a trusted root certificate by using X.509 path validation rules as specified by [RFC3280] section 6.
- The **time stamp authority** (TSA) certificate that was used to sign the timestamp MUST contain the EKU value for timestamping ("1.3.6.1.5.5.7.3.8").

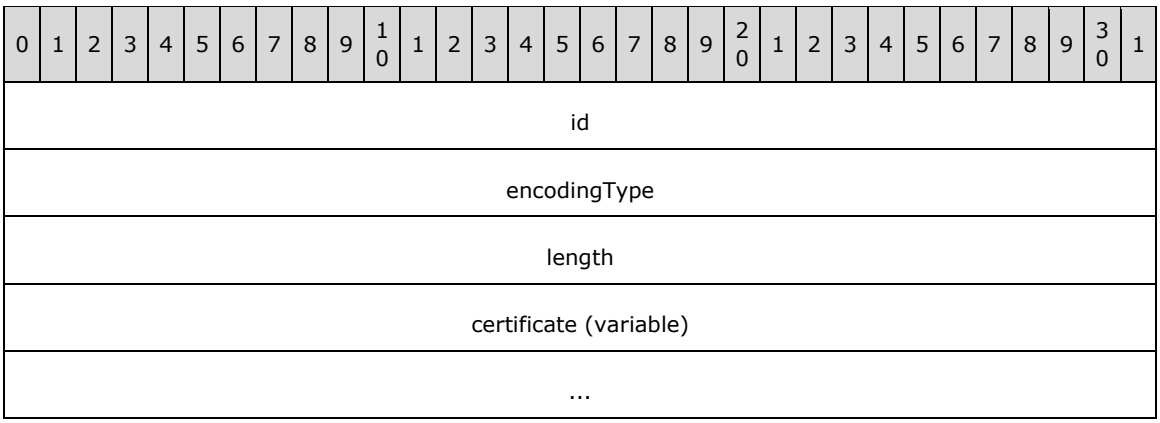
2.3.2.5 Serialized Certificate Store Structure

This section specifies the structures used for storing a **digital certificate**, and optionally a list of properties, in a serialized representation as part of the **digital signature** for a **VBA** project.

2.3.2.5.1 SerializedCertificateEntry

Referenced by: CertStoreCertificateGroup

Specifies a serialized **digital certificate** entry in a serialized **digital certificate store**.



id (4 bytes): An unsigned integer. MUST be 0x00000020.

encodingType (4 bytes): An unsigned integer that MUST be the value 0x00000001, which specifies ASN.1 encoding ([ITUX680-1994]).

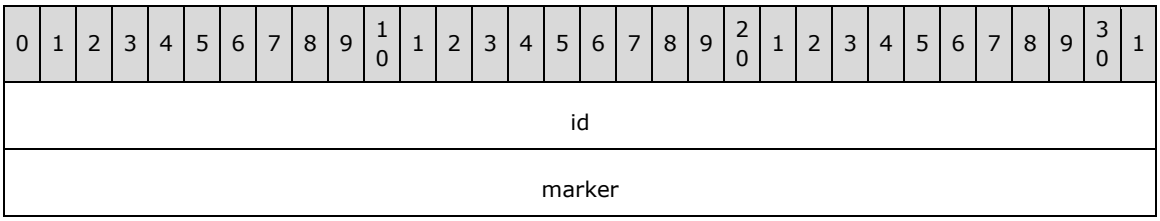
length (4 bytes): An unsigned integer that specifies the count of bytes for the **certificate** field.

certificate (variable): Specifies the certificate data. MUST contain the ASN.1 [ITUX680-1994] DER encoding of an X.509 certificate as specified by [RFC3280].

2.3.2.5.2 EndElementMarkerEntry

Referenced by: DocSigSerializedCertStore, VBASigSerializedCertStore

Specifies a special entry in a serialized **digital certificate store** that marks the end of the store.



...

id (4 bytes): An unsigned integer. MUST be 0x00000000.

marker (8 bytes): A sentinel value 8 bytes in length, the value of which MUST be 0x0000000000000000.

2.3.2.5.3 SerializedPropertyEntry

Referenced by: CertStoreCertificateGroup

Specifies an entry in a serialized **digital certificate store** that contains data for a property associated with a certificate in the store.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
id																															
encodingType																															
length																															
value (variable)																															
...																															

id (4 bytes): An unsigned integer specifying the identifier of the property. MUST be less than or equal to 0x0000FFFF and MUST NOT be the value 0x00000000 or 0x00000020 because these values specify the special entries **SerializedCertificateEntry** (section 2.3.2.5.1) and **EndElementMarkerEntry** (section 2.3.2.5.2).

encodingType (4 bytes): An unsigned integer that MUST be the value 0x00000001, which specifies ASN.1 encoding ([ITUX680-1994]).

length (4 bytes): An unsigned integer that specifies the count of bytes for the **value** field.

value (variable): Specifies the value of the property. This field SHOULD be ignored on read<13>.

2.3.2.5.4 CertStoreCertificateGroup

Referenced by: DocSigSerializedCertStore, VBASigSerializedCertStore

Specifies a grouping of elements in a serialized **digital certificate store** that consists of zero or more properties of a certificate, and the serialized certificate itself.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
elementList (variable)																															
...																															
certificateElement (variable)																															

...

elementList (variable): An array of **SerializedPropertyEntry** (section 2.3.2.5.3). This array can contain zero or more elements. Elements of this array are read and processed until a **SerializedPropertyEntry.id** is read with the unsigned integer value 0x00000020, which specifies the end of this array and the beginning of the **certificateElement** field. The terminating **SerializedPropertyEntry.id** that is read is actually the **certificateElement.id** field and not a part of this field.

certificateElement (variable): A **SerializedCertificateEntry** (section 2.3.2.5.1) specifying a **digital certificate** stored in this digital certificate store.

2.3.2.5.5 VBASigSerializedCertStore

Referenced by: DigSigInfoSerialized

The serialized **digital certificate store** specifies structures for storing a digital certificate store containing a single **digital certificate** and, optionally, a list of properties associated with the certificate.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
version																															
fileType																															
certGroup (variable)																															
...																															
endMarkerElement																															
...																															
...																															

version (4 bytes): An unsigned integer identifying the version of the structure. MUST be 0x00000000.

fileType (4 bytes): An unsigned integer that MUST be the value 0x54524543. This value specifies that the structure is a digital certificate store.

certGroup (variable): A **CertStoreCertificateGroup** (section 2.3.2.5.4) structure that specifies the digital certificate stored in this serialized digital certificate store along with a set of optional properties.

endMarkerElement (12 bytes): An **EndElementMarkerEntry** (section 2.3.2.5.2) structure specifying the end of the structure.

2.3.3 Property Set Storage

An **OLE** property set storage is a format as specified in [MS-OLEPS] that specifies how to store a collection of sets of properties. This section specifies the property sets and data formats used within them that are applicable to this specification. The **Summary Information** (section 2.3.3.2.1)

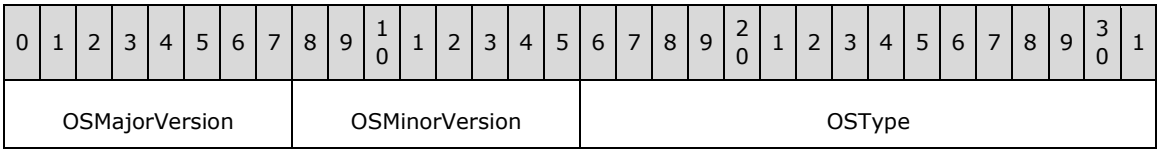
property set SHOULD be written<14>. The **Document Summary Information** (section 2.3.3.2.2) and **User Defined** (section 2.3.3.2.3) property sets are optional.

2.3.3.1 Property Types

This section specifies the data format for property data types that require more detailed specification than what is provided in [MS-OLEPS].

2.3.3.1.1 PropertySetSystemIdentifier

Specifies an operating system type and version for a property set.



OSMajorVersion (1 byte): An unsigned integer specifying the **major version** number of the operating system that wrote the property set.

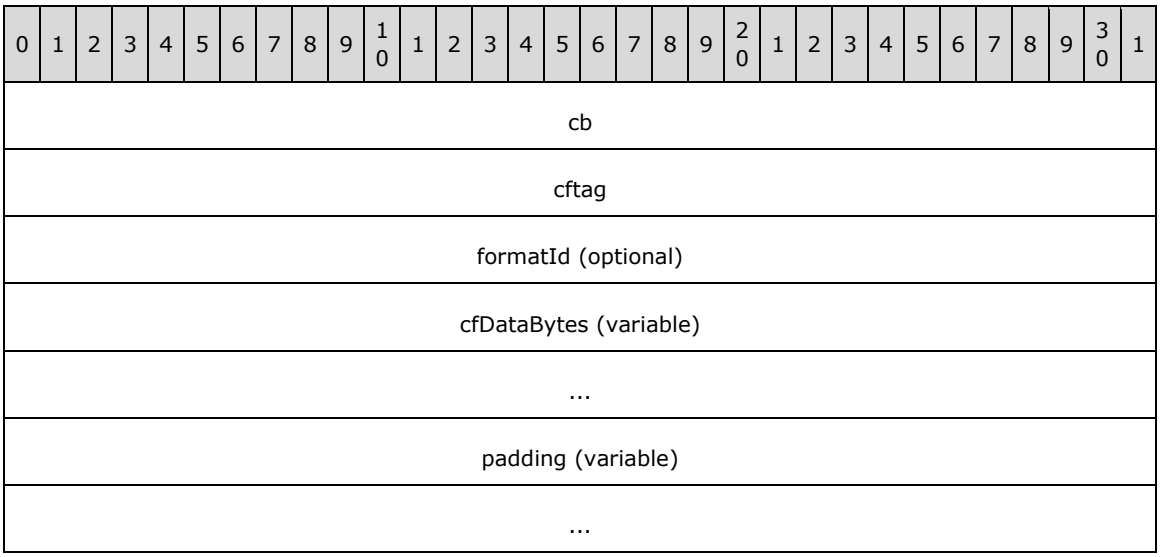
OSMinorVersion (1 byte): An unsigned integer specifying the **minor version** number of the operating system that wrote the property set.

OSType (2 bytes): An unsigned integer that MUST be 0x0002.

2.3.3.1.2 VtThumbnailValue

Referenced by: VtThumbnail

Specifies data for the **thumbnail** property. This type conforms to the **ClipboardData** type as specified in [MS-OLEPS] section 2.11, but this section specifies additional detail applicable to this type.



cb (4 bytes): An unsigned integer that specifies the count of bytes of the remainder of the structure, not including any **padding** field bytes.

cftag (4 bytes): An unsigned integer specifying the **clipboard format** of the data. MUST be 0x00000000, CF_WINDOWS (0xFFFFFFFF), or CF_MACINTOSH (0xFFFFFFFE). A value of 0x00000000 specifies that there is no image data associated with the property.

formatId (4 bytes): An unsigned integer specifying the format of the data stored in the **cfDataBytes** field. MUST NOT exist if the **cftag** field is 0x00000000. MUST exist if **cftag** is not 0x00000000. If it exists, it MUST be one of the following: CF_METAFILEPICT (0x00000003), CF_ENHMETAFILE (0x0000000E), or CF_JPEG (0x00000333).

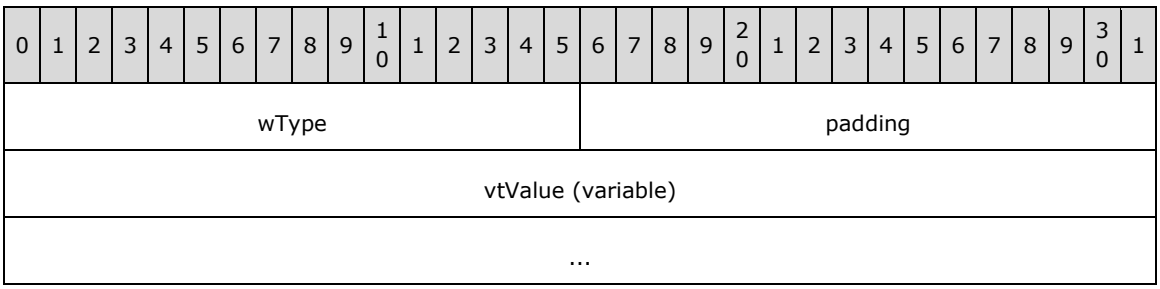
cfDataBytes (variable): An array of bytes containing the data for the thumbnail image. MUST NOT exist if the **cftag** field is 0x00000000. MUST exist if **cftag** is not 0x00000000. If it exists, MUST be (**cb** - 8) bytes in length. The format of the data is specified by **formatId** according to the following table.

formatId Value	Data format
CF_METAFILEPICT (0x00000003)	A **METAFILEPICT** structure as specified by [MS-WMF].
CF_ENHMETAFILE (0x0000000E)	Enhanced metafile image data as specified in [MS-EMF].
CF_JPEG (0x00000333)	**Joint Photographic Experts Group (JPEG)** image data.

padding (variable): An array of bytes. The length of the array MUST be the smallest number of bytes required to pad the size of this structure to a multiple of 4 bytes. The padding SHOULD be 0x00 values, but MAY be undefined values, and MUST be ignored.

2.3.3.1.3 VtThumbnail

Specifies the format for the **thumbnail** property. This type conforms to the VT_CF **TypedPropertyValue** type as specified in [MS-OLEPS] section 2.15, but is presented in additional detail here to specify specific constraints on the type of data that can be contained in this type.



wType (2 bytes): An unsigned integer that MUST be VT_CF (0x0047).

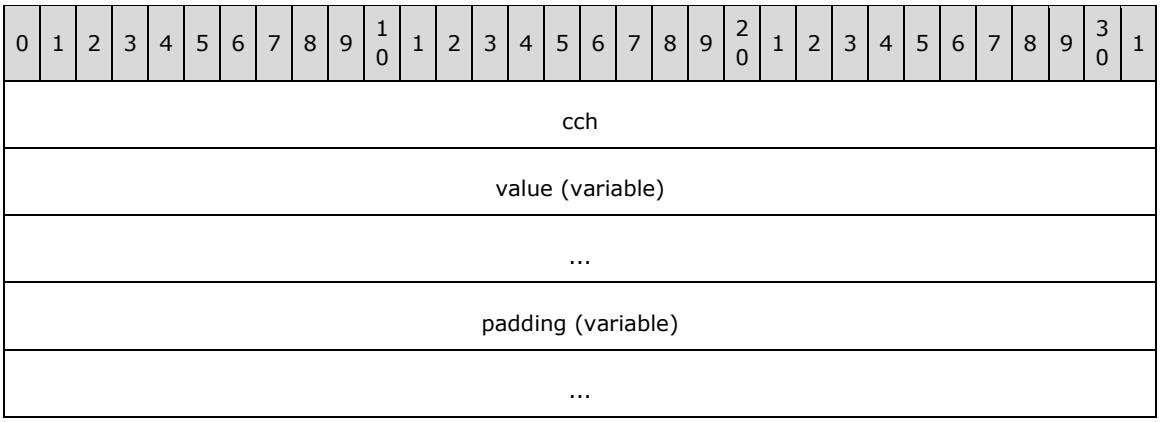
padding (2 bytes): An unsigned integer that MUST be 0x0000. MUST be ignored.

vtValue (variable): MUST be a **VtThumbnailValue** (section 2.3.3.1.2) structure.

2.3.3.1.4 Lpstr

Referenced by: VtString

Specifies data for a null-terminated single-byte character string, the encoding of which corresponds to the value of the enclosing property set's **CodePage** property ([MS-OLEPS] section 2.18.2). This type deviates from the **CodePageString** type specified in [MS-OLEPS] section 2.5 in the way that the **cch** field can be calculated and in the limit to the length of the string.



cch (4 bytes): An unsigned integer specifying the number of single-byte characters in the **value** field. SHOULD be less than or equal to 0x0000FFFF.<15> SHOULD specify the number of characters in the **value** field including the terminating NULL character but not including **padding**. MAY specify the number of characters in both the **value** and **padding** fields.<16>

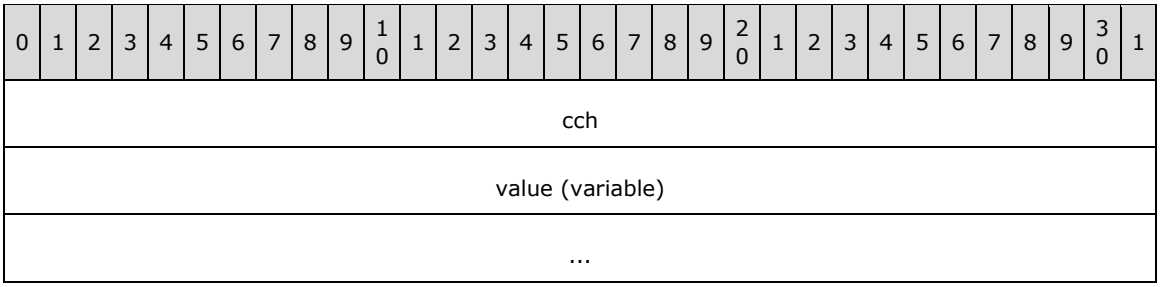
value (variable): A null-terminated array of single-byte characters defining the string.

padding (variable): An array of bytes. The length of the array MUST be the smallest number of bytes required to pad the size of the **value** field to a multiple of 4 bytes. The padding SHOULD be 0x00 values, but MAY be undefined values, and MUST be ignored.<17>

2.3.3.1.5 UnalignedLpstr

Referenced by: VtUnalignedString, VtVecUnalignedLpstrValue

Specifies data for a null-terminated single-byte character string, the encoding of which corresponds to the value of the enclosing property set's **CodePage** property ([MS-OLEPS] section 2.18.2). This type deviates from the **CodePageString** type specified in [MS-OLEPS] section 2.5 in the absence of padding bytes and in the limit to the length of the string.



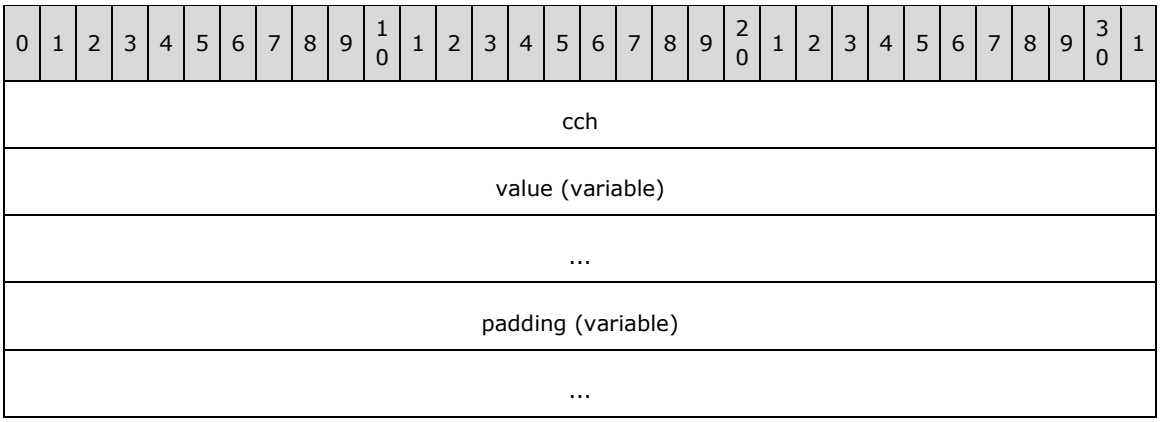
cch (4 bytes): An unsigned integer specifying the number of single-byte characters in the **value** field. SHOULD be less than or equal to 0x0000FFFF.<18>

value (variable): A null-terminated array of single-byte characters defining the string.

2.3.3.1.6 Lpwstr

Referenced by: VtHyperlink, VtString, VtUnalignedString, VtVecLpwstrValue

Specifies data for a null-terminated **Unicode** character string. This type deviates from the **UnicodeString** type specified in [MS-OLEPS] section 2.7 in the limit to the length of the string and in the way the length is specified in characters and not bytes.



cch (4 bytes): An unsigned integer specifying the number of Unicode characters written as the **value** field including the terminating NULL character. SHOULD be less than or equal to 0x0000FFFF.<19>

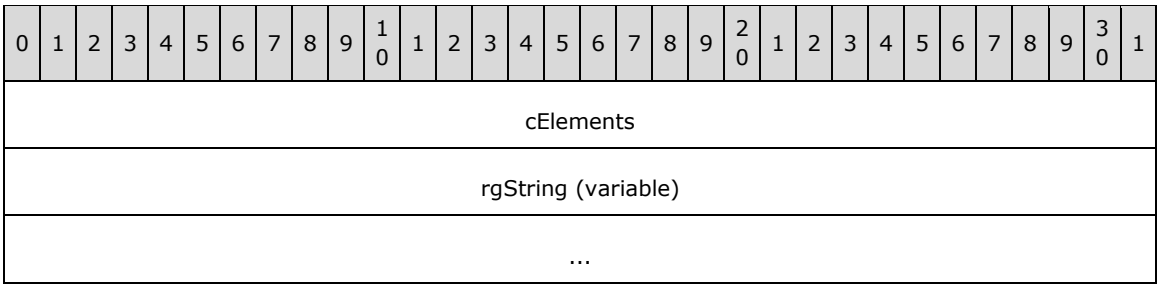
value (variable): A null-terminated array of Unicode characters defining the string.

padding (variable): An array of bytes. The length of the array MUST be the smallest number of bytes required to pad the size of the **value** field to a multiple of 4 bytes. The padding SHOULD be 0x00 values, but MAY be undefined values, and MUST be ignored.<20>

2.3.3.1.7 VtVecLpwstrValue

Referenced by: VtVecLpwstr

Specifies data for a property containing an array of **Unicode** strings. This type conforms to the (VT_VECTOR | VT_LPWSTR) **TypedPropertyValue** value format as specified in [MS-OLEPS] section 2.15, except that the sequence of string structures following the **cElements** field are of type **Lpwstr** (section 2.3.3.1.6).

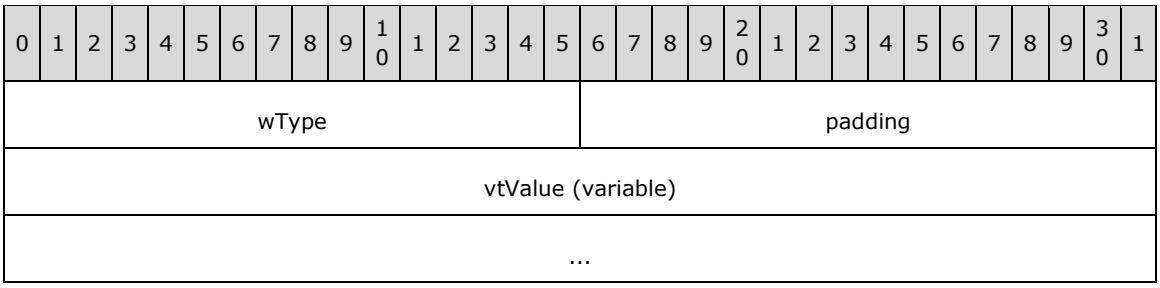


cElements (4 bytes): An unsigned integer specifying the number of elements in **rgString**.

rgString (variable): An array of **Lpwstr** (section 2.3.3.1.6). Specifies the list of values for the property.

2.3.3.1.8 VtVecLpwstr

Specifies the format of a property for which the value is a list of **Unicode** strings. This type conforms to the (VT_VECTOR | VT_LPWSTR) **TypedPropertyValue** type as specified in [MS-OLEPS] section 2.15, except that the format of the strings it contains is as specified in this specification.



wType (2 bytes): An unsigned integer that MUST be equal to VT_VECTOR | VT_LPWSTR (0x101F).

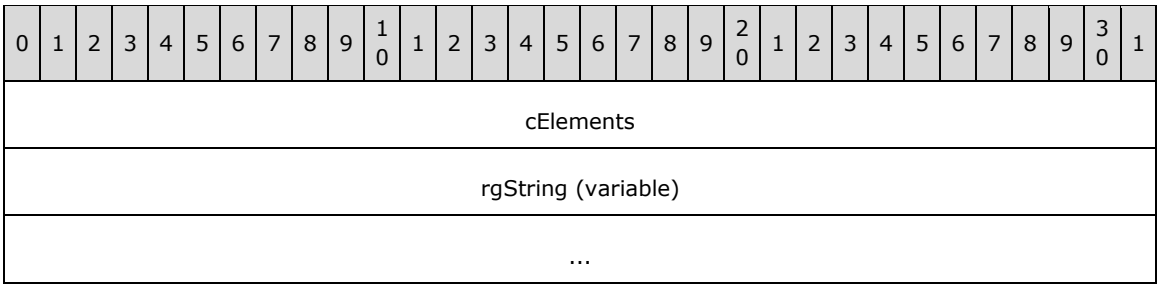
padding (2 bytes): An unsigned integer that MUST be 0x0000. MUST be ignored.

vtValue (variable): MUST be a **VtVecLpwstrValue** (section 2.3.3.1.7) structure.

2.3.3.1.9 VtVecUnalignedLpstrValue

Referenced by: VtVecUnalignedLpstr

Specifies data for a property containing an array of single-byte character strings. This type conforms to the (VT_VECTOR | VT_LPSTR) **TypedPropertyValue** value format as specified in [MS-OLEPS] section 2.15, except that the sequence of string structures following the **cElements** field are of type **UnalignedLpstr** (section 2.3.3.1.5).

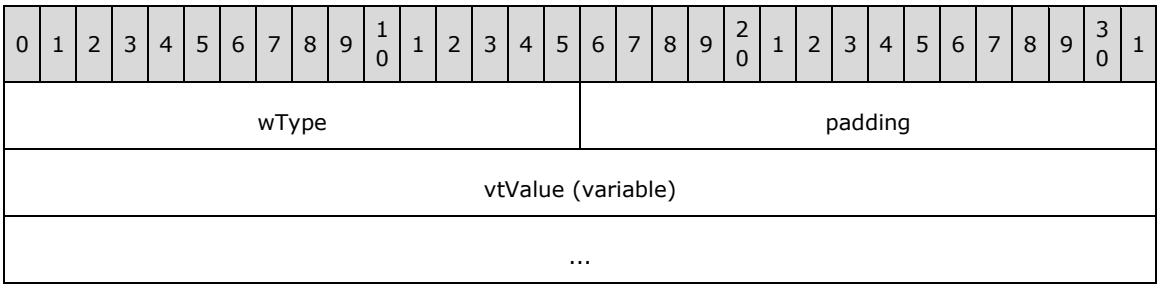


cElements (4 bytes): An unsigned integer specifying the number of elements in **rgString**.

rgString (variable): An array of **UnalignedLpstr** (section 2.3.3.1.5). Specifies the list of values for the property.

2.3.3.1.10 VtVecUnalignedLpstr

Specifies the format of a property for which the value is a list of single-byte character strings. This type conforms to the (VT_VECTOR | VT_LPSTR) **TypedPropertyValue** type as specified in [MS-OLEPS] section 2.15, except that the format of the strings it contains is as specified in this specification.



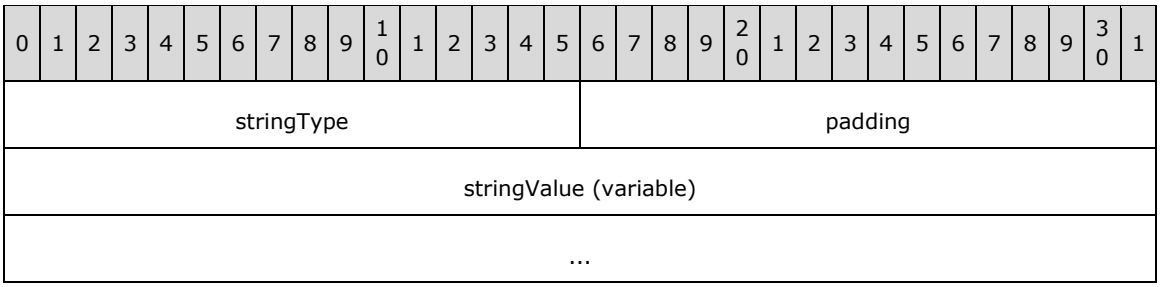
wType (2 bytes): An unsigned integer that MUST be VT_VECTOR | VT_LPSTR (0x101E).

padding (2 bytes): An unsigned integer that MUST be 0x0000. MUST be ignored.

vtValue (variable): MUST be a **VtVecUnalignedLpstrValue** (section 2.3.3.1.9) structure.

2.3.3.1.11 VtString

Specifies the format of a property for which the value is a string.



stringType (2 bytes): An unsigned integer that MUST be VT_LPSTR (0x001E) or VT_LPWSTR (0x001F).

padding (2 bytes): An unsigned integer that MUST be 0x0000. MUST be ignored.

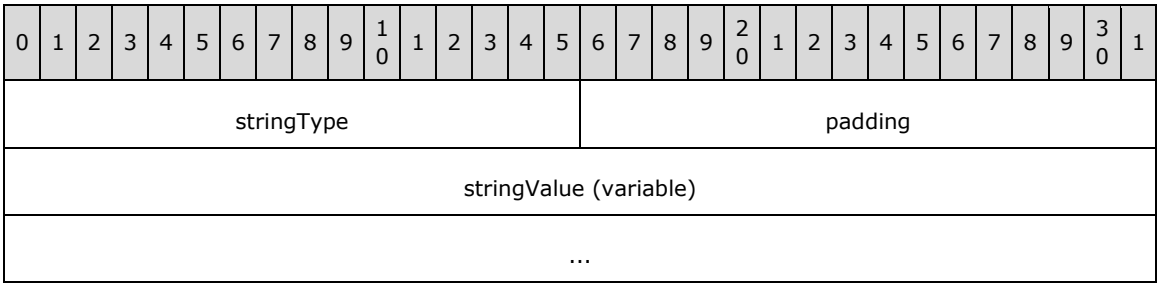
stringValue (variable): A structure that determines its type depending on the value of the **stringType** field according to the following table.

stringType	stringValue type
VT_LPSTR (0x001E)	**Lpstr** (section 2.3.3.1.4)
VT_LPWSTR (0x001F)	**Lpwstr** (section 2.3.3.1.6)

2.3.3.1.12 VtUnalignedString

Referenced by: VtHeadingPair

Specifies the format of a property for which the value is a string but where the **stringValue** field is not padded to any particular length if it is a single-byte character string type.



stringType (2 bytes): An unsigned integer that MUST be VT_LPSTR (0x001E) or VT_LPWSTR (0x001F).

padding (2 bytes): An unsigned integer that MUST be 0x0000. MUST be ignored.

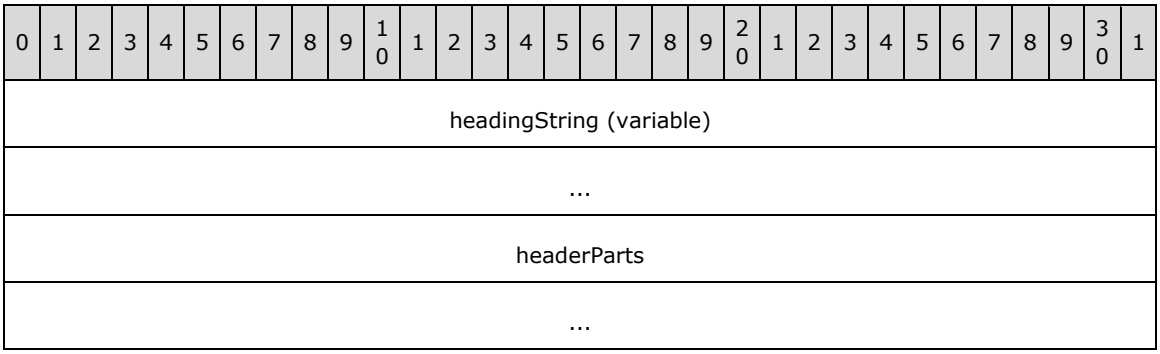
stringValue (variable): A structure that determines its type depending on the value of the **stringType** field according to the following table.

stringType	stringValue type
VT_LPSTR (0x001E)	**UnalignedLpstr** (section 2.3.3.1.5)
VT_LPWSTR (0x001F)	**Lpwstr** (section 2.3.3.1.6)

2.3.3.1.13 VtHeadingPair

Referenced by: VtVecHeadingPairValue

Specifies data for a heading pair for a property.



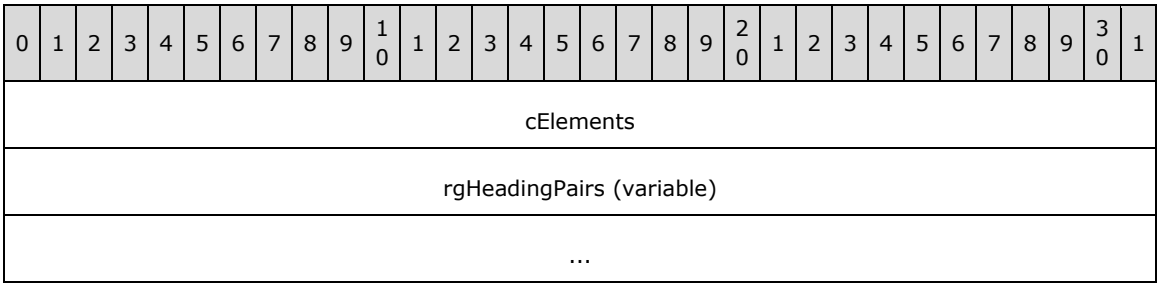
headingString (variable): A structure of type **VtUnalignedString** (section 2.3.3.1.12) specifying the header string.

headerParts (8 bytes): A VT_I4 **TypedPropertyValue** structure as specified in [MS-OLEPS] section 2.15. The **Value** field of the **TypedPropertyValue** structure specifies the number of document parts associated with this header.

2.3.3.1.14 VtVecHeadingPairValue

Referenced by: VtVecHeadingPair

Specifies data for the heading pair property. This type conforms to the (VT_VECTOR | VT_VARIANT) **TypedPropertyValue** value format as specified in [MS-OLEPS] section 2.15, but this section specifies additional detail specifically applicable to this type.



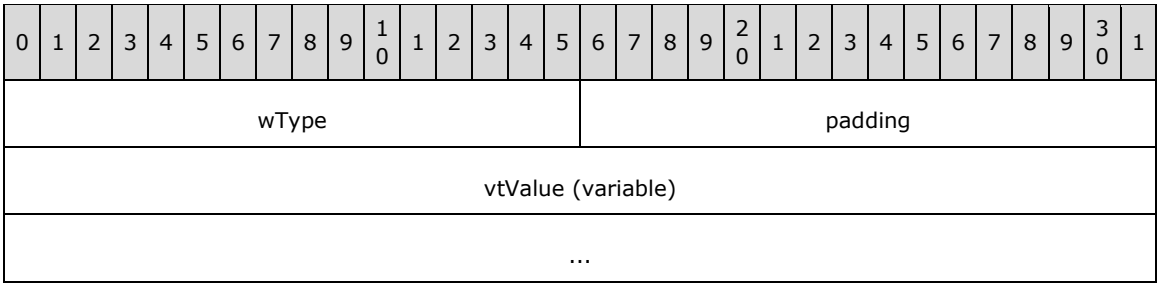
cElements (4 bytes): An unsigned integer specifying the count of elements in the **rgHeadingPairs** field. The number of elements in **rgHeadingPairs** MUST be half of this value. This value MUST be even.

rgHeadingPairs (variable): An array of **VtHeadingPair** (section 2.3.3.1.13). Specifies the list of heading pairs for the property.

2.3.3.1.15 VtVecHeadingPair

Specifies the format for a property representing the headers corresponding to the **GKPIDDSI_DOCPARTS** property data (section 2.3.3.2.2.1). This type conforms to the (VT_VECTOR | VT_VARIANT) **TypedPropertyValue** type as specified in [MS-OLEPS] section 2.15, but is presented in additional detail here to specify specific constraints on the type and format of data that can be contained in this type.

Refer to the **GKPIDDSI_HEADINGPAIR** (section 2.3.3.2.2.1) and **GKPIDDSI_DOCPARTS** (section 2.3.3.2.2.1) properties for additional details.



wType (2 bytes): An unsigned integer that MUST be VT_VECTOR | VT_VARIANT (0x100C).

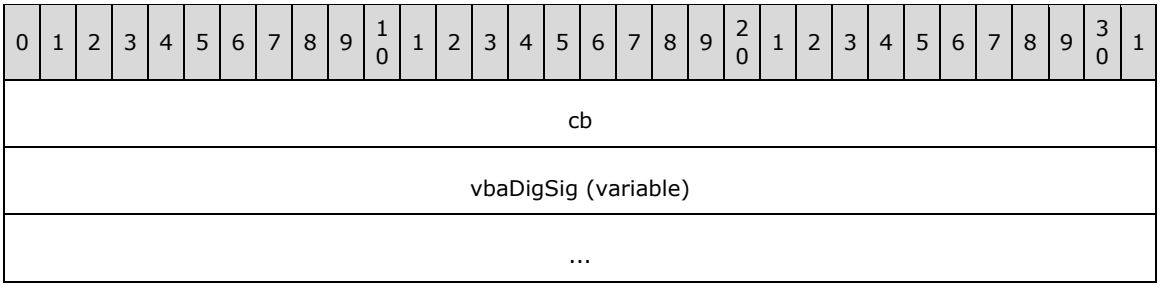
padding (2 bytes): An unsigned integer that MUST be 0x0000. MUST be ignored.

vtValue (variable): MUST be a **VtVecHeadingPairValue** (section 2.3.3.1.14) structure.

2.3.3.1.16 VtDigSigValue

Referenced by: VtDigSig

Specifies the data for a **VBA digital signature** property. This type conforms to the VT_BLOB **TypedPropertyValue** value format as specified in [MS-OLEPS] section 2.15, but this section specifies additional detail applicable to this type.

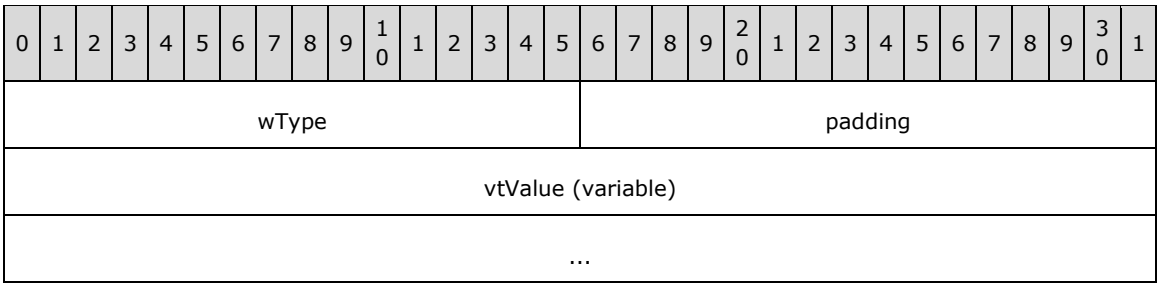


cb (4 bytes): An unsigned integer that specifies the size in bytes of **vbaDigSig**. MUST be equal to **vbaDigSig.cb** + 8.

vbaDigSig (variable): A **DigSigBlob** (section 2.3.2.2) structure that specifies a VBA digital signature.

2.3.3.1.17 VtDigSig

Specifies the format for a property representing a **VBA digital signature**. This type conforms to the VT_BLOB **TypedPropertyValue** type as specified in [MS-OLEPS] section 2.15, but is presented in additional detail here to further specify the content of the property.



wType (2 bytes): An unsigned integer that MUST be equal to VT_BLOB (0x0041).

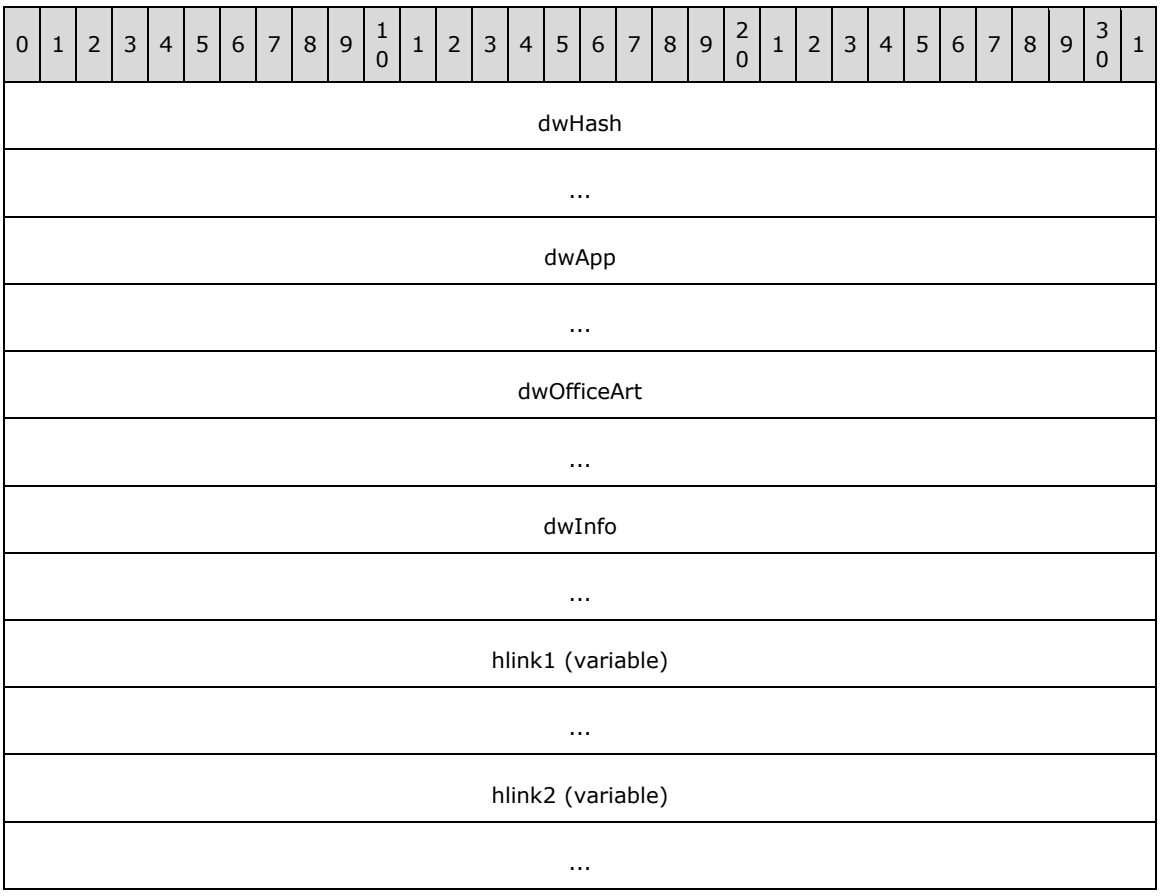
padding (2 bytes): An unsigned integer that MUST be 0x0000. MUST be ignored.

vtValue (variable): MUST be a **VtDigSigValue** (section 2.3.3.1.16) structure.

2.3.3.1.18 VtHyperlink

Referenced by: VecVtHyperlink

Specifies the data format for a **hyperlink** for a property.



dwHash (8 bytes): MUST be a VT_I4 **TypedPropertyValue** as specified in [MS-OLEPS] section 2.15. The **Value** field of this structure SHOULD be calculated as specified in the Hyperlink Hash (section 2.4.2) section with the **hlink1** field and **hlink2** field string values given as input.<21>

dwApp (8 bytes): MUST be a VT_I4 **TypedPropertyValue** as specified in [MS-OLEPS] section 2.15. The **Value** field of this structure is implementation specific.<22>

dwOfficeArt (8 bytes): MUST be a VT_I4 **TypedPropertyValue** as specified in [MS-OLEPS] section 2.15. The **Value** field of this structure MUST be a **MSOSPID** type value ([MS-ODRAW] section 2.1.2) specifying the identifier of the shape ([MS-ODRAW] section 2.2.31) to which this hyperlink applies in the document. If this hyperlink does not apply to a shape, the **Value** field of this structure MUST be 0x00000000.

dwInfo (8 bytes): MUST be a VT_I4 **TypedPropertyValue** as specified in [MS-OLEPS] section 2.15. The **Value** field of this structure is implementation specific.<23> The high-order 2-byte integer of the **Value** field of this structure SHOULD be 0x0000. The unsigned 2-byte integer specified by the high two bytes of this value can be written as 0x0000 by an application that saves the entire file, as it specifies that the hyperlink is in sync with the document contents. If a process changes the hyperlink property element without a corresponding change to the related document content such that the hyperlink needs to be fixed in the document the next time the document is loaded, this value needs to be changed to 0x0001 by the process. If such a process wishes to specify that the hyperlink is to be removed from the document the next time the document is loaded, this value needs to be changed to 0x0002 by the process.

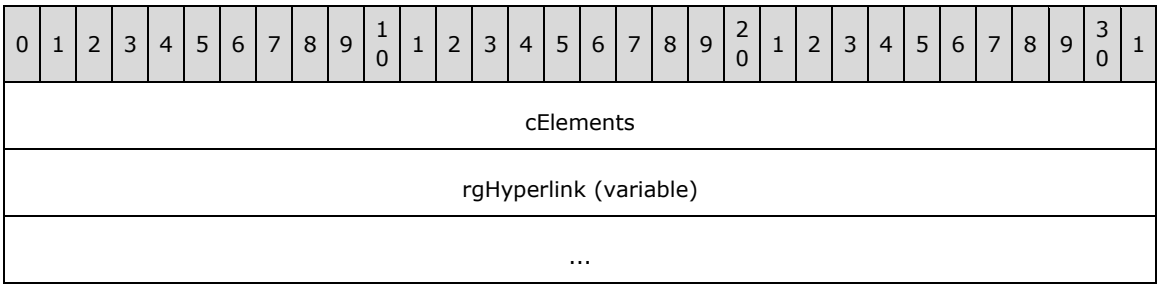
hlink1 (variable): MUST be a **VtString** structure (section 2.3.3.1.11) with **hlink1.wType** equal to VT_LPWSTR. **hlink1.stringValue** specifies the **hyperlink target**.

hlink2 (variable): MUST be a **VtString** structure (section 2.3.3.1.11) with **hlink2.wType** equal to VT_LPWSTR. **hlink2.stringValue** specifies the **hyperlink location**.

2.3.3.1.19 VecVtHyperlink

Referenced by: VtHyperlinkValue

Specifies the data format for an array of **hyperlinks**. This type conforms to the (VT_VECTOR | VT_VARIANT) **TypedPropertyValue** value format as specified in [MS-OLEPS] section 2.15, but this section specifies additional details specific to this type.



cElements (4 bytes): An unsigned integer specifying the count of elements in the **rgHyperlink** field. The number of elements in **rgHyperlink** MUST be 1/6 of this value. This value MUST be evenly divisible by 6.

rgHyperlink (variable): An array of **VtHyperlink** (section 2.3.3.1.18). Specifies the list of hyperlinks for the property.

2.3.3.1.20 VtHyperlinkValue

Referenced by: VtHyperlinks

Specifies the data for a **hyperlinks** property. This type conforms to the VT_BLOB **TypedPropertyValue** value format as specified in [MS-OLEPS] section 2.15, but this section specifies additional detail applicable to this type.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
cbData																															
vecHyperlink (variable)																															
...																															

cbData (4 bytes): An unsigned integer that specifies the size in bytes of **vecHyperlink**.

vecHyperlink (variable): MUST be a **VecVtHyperlink** structure (section 2.3.3.1.19).

2.3.3.1.21 VtHyperlinks

Specifies the format for a property representing a set of **hyperlinks**. This type conforms to the VT_BLOB **TypedPropertyValue** type as specified in [MS-OLEPS] section 2.15, but is presented in more detail here to further specify the contents of the data in this type.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
wType																padding															
vtValue (variable)																															
...																															

wType (2 bytes): An unsigned integer that MUST be equal to VT_BLOB (0x0041).

padding (2 bytes): An unsigned integer that MUST be 0x0000. MUST be ignored.

vtValue (variable): MUST be a **VtHyperlinkValue** structure (section 2.3.3.1.20).

2.3.3.2 OLE Property Sets

This section defines the format of a property set storage **stream** as specified in [MS-OLEPS]. This format applies to the **Summary Information** (section 2.3.3.2.1) and **Document Summary Information** (section 2.3.3.2.2) streams.

2.3.3.2.1 Summary Information Property Set

This section defines a simple **OLE** property set as specified in [MS-OLEPS] that conforms to the well-known property set format SummaryInformation as specified in [MS-OLEPS] section 2.23 and [MS-OLEPS] section 2.25.1. This specifies that the property set has the identifier **GUID** FMTID_SummaryInformation and is contained in the document **stream** named "\005SummaryInformation". It contains a fixed set of recognized properties, which are generally specified in the [MS-OLEPS] specification, but are specified in the following section with more details regarding their use or deviations from the simple OLE property set format pursuant to this specification. The format of this property set is as specified in [MS-OLEPS], except where stated otherwise in the following section. In addition, this property set MUST have the following:

- This property set MUST have its **PropertySetStream** structure **Version** field set to 0x00000000.
- This property set MUST have its **PropertySetStream** structure **CLSID** field set to **GUID_NULL**.

- This property set MUST have a **PropertySetSystemIdentifier** structure (section 2.3.3.1.1) for its **PropertySetStream** structure **SystemIdentifier** field.

For additional information about the OLE property set storage format, see [MS-OLEPS].

2.3.3.2.1.1 PIDSI

Specifies the list of properties that can be defined in the **Summary Information property set** (section 2.3.3.2.1). This set MUST NOT contain a **Dictionary** property ([MS-OLEPS] section 2.18.1). All properties are optional, except the **GKPIDSI_CODEPAGE** property which MUST be written if the **Summary Information property set** (section 2.3.3.2.1) is present. All persisted string type properties (**Lpstr** (section 2.3.3.1.4), **UnalignedLpstr** (section 2.3.3.1.5), and **Lpwstr** (section 2.3.3.1.6)) MUST share the same **code page** encoding as specified by the **GKPIDSI_CODEPAGE** property. If the string properties are persisted as **Lpwstr** (section 2.3.3.1.6) type properties, the **GKPIDSI_CODEPAGE** property MUST be set to CP_WINUNICODE (0x04B0). No order in persistence of the properties is enforced. All properties are persisted as **TypedPropertyValue** types as specified in [MS-OLEPS] section 2.15, except where indicated otherwise for the purpose of additional clarification or deviation from the specification.

The **Value** column of the following table specifies the value of the **PropertyIdentifier** field for the **PropertyIdentifierAndOffset** structure ([MS-OLEPS] section 2.19) for the specified property.

The **Meaning** column of the following table specifies the property data type and the property's meaning. The property data type specifies the format of the **Property** field of the **PropertySet** structure ([MS-OLEPS] section 2.20) for the given property.

Name	Value	Meaning
GKPIDSI_CODEPAGE	0x00000001	MUST be a VT_I2 **TypedPropertyValue** [MS-OLEPS] section 2.15) property. MUST conform to the **CodePage** property requirements as specified in [MS-OLEPS] section 2.18.2.
GKPIDSI_TITLE	0x00000002	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the document title.
GKPIDSI_SUBJECT	0x00000003	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the document subject.
GKPIDSI_AUTHOR	0x00000004	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the document author.
GKPIDSI_KEYWORDS	0x00000005	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the document keywords.
GKPIDSI_COMMENTS	0x00000006	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies comments corresponding to the document.
GKPIDSI_TEMPLATE	0x00000007	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the name of the template on which the document is based.
GKPIDSI_LASTAUTHOR	0x00000008	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the name of the author who last modified the document.
GKPIDSI_REVNUMBER	0x00000009	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the revision number for the document. MUST be the string representation of a non-negative whole number.
GKPIDSI_EDITTIME	0x0000000A	MUST be a VT_FILETIME **TypedPropertyValue** ([MS-OLEPS]

Name	Value	Meaning
		section 2.15) property. The value of the property does not specify a date and time as would be typical of the **FILETIME** data type ([MS-DTYP] section 2.3.3), but instead specifies duration. The value specifies the total time that the document has been opened for edit (in hundreds of nanoseconds).
GKPIDSI_LASTPRINTED	0x0000000B	MUST be a VT_FILETIME **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The value of the property specifies the date and time the document was last printed, in **UTC** time.
GKPIDSI_CREATE_DTM	0x0000000C	MUST be a VT_FILETIME **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The value of the property specifies the date and time the document was created, in UTC time.
GKPIDSI_LASTSAVE_DTM	0x0000000D	MUST be a VT_FILETIME **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The value of the property specifies the date and time the document was last saved, in UTC time.
GKPIDSI_PAGECOUNT	0x0000000E	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The integer value of the property specifies the page count of the document. MAY be ignored.<24>
GKPIDSI_WORDCOUNT	0x0000000F	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The integer value of the property specifies an estimate or an exact count of the words in the document.<25> MAY be ignored. <26>
GKPIDSI_CHARCOUNT	0x00000010	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The integer value of the property specifies an estimate of the character count of the document. MAY be ignored.<27>
GKPIDSI_THUMBNAIL	0x00000011	MUST be a **VtThumbnail** property (section 2.3.3.1.3). **VtThumbnail.vtValue.cfDataBytes** specifies the data that represents the **thumbnail** image for the document.
GKPIDSI_APPNAME	0x00000012	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the name of the application that produced the document.
GKPIDSI_DOC_SECURITY	0x00000013	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The value of the property specifies a bit field that details security properties of the document as follows: ▪ If the value is equal to SECURITY_NONE (0x00000000), there are no security states specified by the property. ▪ If the SECURITY_PASSWORD bit is set (0x00000001), the document MUST be password protected. ▪ If the SECURITY_READONLYRECOMMEND bit is set (0x00000002), it specifies that opening the document read-only is recommended but not enforced. ▪ If the SECURITY_READONLYENFORCED bit is set (0x00000004), it specifies that the document is opened read-only. ▪ If the SECURITY_LOCKED bit is set (0x00000008), it specifies that the document is opened read-only except for annotations.

2.3.3.2.2 Document Summary Information Property Set

This section defines a simple **OLE** property set as specified in [MS-OLEPS] containing application-defined properties. This property set conforms to [MS-OLEPS] section 2.21 and [MS-OLEPS] section 2.23 with regards to the **stream** name requirements and format identifier **GUID** for the FMTID_DocSummaryInformation property set. The Document Summary Information property set is contained in the document stream named **"\005DocumentSummaryInformation"** and contains a fixed set of recognized properties, which are specified in the following section. In addition, this property set MUST have the following:

- This property set MUST have its **PropertySetStream** structure **Version** field set to 0x00000000.
- This property set MUST have its **PropertySetStream** structure **CLSID** field set to **GUID_NULL**.
- This property set MUST have a **PropertySetSystemIdentifier** structure (section 2.3.3.1.1) for its **PropertySetStream** structure **SystemIdentifier** field.

The total size of this property set's **PropertySet** structure ([MS-OLEPS] section 2.20) MUST be padded to a multiple of 4 bytes. The padding, if needed, MUST be located after the final property value of the property set and MUST be the minimum size required to produce a **PropertySet** structure with a size that is a multiple of 4 bytes. The contents of the padding are undefined and MUST be ignored. The **PropertySet** structure's **Size** field MUST include the count of padding bytes in its calculation.

For additional information about the OLE property set storage format, see [MS-OLEPS].

2.3.3.2.2.1 PIDDSI

Specifies the list of properties that can be defined in the **Document Summary Information property set** (section 2.3.3.2.2). This set MUST NOT contain a **Dictionary** property ([MS-OLEPS] section 2.18.1). All properties are optional, except the **GKPIDDSI_CODEPAGE** property (section 2.3.3.2.2.1) which MUST be written if the **Document Summary Information property set** (section 2.3.3.2.2) is present. All persisted string type properties (**Lpstr** (section 2.3.3.1.4), **UnalignedLpstr** (section 2.3.3.1.5), and **Lpwstr** (section 2.3.3.1.6)) MUST share the same **code page** encoding as specified by the **GKPIDDSI_CODEPAGE** property. If the string properties are persisted as **Lpwstr** (section 2.3.3.1.6) type properties, the **GKPIDDSI_CODEPAGE** property MUST be set to CP_WINUNICODE (0x04B0). No order in persistence of properties is enforced. All properties are persisted as **TypedPropertyValue** types as specified in [MS-OLEPS] section 2.15, except where indicated otherwise for the purpose of additional clarification or deviation from the specification.

The **Value** column of the following table specifies the value of the **PropertyIdentifier** field for the **PropertyIdentifierAndOffset** structure ([MS-OLEPS] section 2.19) for the specified property.

The **Meaning** column of the following table specifies the property type and description. The type specifies the format of the **Property** field of the **PropertySet** structure ([MS-OLEPS] section 2.20) for the given property.

Name	Value	Meaning
GKPIDDSI_CODEPAGE	0x00000001	MUST be a VT_I2 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. MUST conform to the **CodePage** property requirements as specified in [MS-OLEPS] section 2.18.2.
GKPIDDSI_CATEGORY	0x00000002	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies a category name for the document.
GKPIDDSI_PRESFORMAT	0x00000003	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the presentation format

Name	Value	Meaning
		type of the document. MAY be ignored.<28> MUST be one of the following values: ▪ Empty string ▪ On-screen Show<29> ▪ On-screen Show (4:3)<30> ▪ Letter Paper (8.5x11 in) ▪ Ledger Paper (11x17 in) ▪ A3 Paper (297x420 mm) ▪ A4 Paper (210x297 mm) ▪ B4 (ISO) Paper (250x353 mm) ▪ B5 (ISO) Paper (176x250 mm) ▪ B4 (JIS) Paper (257x364 mm) ▪ B5 (JIS) Paper (182x257 mm) ▪ Hagaki Card (100x148 mm) ▪ 35mm Slides ▪ Overhead ▪ Banner ▪ Custom ▪ On-screen Show (16:9)<31> ▪ On-screen Show (16:10)<32>
GKPIDDSI_BYTECOUNT	0x00000004	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The integer value of the property specifies an estimate of the size of the document in bytes. MAY be ignored.<33>
GKPIDDSI_LINECOUNT	0x00000005	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The integer value of the property specifies an estimate of the number of text lines in the document. MAY be ignored.<34>
GKPIDDSI_PARACOUNT	0x00000006	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The integer value of the property specifies either an estimate or an exact count of the number of paragraphs in the document.<35> MAY be ignored.<36>
GKPIDDSI_SLIDECOUNT	0x00000007	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The integer value of the property specifies the number of slides in the document. MAY be ignored.<37>
GKPIDDSI_NOTECOUNT	0x00000008	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The integer value of the property specifies the number of notes in the document. MAY be

Name	Value	Meaning
		ignored.<38>
GKPIDDSI_HIDDENCOUNT	0x00000009	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The integer value of the property specifies the number of hidden slides in the document. MAY be ignored.<39>
GKPIDDSI_MMCLIPCOUNT	0x0000000A	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The integer value of the property specifies the number of multimedia clips in the document. MAY be ignored.<40>
GKPIDDSI_SCALE	0x0000000B	MUST be a VT_BOOL **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The value of the property MUST be FALSE.
GKPIDDSI_HEADINGPAIR	0x0000000C	MUST be a **VtVecHeadingPair** property (section 2.3.3.1.15). Each **VtHeadingPair** element (section 2.3.3.1.13) in **VtVecHeadingPair.vtValue.rgHeadingPairs** defines a heading string and a count of document parts as found in the **GKPIDDSI_DOCPARTS** property (section 2.3.3.2.2.1) to which this heading applies. The total sum of document counts for all headers in this property MUST be equal to the number of elements in the **GKPIDDSI_DOCPARTS** property (section 2.3.3.2.2.1) property.
GKPIDDSI_DOCPARTS	0x0000000D	MUST be a **VtVecUnalignedLpstr** (section 2.3.3.1.10) or **VtVecLpwstr** property (section 2.3.3.1.8). Each string element of the vector specifies a part of the document. The elements of this vector are ordered according to the header they belong to as defined in the **GKPIDDSI_HEADINGPAIR** property (section 2.3.3.2.2.1). Example: The first element of the heading pair vector indicates that it has four document parts associated with it. Elements 1 to 4 of the document parts vector are grouped under this header. The next element of the heading pair vector indicates that it has three document parts associated with it. The document part vector elements 5 to 7 are grouped under this header, and so on.
GKPIDDSI_MANAGER	0x0000000E	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the manager associated with the document.
GKPIDDSI_COMPANY	0x0000000F	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the company associated with the document's authoring.
GKPIDDSI_LINKSDIRTY	0x00000010	MUST be a VT_BOOL **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The property value specifies TRUE (any value other than 0x00000000) if any of the values for the linked properties in the **User Defined Property Set** (section 2.3.3.2.3) have changed outside of the application, which would require the application to update the linked fields on document load.<41>
GKPIDDSI_CCHWITHSPACES	0x00000011	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The integer value of the property

Name	Value	Meaning
		specifies an estimate of the number of characters in the document, including **whitespace**. MAY be ignored.<42>
GKPIDDSI_SHAREDDOC	0x00000013	MUST be a VT_BOOL **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The property value MUST be FALSE (0x00000000).
GKPIDDSI_LINKBASE	0x00000014	MUST NOT be written. The **base URL** property is persisted to the **User Defined Property Set** (section 2.3.3.2.3) with the **_PID_LINKBASE** property name.
GKPIDDSI_HLINKS	0x00000015	MUST NOT be written. The **hyperlinks** property is persisted to the **User Defined Property Set** (section 2.3.3.2.3) with the **_PID_HLINKS** property name.
GKPIDDSI_HYPERLINKSCHANGED	0x00000016	MUST be a VT_BOOL **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The property value specifies TRUE (any value other than 0x00000000) if the **_PID_HLINKS** property in the **User Defined Property Set** (section 2.3.3.2.3) has changed outside of the application, which would require the application to update the hyperlink on document load.<43>
GKPIDDSI_VERSION	0x00000017	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. Specifies the version of the application that wrote the property set storage. The two high-order bytes specify an unsigned integer specifying the **major version** number. The two low-order bytes specify an unsigned integer specifying the **minor version** number. The value MUST have the major version number set to a nonzero value, and the minor version number SHOULD be 0x0000. The minor version number MAY be set to the minor version number of the application that wrote the property set storage.<44>
GKPIDDSI_DIGSIG	0x00000018	MUST be a **VtDigSig** property (section 2.3.3.1.17). **VtDigSig.vtValue** specifies the data of the **VBA digital signature** for the VBA project embedded in the document. MUST NOT exist if the VBA project of the document does not have a digital signature or if the project is absent. MAY be ignored.<45>
GKPIDDSI_CONTENTTYPE	0x0000001A	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the content type of the file. MAY be absent.<46>
GKPIDDSI_CONTENTSTATUS	0x0000001B	MUST be a **VtString** property (section 2.3.3.1.11). **VtString.stringValue** specifies the document status. MAY be absent.<47>
GKPIDDSI_LANGUAGE	0x0000001C	MUST be a **VtString** property (section 2.3.3.1.11). SHOULD be absent.<48>
GKPIDDSI_DOCVERSION	0x0000001D	MUST be a **VtString** property (section 2.3.3.1.11). SHOULD be absent.<49>

2.3.3.2.3 User Defined Property Set

This section defines a simple **OLE** property set, as specified in [MS-OLEPS], containing arbitrary user-defined properties. This property set conforms to [MS-OLEPS] section 2.21 and [MS-OLEPS] section 2.23 with regards to **stream** name requirements and format identifier **GUID** for the FMTID_UserDefinedProperties property set. The User Defined property set is contained in the

document stream named **"\005DocumentSummaryInformation"**. It contains properties of arbitrary type and value (within the constraints specified in section 2.3.3.2.3.1), and assigned any property name that meets the [MS-OLEPS] specification except for those names that are reserved as specified in section 2.3.3.2.3.2. This property set also supports the concept of linked properties that use two property entries, one of which does not have an associated property name. The specification for these properties is specified in section 2.3.3.2.3.3. Properties stored in this property set have an effective **PropertyIdentifier** maximum of 0x00FFFFFF because of reserved high-order bits for property entry link specification. In addition, this property set MUST have the following:

- This property set MUST have its **PropertySetStream** structure **Version** field set to 0x00000000.
- This property set MUST have its **PropertySetStream** structure **CLSID** field set to **GUID_NULL**.
- This property set MUST have a **PropertySetSystemIdentifier** structure (section 2.3.3.1.1)for its **PropertySetStream** structure **SystemIdentifier** field.

The total size of this property set's **PropertySet** structure ([MS-OLEPS] section 2.20) MUST be padded to a multiple of 4 bytes. The padding, if needed, MUST be located after the final property value of the property set and MUST be the minimum size required to produce a **PropertySet** structure with a size that is a multiple of 4 bytes. The contents of the padding are undefined and MUST be ignored. The **PropertySet** structure's **Size** field MUST include the count of padding bytes in its calculation.

For additional information about the OLE property set storage format, see [MS-OLEPS].

2.3.3.2.3.1 User Defined Property Set Constraints

The User Defined property set (section 2.3.3.2.3) is a location where user-defined properties can be persisted by associating arbitrary property names to data of arbitrary types. There are limits on the format the data can take, which are specified in this section.

2.3.3.2.3.1.1 Required Properties

Because the format of the User Defined property set (section 2.3.3.2.3) MUST associate a name with each property (except for the **Dictionary** and **CodePage** properties and links associated with properties), the property set MUST have a **Dictionary** property as specified in [MS-OLEPS] section 2.18.1 to make this association, and MUST follow the rules as specified in [MS-OLEPS] regarding property name and property identifier uniqueness.

The User Defined property set (section 2.3.3.2.3) MUST also contain the **CodePage** property as specified in [MS-OLEPS] section 2.18.2.

2.3.3.2.3.1.2 Supported Types

The User Defined property set (section 2.3.3.2.3) does not support every property type defined in the [MS-OLEPS] specification. Each property in this property set (with the exception of the **Dictionary** and **CodePage** properties) MUST be in one of the following formats:

- **VtString** (section 2.3.3.1.11)
- **TypedPropertyValue** ([MS-OLEPS] section 2.15), the type of which MUST be one of the following types:
  - VT_I4 (0x0003)
  - VT_R8 (0x0005)
  - VT_BOOL (0x000B)
  - VT_FILETIME (0x0040)

- VT_BLOB (0x0041)

2.3.3.2.3.2 Reserved Properties

The User Defined property set (section 2.3.3.2.3) has a number of names that are reserved for specific data. These names MUST only be used to specify the data for which the names are reserved. All of these properties are optional and can be absent. The reserved names and the data they specify are as specified in the following tables.

Reserved name	Format and description
_PID_GUID	MUST be a VT_BLOB **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. SHOULD NOT be written.<50> MUST be ignored if encountered on read. The **Size** field of the **BLOB** data (specified in [MS-OLEPS] section 2.9) specifies the count of bytes for a null-terminated **Unicode** string stored in the **Bytes** field of the **BLOB** excluding any padding. If **Size** is 0, no data is written for the **Bytes** field and the property value MUST be treated as an empty string. **Size** MUST be even. The property data MUST be padded to a multiple of 4 bytes, but the contents of the padding are undefined and MUST be ignored. The property specifies a unique **GUID**, as specified in [MS-DTYP], which specifies the identifier for the document. MUST NOT be linked.
_PID_LINKBASE	MUST be a VT_BLOB **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. The **Size** field of the **BLOB** data (specified in [MS-OLEPS] section 2.9) specifies the number of bytes for a null-terminated Unicode string stored in the **Bytes** field of the **BLOB** excluding any padding. If **Size** is 0, no data is written for the **Bytes** field and the property value MUST be treated as an empty string. **Size** MUST be even. The property data MUST be padded to a multiple of 4 bytes, but the contents of the padding are undefined and MUST be ignored. The property specifies the **base URL** for relative **hyperlinks** in the document. MUST NOT be linked.
_PID_HLINKS	MUST be a **VtHyperlinks** property (section 2.3.3.1.21). Specifies a list of hyperlinks contained in the document. MUST NOT be linked.
_MarkAsFinal	MUST be a VT_BOOL **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. If the property exists and the value is true, this specifies that the document is the final draft. MAY be ignored.<51>
Microsoft Theme	MUST be a **VtString** property (section 2.3.3.1.11). Specifies the name of the most recent theme applied to the document. MAY be ignored.<52>
Presentation	MUST be a **VtString** property (section 2.3.3.1.11). If the presentation document is created by using the first slide loaded from a slide library, this property in the created document specifies the document name that the first slide originated from. MAY be ignored.<53>
SlideDescription	MUST be a **VtString** property (section 2.3.3.1.11). If the presentation document is created by using the first slide loaded from a slide library, this property in the created document specifies a description of the slide that was used. MAY be ignored.<54>
_AllowSignedDocumentWithoutReadOnly	MUST be a VT_BOOL **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. If present and the value is true it specifies that the document that contains a document signature, as specified in [MS-OFFCRYPTO] section 2.5, and would otherwise be opened as read-only MUST instead be opened with both read and write permissions. MAY be ignored.<55>

Reserved name	Format and description

_TemplateID	MUST be a **VtString** property (section 2.3.3.1.11). The value of this property can be used to open a Help topic that corresponds to the template. MAY be ignored.<56>
	
	The format of the **VtString.stringValue.value** field of this property, if it exists, MUST conform to the following ABNF [RFC5234] representation:
	
	`_TemplateID = "TC" TemplateID LCID %x00`
	`TemplateID  = 8(DIGIT)`
	`LCID        = 4(DIGIT)`
	
	TemplateID: Specifies the unique identifier of the template that was used to create the document.
	
	LCID: Specifies the **LCID** of the template that was used to create the document.
	
	<DIGIT> is defined in [RFC5234] Appendix B.

The following properties are associated with **document workspace** management. All of these properties MAY be ignored.<57>

Reserved name	Format and description
_SourceUrl	MUST be a **VtString** property (section 2.3.3.1.11). Specifies the **URL** of the source document on a server for a working copy of that document that is opened from a server document workspace.
_CheckOutSrcUrl	MUST be a **VtString** property. Specifies the URL of the document workspace from which the document was checked out. The length of the property value string MUST be less than or equal to 256 characters, including the terminating NULL character.<58>
_LiveCopyIndex	MUST be a VT_I4 **TypedPropertyValue** ([MS-OLEPS] section 2.15) property. Specifies a time stamp for comparing the current state of a server document to a local copy of that document. The value MUST be generated by taking the **modified date-time** of the document file as a **FILETIME** data type ([MS-DTYP] section 2.3.3), multiplying the high-order 4-byte unsigned integer of the **modified date-time** by 3, and adding the low-order 4-byte unsigned integer of the **modified date-time**, producing a 4-byte unsigned integer **hash** value that is the value of the property. The value of this property can be used to resolve potential conflicts between two different copies of a document stored in a document workspace. If this property value of a particular document is the same as the **modified date-time** of that document file, hashed as described previously, the document MUST be considered to be equivalent to the original shared version of the file on the document workspace. If one of any two copies of a document is the original shared version and the two copies of the document differ, then the other document MUST be considered newer.

Reserved name	Format and description

_SharedFileIndex
_SharedFileIndex1
…
_SharedFileIndex*N*
	MUST be a **VtString** property. If the length of the property value exceeds 255 characters, excluding the terminating NULL character, then it MUST be segmented into multiple properties, each no larger than 255 characters plus a terminating NULL character. If segmented, each segment MUST be a property of the **VtString** type and MUST be null-terminated. Each segment except for the last MUST contain 255 characters of the value, with the last containing all remaining characters. The first segment MUST be named "_SharedFileIndex", and each successive segment MUST be named with an incremental index after this root name—so the second chunk would be named "_SharedFileIndex1", the third "_SharedFileIndex2", the eleventh "_SharedFileIndex10", and so on.
	
	The reconstructed value of the property specifies the workspace to which the document belongs. The value **SharedFileIndexValue** is formatted according to the following ABNF [RFC5234] representation:
	
	`SharedFileIndexValue = [VersionNumber] GUID URL`
	
	`VersionNumber = 0000020000000000`
	
	`GUID = ASCII-LEFT-CURLY-BRACKET 8(ASCII-DIGIT-HEXADECIMAL) ASCII-HYPHEN-MINUS 4(ASCII-DIGIT-HEXADECIMAL) ASCII-HYPHEN-MINUS 4(ASCII-DIGIT-HEXADECIMAL) ASCII-HYPHEN-MINUS 4(ASCII-DIGIT-HEXADECIMAL) ASCII-HYPHEN-MINUS 12(ASCII-DIGIT-HEXADECIMAL) ASCII-RIGHT-CURLY-BRACKET`
	
	`URL = *(UTF16-ANY)`
	
	VersionNumber: An optional field. If present it MUST be "0000020000000000" and is a reserved value.
	
	GUID: MUST be a GUID, as specified in [MS-DTYP], which specifies a unique identifier for the document.
	
	URL: Specifies the URL of the requested location of the file on the document workspace. If the location is already occupied by a document with a different GUID, the document could be relocated.

The following properties are associated with **smart document** solutions.

Reserved name	Format and description
Solution ID	MUST be a **VtString** property. It specifies the identifier<59> for the associated smart document solution. This property can be used to check whether the specified smart document solution is already installed and, if so, use the existing solution information to determine if the solution needs to be updated. If it is not previously installed, the solution is installed based on the value of the "Solution URL" property. MAY be ignored<60>. For more information about smart documents, see [MSDN-SDO]. The length of the property's **VtString.stringValue.value** field MUST be less than or equal to 51 characters, including the terminating NULL character.
Solution URL	MUST be a **VtString** property. It specifies the path to an **XML expansion pack manifest** file, which defines the contents of the smart document solution associated with the document. MAY be ignored<61>. For more information about XML expansion pack manifest files, see [MSDN-SDO]. The length of the property's **VtString.stringValue.value** field MUST be less than or equal to 256 characters, including the terminating NULL character.

Reserved name	Format and description
Solution Template ID	MUST be a **VtString** property. It specifies the identifier of a document template installed as part of the XML expansion pack. This property is used to ensure that the local copy of the document template installed as part of the XML expansion pack is updated when the document template referenced by the manifest file is updated. MAY be ignored<62>. For more information about XML expansion pack manifest files, see [MSDN-SDO]. The length of the property's **VtString.stringValue.value** field MUST be less than or equal to 256 characters, including the terminating NULL character.
Other Solution ID	MUST be a **VtString** property. It specifies the identifier<63> of a solution of type 'other' installed as a part of a XML expansion pack. For more information about XML expansion pack and types of solutions, see [MSDN-SDO]. This property is used when the solution itself does not affect the document behavior but it can be used to check if the referenced solution needs to be updated whenever the document is loaded. MAY be ignored<64>. The length of the property's **VtString.stringValue.value** field MUST be less than or equal to 51 characters, including the terminating NULL character.

The following properties are associated with **managed code** document **add-in** solutions<65>. For more information about managed code document add-in solutions, see [MSDN-AVSTOS]. These properties MUST both exist if present.

Reserved name	Format and description
_AssemblyLocation _AssemblyLocation0 _AssemblyLocation1 … _AssemblyLocation*N*	MUST be a **VtString** property. If the length of the property value exceeds 255 characters, excluding the terminating NULL character, then it MUST be segmented into multiple properties, each no larger than 255 characters plus a terminating NULL character. If segmented, each segment MUST be a property of the **VtString** type and MUST be null-terminated. Each segment except for the last MUST contain 255 characters of the value, with the last containing all remaining characters. If not segmented, the property name MUST be "_AssemblyLocation" or "_AssemblyLocation0". If segmented, the property segment names are generated by taking the root name "_AssemblyLocation" and appending the string representation of the decimal numbers 0, 1, 2, and so on. Therefore, the first segment MUST be named "_AssemblyLocation0", the second "_AssemblyLocation1", the eleventh "_AssemblyLocation10", and so on. The reconstructed value of the property specifies the path to be used to locate the managed **assembly** file specified by the "_AssemblyFile" property. MAY be ignored.<66>
_AssemblyFile _AssemblyFile0 _AssemblyFile1 … _AssemblyFile*N*	MUST be a **VtString** property. If the length of the property value exceeds 255 characters, excluding the terminating NULL character, then it MUST be segmented into multiple properties, each no larger than 255 characters plus a terminating NULL character. If segmented, each segment MUST be a property of the **VtString** type and MUST be null-terminated. Each segment except for the last MUST contain 255 characters of the value, with the last containing all remaining characters. If not segmented, the property name MUST be "_AssemblyFile" or "_AssemblyFile0". If segmented, the property segment names are generated by taking the root name "_AssemblyFile" and appending the string representation of the decimal numbers 0, 1, 2, and so on. Therefore, the first segment MUST be named "_AssemblyFile0", the second "_AssemblyFile1", the eleventh "_AssemblyFile10", and so on. The reconstructed value of the property specifies the file name of the managed assembly file containing the code for the document add-in

Reserved name	Format and description
	solution. MAY be ignored.<67>

The following properties specify information about a document that has been sent by using email, either for review or as an attachment, from an author to a reviewer or from a reviewer back to the author. These properties MAY be ignored.<68>

Reserved name	Format and description
_ReviewCycleID	MUST be a VT_I4 **TypedPropertyValue** property ([MS-OLEPS] section 2.15). Specifies a unique identifier for a document that was sent for review.
_TentativeReviewCycleID	MUST be a VT_I4 **TypedPropertyValue** property ([MS-OLEPS] section 2.15). Specifies a unique identifier for a document that was sent for review. If this property and the "_ReviewCycleID" property both exist in a document, then they MUST have the same value.
_ReviewingToolsShownOnce	MUST be a **VtString** property. MUST have a value of "", the empty string. Specifies that the client application has opened the document that was sent for review.
_NewReviewCycle	MUST be a **VtString** property. MUST have a value of "", the empty string. Specifies that the document was sent as an attachment or for review. MAY be ignored.<69>
_AuthorEmail	MUST be a **VtString** property. Specifies the email address from which the document was sent. The length of the property's **VtString.stringValue.value** field MUST be less than or equal to 256 characters, including the terminating NULL character.
_AuthorEmailDisplayName	MUST be a **VtString** property. Specifies the display name of the email address from which the document was sent. The length of the property's **VtString.stringValue.value** field MUST be less than or equal to 256 characters, including the terminating NULL character.
_EmailSubject	MUST be a **VtString** property. Specifies the subject of the email message with which the document was sent. The length of the property's **VtString.stringValue.value** field MUST be less than or equal to 256 characters, including the terminating NULL character.
_AdHocReviewCycleID	MUST be a VT_I4 **TypedPropertyValue** property ([MS-OLEPS] section 2.15). Specifies a unique identifier for reviewing a document that was sent as an attachment.
_PreviousAdHocReviewCycleID	MUST be a VT_I4 **TypedPropertyValue** property ([MS-OLEPS] section 2.15). Specifies a unique identifier for reviewing a document that was sent as an attachment. SHOULD be ignored.<70> MAY be changed if the "_AdHocReviewCycleID" property was present when the document was opened; then the value of the "_PreviousAdHocReviewCycleID" property when the document is again sent as an attachment MUST be the same as the original value of "_AdHocReviewCycleID".
_EmailStoreID _EmailStoreID0 _EmailStoreID1 … _EmailStoreID*N*	MUST be a **VtString** property. Specifies an identifier of the email server of the reviewer who is returning the document to the author. If the length of the property's **VtString.stringValue.value** field exceeds 255 characters, excluding the terminating NULL character, then it MUST be segmented into multiple properties, each no larger than 255 characters plus a terminating NULL character. If segmented, each segment MUST be a property of the **VtString** type and MUST be null-terminated. Each segment except for the last MUST contain 255 characters of the value, with the last containing all remaining characters. The name of each segment MUST be the base property name appended with the ASCII character whose value is equal to the value of the character '0' plus one less than the ordinal of the segment; hence, the first segment MUST be named "_EmailStoreID0", the second "_EmailStoreID1", the eleventh "_EmailStoreID10", and so on.
_EmailEntryID	MUST be a **VtString** property. Specifies an identifier of the email

Reserved name	Format and description
_EmailEntryID0 _EmailEntryID1 … _EmailEntryID*N*	message of the reviewer who is returning the document to the author. If the length of the property's **VtString.stringValue.value** field exceeds 255 characters, excluding the terminating NULL character, then it MUST be segmented into multiple properties, each no larger than 255 characters plus a terminating NULL character. If segmented, each segment MUST be a property of the **VtString** type and MUST be null-terminated. Each segment except for the last MUST contain 255 characters of the value, with the last containing all remaining characters. The name of each segment MUST be the base property name appended with the ASCII character whose value is equal to the value of the character '0' plus one less than the ordinal of the segment; hence, the first segment MUST be named "_EmailEntryID0", the second "_EmailEntryID1", the eleventh "_EmailEntryID10", and so on.

The following properties are associated with sensitivity labels.

Excel, Word, and PowerPoint may add specific Custom File Properties if a sensitivity label is applied.

Excel, Word, and PowerPoint may persist metadata from the Microsoft Information Protection Software Development Kit to Custom File Properties. Additionally, the property name "Sensitivity" is written and set to the GUID of the label ID applied to the file. The label ID is given by the Microsoft Information Protection Software Development Kit corresponding to the applied label.

Word may persist additional Custom File Properties if a sensitivity label is applied, as shown in the table below.

Reserved Property Name	Format and Description
ClassificationContentMarkingHeaderFontProps	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a header should be applied. The value is a # followed by the header text color in RRGGBB format, where RR, GG, and BB each represent a hexadecimal number indicating the amount of red (RR), green (GG), and blue (BB) to display on a scale of 0 (none) to FF (all). For example, #000000 indicates black and #FFFFFF indicates white. This is followed by a comma and the text font size in points, another comma, and the text font face name.
ClassificationContentMarkingHeaderShapeIds	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a header should be applied. The value is a comma separated list of shape IDs, written in hexadecimal without the leading 0x, which comprise the shapes which are used to render the headers.
ClassificationContentMarkingHeaderShapeIds-***<suffix>***	MUST be a **vtString** property. Written if the ClassificationContentMarkingHeaderShapeIds Custom File Property is written and the value grows too long because of the number of shapes inserted in the file. ***<Suffix>*** is a hexadecimal number without the leading 0x that starts with 1 and increments by 1. For example, ClassificationContentMarkingHeaderShapeIds-1 is written if ClassificationContentMarkingHeaderShapeIds Custom File Property is written and the value is too large because of the number of shapes inserted in the file. ClassificationContentMarkingHeaderShapeIds-2 is written if ClassificationContentMarkingHeaderShapeIds-1 Custom File Property is written and the value is too large because of the number of shapes inserted in the

Reserved Property Name	Format and Description
	file.
ClassificationContentMarkingHeaderText	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a header should be applied. The value is the header text displayed in the document as part of applying the sensitivity label.
ClassificationContentMarkingFooterFontProps	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a footer should be applied. The value is a # followed by the footer text color in RRGGBB format, where RR, GG, and BB each represent a hexadecimal number indicating the amount of red (RR), green (GG), and blue (BB) to display on a scale of 0 (none) to FF (all). For example, #000000 indicates black and #FFFFFF indicates white. This is followed by a comma and the text font size in points, another comma, and the text font face name.
ClassificationContentMarkingFooterShapeIds	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a footer should be applied. The value is a comma separated list of shape IDs, written in hexadecimal without the leading 0x, which comprise the shapes which are used to render the footers.
ClassificationContentMarkingFooterShapeIds-***<suffix>***	MUST be a **vtString** property. Written if the ClassificationContentMarkingFooterShapeIds Custom File Property is written and the value grows too long because of the number of shapes inserted in the file. ***<Suffix>*** is a hexadecimal number without the leading 0x that starts with 1 and increments by 1. For example, ClassificationContentMarkingHeaderShapeIds-1 is written if ClassificationContentMarkingFooterShapeIds Custom File Property is written and the value is too large because of the number of shapes inserted in the file. ClassificationContentMarkingFooterShapeIds-2 is written if ClassificationContentMarkingFooterShapeIds-1 Custom File Property is written and the value is too large because of the number of shapes inserted in the file.
ClassificationContentMarkingFooterText	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a footer should be applied. The value is the footer text displayed in the document as part of applying the sensitivity label.
ClassificationWatermarkFontProps	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a watermark should be applied. The value is a # followed by the watermark text color in RRGGBB format, where RR, GG, and BB each represent a hexadecimal number indicating the amount of red (RR), green (GG), and blue (BB) to display on a scale of 0 (none) to FF (all). For example, #000000 indicates black and #FFFFFF indicates white. This is followed by a comma and the text font size in points, another comma, and the text font face name.
ClassificationWatermarkShapeIds	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a watermark should be applied. The value is a comma separated list of shape IDs, written in hexadecimal without the leading 0x, which comprise the shapes which are used to render

Reserved Property Name	Format and Description
	the watermarks.
ClassificationWatermarkShapeIds-***<suffix>***	MUST be a **vtString** property. Written if the ClassificationWatermarkShapeIds Custom File Property is written and the value grows too long because of the number of shapes inserted in the file. ***<Suffix>*** is a hexadecimal number without the leading 0x that starts with 1 and increments by 1. For example, ClassificationWatermarkShapeIds-1 is written if ClassificationWatermarkShapeIds Custom File Property is written and the value is too large because of the number of shapes inserted in the file. ClassificationWatermarkShapeIds-2 is written if ClassificationWatermarkShapeIds-1 Custom File Property is written and the value is too large because of the number of shapes inserted in the file.
ClassificationWatermarkText	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a watermark should be applied. The value is the watermark text displayed in the document as part of applying the sensitivity label.

PowerPoint may persist additional Custom File Properties if a sensitivity label is applied, as shown in the table below.

Custom Property Name	Value Description
ClassificationContentMarkingHeaderText	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a header should be applied to the design master. The value is the header text displayed in the presentation as part of applying the sensitivity label.
ClassificationContentMarkingHeaderLocations	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a header should be applied to the design master. The value is the design name, followed by a colon and the shape ID. If there are multiple design masters, they are all listed and are separated by a reverse solidus (\). Colon (:) and reverse solidus (\) characters in the design name are escaped with a reverse solidus to (\:) and (\\) respectively.
ClassificationContentMarkingFooterText	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a footer should be applied to the design master. The value is the footer text displayed in the presentation as part of applying the sensitivity label.
ClassificationContentMarkingFooterLocations	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a footer should be applied to the design master. The value is the design name, followed by a colon and the shape ID. If there are multiple design masters, they are all listed and are separated by a reverse solidus (\). Colon (:) and reverse solidus (\) characters in the design name are escaped with a reverse solidus to (\:) and (\\) respectively.
ClassificationWatermarkText	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a watermark should be applied to the design master. The value is the

Custom Property Name	Value Description
	watermark text displayed in the presentation as part of applying the sensitivity label.
ClassificationWatermarkLocations	MUST be a **vtString** property. Written if the sensitivity label applied indicates that a watermark should be applied to the design master. The value is the watermark name, followed by a colon and the shape ID. If there are multiple design masters, they are all listed and are separated by a reverse solidus (\). Colon (:) and reverse solidus (\) characters in the design name are escaped with a reverse solidus to (\:) and (\\) respectively.

Excel, Word, and PowerPoint may persist additional Custom File Properties as metadata in **vtString** properties from the Microsoft Information Software Development Kit as metadata if a sensitivity label is applied, an example of which is shown in the table below.

Custom Property Name	Example Value
MSIP_Label_d9f23ae3-a239-45ea-bf23-0123456789ab_Enabled	true
MSIP_Label_d9f23ae3-a239-45ea-bf23-0123456789ab_SetDate	2018-09-24T21:38:47-0800
MSIP_Label_d9f23ae3-a239-45ea-bf23-0123456789ab_Method	Privileged
MSIP_Label_d9f23ae3-a239-45ea-bf23-0123456789ab_Name	Header, Footer, and Watermark
MSIP_Label_d9f23ae3-a239-45ea-bf23-0123456789ab_SiteId	242d863a-283c-5e0f-ae66-d99e8a20f526
MSIP_Label_d9f23ae3-a239-45ea-bf23-0123456789ab_ActionId	9808f4bb-209e-4696-8307-00003bb82621
MSIP_Label_d9f23ae3-a239-45ea-bf23-0123456789ab_ContentBits	7
Sensitivity	d9f23ae3-a239-45ea-bf23-0123456789ab

2.3.3.2.3.3 Linked Properties

The specification for a simple **OLE** property set storage as specified in [MS-OLEPS] specifies that properties contained in a property set storage can be named, when that property set storage contains the special **Dictionary** property. As specified previously, the **User Defined property set** (section 2.3.3.2.3) MUST contain a **Dictionary** property, and all properties of the property set except those that are special properties as specified in [MS-OLEPS] section 2.18 have names. However, the properties in this property set, excluding the special properties and **Reserved Properties** (section 2.3.3.2.3.2), can also be linked, meaning they receive their value from document content instead of from explicit value assignment.

A link for any property that allows it, if applicable, MUST be specified by creating an additional entry in the property set that is unnamed (that is, does not have an associated entry in the **Dictionary** property table). The **PropertyIdentifier** ([MS-OLEPS] section 2.19) for an entry created for this

purpose is equal to its associated property's **PropertyIdentifier** combined with the value **0x01000000** by using the bitwise **OR** operation.

A property's link entry MUST be in **VtString** (section 2.3.3.1.11) format.

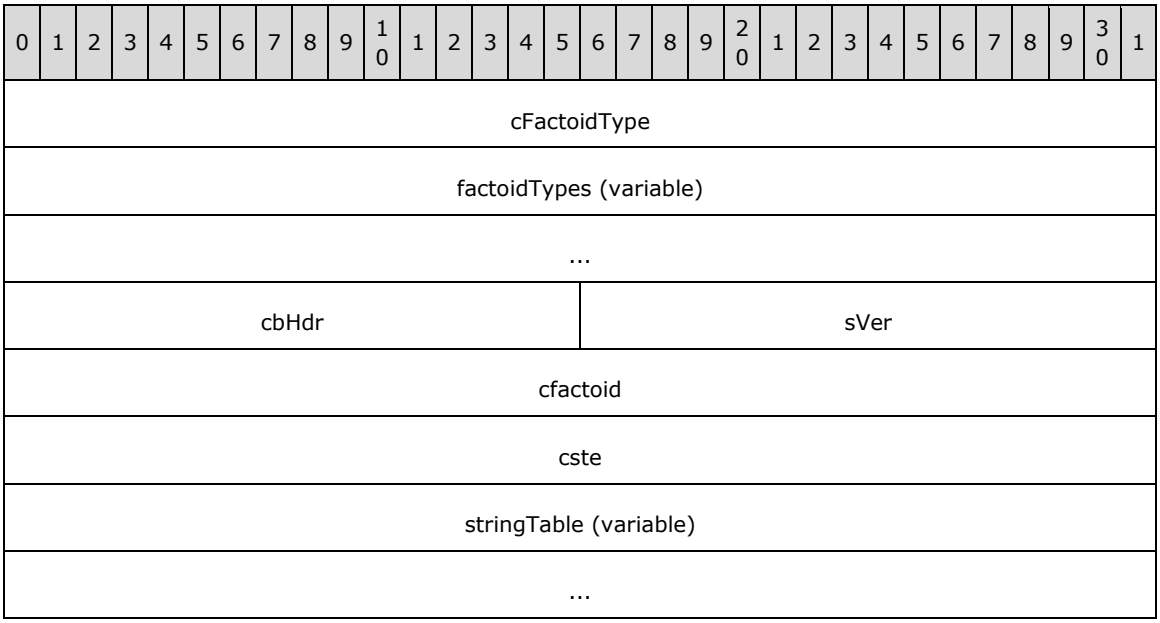
The presence of such an associated link for a given property specifies that the property, when written, MUST generate its value from the document content to which it is linked.<71> On document load, pursuant to the specification of the **GKPIDDSI_LINKSDIRTY** property flag (section 2.3.3.2.2.1), the application MUST update the linked document content in accordance with the new value of any linked property that was changed outside of the application.

2.3.4 SmartTag Objects

The **smart tags** are customer-defined semantics which can be embedded in a document. Smart tags allow semantic information to be added around words or types of data (for example, dates, phone numbers or addresses) within a document to provide information about the type of data contained within.

2.3.4.1 PropertyBagStore

This structure specifies the shared data for the **smart tags** embedded in the document.



cFactoidType (4 bytes): Unsigned integer specifying the count of elements in the **factoidTypes** member.

factoidTypes (variable): An array of **FactoidType** (section 2.3.4.2). Specifies the list of smart tag types.

cbHdr (2 bytes): Unsigned integer specifying the size in bytes of the **cbHdr**, **sVer**, **cfactoid**, and **cste** fields. MUST be 0xC.

sVer (2 bytes): Unsigned integer specifying the version number of the structure. The high-order byte specifies the **major version** number. The low-order byte specifies the **minor version** number. MUST be 0x0100.

cfactoid (4 bytes): Unsigned integer reserved for future use. MUST be ignored.

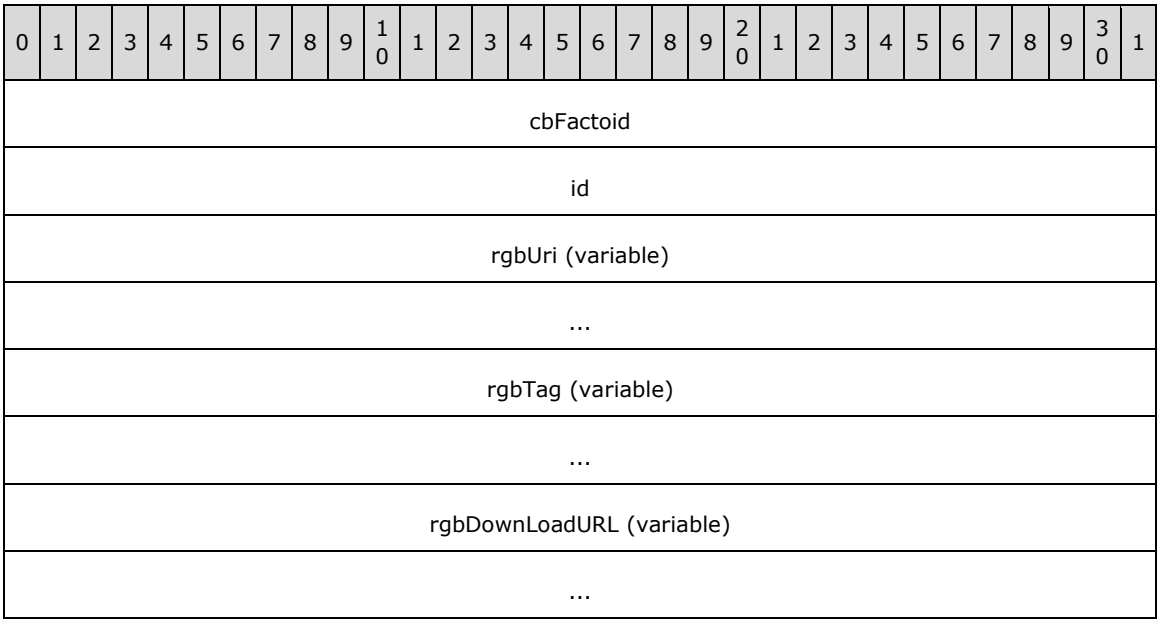
cste (4 bytes): Unsigned integer specifying the count of elements in the **stringTable** field.

stringTable (variable): An array of **PBString** (section 2.3.4.5). Specifies the list of strings. Elements of this table are referenced by their indices to form key/value pairs by the **keyIndex** and **valueIndex** fields in **Property** (section 2.3.4.4), which is in the **properties** field of **PropertyBag** (section 2.3.4.3).

2.3.4.2 FactoidType

Referenced by: PropertyBagStore

This structure specifies the type of **smart tag**.



cbFactoid (4 bytes): Unsigned integer specifying the count of total bytes, excluding itself, in the **FactoidType** structure.

id (4 bytes): Unsigned integer specifying the identifier of this smart tag type. There is a many-to-one mapping from the **PropertyBag** (section 2.3.4.3) to **FactoidType** using their respective **id** fields. MUST be less than or equal to 0xFFFF.

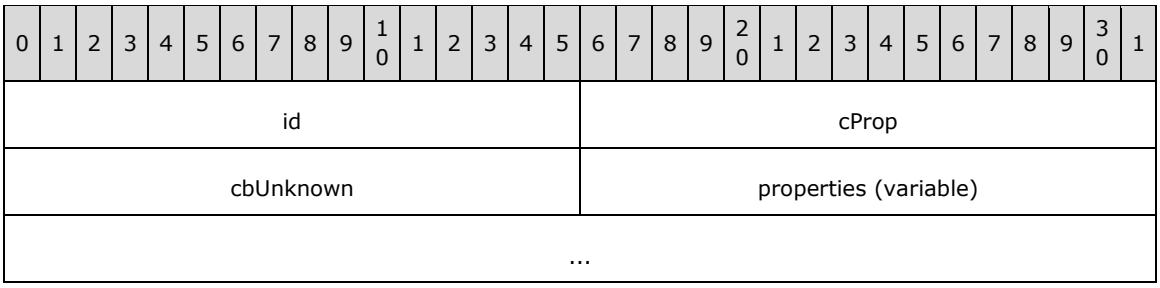
rgbUri (variable): A **PBString** structure (section 2.3.4.5) specifying the **XML namespace Uniform Resource Identifier (URI)** for the smart tag type.

rgbTag (variable): A **PBString** structure (section 2.3.4.5) specifying the tag name for the smart tag type.

rgbDownLoadURL (variable): A **PBString** structure (section 2.3.4.5) specifying the **URL** to download the particular smart tag type.

2.3.4.3 PropertyBag

This structure specifies the **smart tag** data.



id (2 bytes): Unsigned integer specifying the **id** of **FactoidType** (section 2.3.4.2) in the **factoidTypes** list of the **PropertyBagStore** (section 2.3.4.1).

cProp (2 bytes): Unsigned integer specifying the count of elements in the **properties** field.

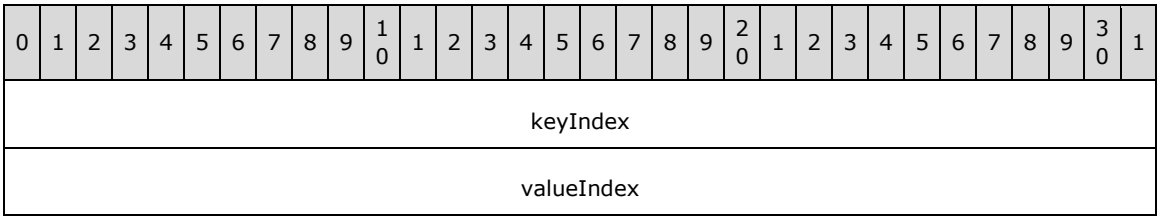
cbUnknown (2 bytes): Unused, reserved for future use. MUST be 0x0 and MUST be ignored.

properties (variable): An array of **Property** (section 2.3.4.4). It is a list of key/value indexes into the **stringTable** field of the **PropertyBagStore** (section 2.3.4.1) structure.

2.3.4.4 Property

Referenced by: PropertyBag

This structure specifies the indexes into the string table entries of the **stringTable** field in the **PropertyBagStore** (section 2.3.4.1) to form a key/value pair. It is used by the **smart tag recognizer** to store additional information that relates to the smart tag in a collection of key/value pairs, known as a property bag. This information can be later used to perform common tasks for the data type.



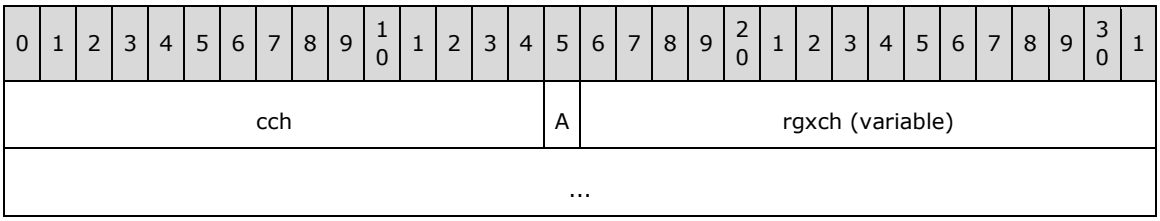
keyIndex (4 bytes): Unsigned integer specifying the key index.

valueIndex (4 bytes): Unsigned integer specifying the value index.

2.3.4.5 PBString

Referenced by: FactoidType, PropertyBagStore

This structure specifies a null-terminated string encoded either using a **Unicode** or **ANSI character set** format.



cch (15 bits): Specifies the count of characters in the string **rgxch**.

A - fAnsiString (1 bit): If set to 0x1 the string **rgxch** MUST be an ANSI character set string. If set to 0x0 then it MUST be a Unicode string.

rgxch (variable): A null-terminated ANSI character set or a Unicode string depending on the value of **fAnsiString** field.

2.3.5 RefEdit Control

A TextBox control, as specified by a **MorphData** control structure in [MS-OFORMS] section 2.2.5, with the following exceptions:

- **ShowDropButtonWhen**, specified in [MS-OFORMS] section 2.5.77, MUST have the value fmShowDropButtonWhenAlways.
- **DropButtonStyle**, specified in [MS-OFORMS] section 2.5.22, MUST have the value fmDropButtonStyleReduce.

All other properties of a **RefEdit** control are specified to have the same file format default values as a **MorphData** control structure, as specified in [MS-OFORMS] section 2.2.5, with a **DisplayStyle** property value of "fmDisplayStyleText", as specified in [MS-OFORMS] section 2.5.20.

2.3.6 Custom XML Data Storage

The Custom XML Data Storage specifies how to store a collection of XML fragments. It can be used to roundtrip arbitrary custom XML data with the document. This **storage** SHOULD<72> be created to store any XML fragment.

The name of this storage MUST be "MsoDataStore". Within this storage, zero or more sub-storages exist. The name of each of these sub-storages MUST be unique. Each of these sub-storages MUST contain two **streams** within it as specified in section 2.3.6.1 and 2.3.6.2 respectively.

2.3.6.1 Custom XML Data Storage Item

The name of this **stream** MUST be "Item". An instance of this stream contains an XML fragment.

Custom XML Data Storage items whose root namespace appears in the following table are further specified.

Root namespace	Explanation
http://schemas.microsoft.com/office/2006/customDocumentInformationPanel	Custom Property Editor (section 2.3.6.1.1)
http://schemas.microsoft.com/office/2006/metadata/customXsn	Custom Xsn (section 2.3.6.1.2)
http://schemas.microsoft.com/office/2006/metadata/contentType	Schema for Content Type (section 2.3.6.1.3)
http://schemas.microsoft.com/office/2006/coverPageProps	Cover Page Properties (section 2.3.6.1.4)

Root namespace	Explanation
http://schemas.microsoft.com/office/2006/metadata/longProperties	Long Properties (section 2.3.6.1.5)
office.server.policy	Collaborative Application Markup Language (CAML) Structure ([MS-WSSCAML] section 2.5.1.7)

2.3.6.1.1 Custom Property Editor

This section specifies the **XML schema** for the list and settings for any XSN files (as specified by InfoPath Form Template Format Structure Specification [MS-IPFF]) that are associated with the document to edit information in its custom XML data items.

2.3.6.1.1.1 XMLNamespace

Specifies the root namespace of custom XML data items that are appropriate for editing by the XSN specified in the **XSNLocation** element (section 2.3.6.1.1.2).

Parent element: customPropertyEditor (section 2.3.6.1.1.5)

```
<xsd:element name="XMLNamespace" type="xsd:anyURI"/>
```

2.3.6.1.1.2 XSNLocation

Specifies the path to the XSN file.

Parent element: customPropertyEditor (section 2.3.6.1.1.5)

```
<xsd:element name="XSNLocation" type="xsd:string"/>
```

2.3.6.1.1.3 showOnOpen

This element specifies whether to present a UI to edit either the core properties or properties in one of the custom XML data items in the document as specified by the **defaultPropertyEditorNamespace** (section 2.3.6.1.1.4). A setting of "true" specifies that the editing UI is displayed. If set to any other value or not present, the editing UI is not presented.

Parent element: customPropertyEditors (section 2.3.6.1.1.6)

```
<xsd:element name="showOnOpen" type="xsd:boolean"/>
```

2.3.6.1.1.4 defaultPropertyEditorNamespace

This element specifies the **XML namespace** that identifies the XSN to be shown by default when editing the properties contained in the custom XML data items of the document.

Parent element: customPropertyEditors (section 2.3.6.1.1.6)

```
<xsd:element name="defaultPropertyEditorNamespace" type="xsd:anyURI"/>
```

2.3.6.1.1.5 customPropertyEditor

This element associates a namespace with the location of an XSN file (as specified by InfoPath Form Template Format Structure Specification [MS-IPFF]). The **XMLNamespace** (section 2.3.6.1.1.1) child specifies which custom XML data items are suitable for editing; the **XSNLocation** (section 2.3.6.1.1.2) child specifies a path to the XSN file.

Parent element: customPropertyEditors (section 2.3.6.1.1.6)

Child elements
XMLNamespace (section 2.3.6.1.1.1)
XSNLocation (section 2.3.6.1.1.2)

```
<xsd:complexType name="CT_CustomPropertyEditor">
  <xsd:sequence>
    <xsd:element name="XMLNamespace" type="xsd:anyURI"/>
    <xsd:element name="XSNLocation" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.6.1.1.6 customPropertyEditors

This element specifies the list and settings for any XSN files (as specified by InfoPath Form Template Format Structure Specification [MS-IPFF]) that are associated with the document to edit information in its custom XML data items.

Child elements
customPropertyEditor (section 2.3.6.1.1.5)
defaultPropertyEditorNamespace (section 2.3.6.1.1.4)
showOnOpen (section 2.3.6.1.1.3)

```
<xsd:complexType name="CT_CustomPropertyEditors">
  <xsd:sequence>
    <xsd:element name="showOnOpen" type="xsd:boolean"/>
    <xsd:element name="defaultPropertyEditorNamespace" type="xsd:anyURI"/>
    <xsd:element name="customPropertyEditor" type="CT_CustomPropertyEditor"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.6.1.2 Custom Xsn

This section specifies the **XML schema** for specifying whether an XSN file (as specified by InfoPath Form Template Format Structure Specification [MS-IPFF]) is presented for editing information in the

custom XML data item with the namespace of http://schemas.microsoft.com/office/2006/metadata/properties in the document.

2.3.6.1.2.1 ST_TrueFalse

This simple type specifies restriction for specifying the Boolean values.

False : **false**

True : **true**

Referenced by: cached, openByDefault

```
<xsd:simpleType name="ST_TrueFalse">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="True"/>
    <xsd:enumeration value="False"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.3.6.1.2.2 xsnLocation

This element specifies the location of an XSN file (as specified by InfoPath Form Template Format Structure Specification [MS-IPFF]) to use for editing information in the custom XML data item with the namespace http://schemas.microsoft.com/office/2006/metadata/properties in the document.

Parent element: customXsn (section 2.3.6.1.2.6)

```
<xsd:element name="xsnLocation" type="xsd:string"/>
```

2.3.6.1.2.3 cached

This element specifies whether to use the XSN specified by the **xsnLocation** (section 2.3.6.1.2.2) for editing the information in the custom XML data item with the namespace http://schemas.microsoft.com/office/2006/metadata/properties in the document. If set to **false** the specified XSN MUST be used. If set to "True" the specified XSN MUST NOT be used.

Parent element: customXsn (section 2.3.6.1.2.6)

```
<xsd:element name="cached" type="ST_TrueFalse"/>
```

2.3.6.1.2.4 openByDefault

This element specifies whether to present UI to edit information in the custom XML data item with the namespace http://schemas.microsoft.com/office/2006/metadata/properties in the document, immediately after opening the same.

A value of **true** specifies that the UI is shown by default. A value of **false** or the absence of the element specifies that the UI is not shown.

Parent element: customXsn (section 2.3.6.1.2.6)

```
<xsd:element name="openByDefault" type="ST_TrueFalse"/>
```

2.3.6.1.2.5 xsnScope

Specifies the path to use for resolving any **relative paths** that appear within the XSN specified by **xsnLocation** (section 2.3.6.1.2.2).

Parent element: customXsn (section 2.3.6.1.2.6)

```
<xsd:element name="xsnScope" type="xsd:string"/>
```

2.3.6.1.2.6 customXsn

This element specifies whether an XSN file (as specified by InfoPath Form Template Format Structure Specification [MS-IPFF]) is to be presented for editing information in the custom XML data item with the namespace of "http://schemas.microsoft.com/office/2006/metadata/properties" in the document.

Child elements:

cached (section 2.3.6.1.2.3)

openByDefault (section 2.3.6.1.2.4)

xsnLocation (section 2.3.6.1.2.2)

xsnScope (section 2.3.6.1.2.5)

```
<xsd:complexType name="CT_CustomXsn">
  <xsd:sequence>
    <xsd:element name="xsnLocation" type="xsd:string"/>
    <xsd:element name="cached" type="ST_TrueFalse"/>
    <xsd:element name="openByDefault" type="ST_TrueFalse"/>
    <xsd:element name="xsnScope" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.6.1.3 Schema for Content Type

This section specifies the **XML schema** for the custom XML data item that specifies properties of the content type associated with the document.

2.3.6.1.3.1 ContentTypeId

This simple type specifies restriction for specifying a content type identifier for a document.

Referenced by: contentTypeSchema.SchemaForContentType@contentTypeID

```
<xsd:simpleType name="ContentTypeId">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="0x([0-9A-Fa-f][1-9A-Fa-f]|[1-9A-Fa-f][0-9A-Fa-f]|00[0-9A-Fa-f]{32})*"/>
    <xsd:minLength value="2"/>
    <xsd:maxLength value="1026"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.3.6.1.3.2 IntNonNegative

This simple type specifies the restriction for specifying non-negative integers.

Referenced by: contentTypeSchema.SchemaForContentType@contentTypeVersion, DummyContentTypeElement.DummyContentType@index

```
<xsd:simpleType name="IntNonNegative">
  <xsd:restriction base="xsd:int">
    <xsd:minInclusive value="0"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.3.6.1.3.3 UniqueIdentifierWithoutBraces

This simple type specifies the restriction on the **UniqueIdentifierWithoutBracesOrEmpty** (section 2.3.6.1.3.4) data type for specifying a **GUID**.

Referenced by: DummyContentTypeElement.DummyContentType@web

```
<xsd:simpleType name="UniqueIdentifierWithoutBraces">
  <xsd:restriction base="UniqueIdentifierWithoutBracesOrEmpty">
    <xsd:minLength value="1"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.3.6.1.3.4 UniqueIdentifierWithoutBracesOrEmpty

This simple type specifies the restrictions for specifying a **GUID** without braces or an empty string.

Referenced by: DummyContentTypeElement.DummyContentType@web

```
<xsd:simpleType name="UniqueIdentifierWithoutBracesOrEmpty">
  <xsd:restriction base="xsd:string">
    <xsd:pattern value="[0-9a-fA-F]{8}\-[0-9a-fA-F]{4}\-[0-9a-fA-F]{4}\-[0-9a-fA-
F]{12}|"/>
    <xsd:minLength value="0"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.3.6.1.3.5 DummyContentTypeElement

This specifies the child schema for the **contentTypeSchema** (section 2.3.6.1.3.7).

Attributes:

anchorId : Specifies the term that parents all terms that can be selected as a valid value for the property corresponding to this **xsd:element**.

If set to 00000000-0000-0000-0000-000000000000, specifies that all terms in the term set specified by the **termSetId** attribute MUST be treated as valid values for the property corresponding to this **xsd:element**.

MUST be present on **xsd:elements** for which the **taxonomy** attribute is set to **true**.

If applied to any other element, this attribute MUST be ignored.

decimals : When applied to an **xsd:element** in a schema other than the root schema whose content model is an xsd:restriction based on {http://schemas.microsoft.com/office/2006/documentManagement/types}Currency or {http://schemas.microsoft.com/office/2006/documentManagement/types}Number, specifies the number of fractional digits to display in the property-editing UI for the property corresponding to this **xsd:element**.

If applied to any other element, this attribute MUST be ignored.

default : When applied to an xsd:element in a schema other than the root schema or the core schema, specifies the initial value of the property corresponding to this **xsd:element**.

When applied to an xsd:element in a schema other than the root schema or core schema whose content model is an xsd:restriction based on {http://schemas.microsoft.com/office/2006/documentManagement/types}DateTime, if set to "[today]", specifies that the initial value of the property corresponding to this **xsd:element** MUST be set to the current **UTC** date and time.

When applied to an xsd:element in a schema other than the root schema or core schema whose content model is an xsd:restriction based on {http://schemas.microsoft.com/office/2006/documentManagement/types}Boolean, the value of 1 is interpreted as logical **true** and all other values as logical **false** in determining the initial value of the property corresponding to this **xsd:element**.

If applied to any other element, this attribute MUST be ignored.

description : When applied to an xsd:element in a schema other than the root schema, specifies the description of the property corresponding to this xsd:element.

If applied to any other element, this attribute MUST be ignored.

displayName : When applied to an **xsd:element** in a schema other than the root schema or core schema, specifies the display name for the property corresponding to this xsd:element.

If applied to any other element, this attribute MUST be ignored.

fieldId : Specifies a unique identifier for the field.

MUST be present on xsd:elements for which the **taxonomy** attribute is set to **true**.

If applied to any other element, this attribute MUST be ignored.

fieldsID : This attribute MUST be ignored.

format : When applied to an **xsd:element** in a schema other than the root schema or core schema whose content model is an xsd:restriction based on {http://schemas.microsoft.com/office/2006/documentManagement/types}DateTime, if set to "DateTime" specifies that the property value corresponding to this **xsd:element** MUST include both the date and time, and if not present or set to any other value, specifies that the property value MUST include only the date.

If applied to any other element, this attribute MUST be ignored.

hidden : When applied to an xsd:element in a schema other than the root schema or core schema, if set to **true**, specifies that the property corresponding to this **xsd:element** MUST NOT be presented for editing.

If applied to any other element, this attribute MUST be ignored.

index : Specifies the ordering on the property-editing UI of **xsd:elements** bearing this attribute from schemas other than the root schema.

First, the **xsd:elements** are broken into groups, as follows.

The first group is composed of any **xsd:element** that is all of the following:

- Is not the "description" element in the core properties.
- Is not a simpleType restriction based on {http://schemas.microsoft.com/office/2006/documentManagement/types}Note.
- Is not a complex content based on any of {http://schemas.microsoft.com/office/2006/documentManagement/types}MultiChoice, {http://schemas.microsoft.com/office/2006/documentManagement/types}MultiChoiceFillIn, or {http://schemas.microsoft.com/office/2006/documentManagement/types}MultiChoiceLookup.

The second group is composed of the remaining xsd:elements bearing this attribute from schemas other than the root schema.

The order within each group is the ascending order of the **index** attribute values.

If an xsd:element in a schema other than the root schema or core schema does not have this attribute, it MUST NOT be presented for editing.

If applied to any other element, this attribute MUST be ignored.

indexed : When applied to an **xsd:element** for which neither the **hidden** nor **readOnly** attributes are set to **true** in a schema other than the root schema or core schema, if set to **true**, specifies that the property corresponding to **this xsd:element** MUST be presented for editing.

If applied to any other element, this attribute MUST be ignored.

internalName : Specifies an internal name for the xsd:element bearing this attribute. It MUST be present on **xsd:elements** in a schema other than the root schema or core schema.

If applied to any other element, this attribute MUST be ignored.

isKeyword : If set to **true**, specifies that the property-editing UI for the property corresponding to this xsd:element uses all available term stores and term sets. If set to **false**, specifies that the property-editing UI for the property corresponding to this xsd:element uses the term set specified by the **termSetId** attribute.

MUST be present on xsd:elements for which the **taxonomy** attribute is set to **true**.

If applied to any other element, this attribute MUST be ignored.

LCID : When applied to an xsd:element in a schema other than the root schema or core schema whose content model is an xsd:restriction based on {http://schemas.microsoft.com/office/2006/documentManagement/types}Currency, specifies the **locale** identifier as specified in [MS-LCID] used to determine the currency symbol used in the display of the property corresponding to this xsd:element in the property-editing UI.

If applied to any other element, this attribute MUST be ignored.

list : Specifies the list to obtain data from, when displaying the property corresponding to this xsd:element in the property-editing UI.

MUST be present on xsd:elements for which neither the **hidden** nor **readOnly** attributes are set to **true** in a schema other than the root schema or core schema whose content model is an xsd:restriction based on {http://schemas.microsoft.com/office/2006/documentManagement/types}Lookup or {http://schemas.microsoft.com/office/2006/documentManagement/types}MultiLookup.

If applied to any other element, this attribute MUST be ignored.

open : If set to **true**, specifies that new terms are allowed to be added to the term set specified by the **termSetId** attribute. If set to **false**, the property-editing UI MUST NOT allow new terms to be added to the term set specified by the **termSetId** attribute.

MUST be present on xsd:elements for which the **taxonomy** attribute is set to **true**, and MUST be set to **true** if new terms are allowed to be added to the term set specified by the **termSetId** attribute. MUST be set to **false** otherwise.

If applied to any other element, this attribute MUST be ignored.

percentage : When applied to an **xsd:element** in a schema other than the root schema or core schema whose content model is an xsd:restriction based on {http://schemas.microsoft.com/office/2006/documentManagement/types}Currency or {http://schemas.microsoft.com/office/2006/documentManagement/types}Number, if set to **true**, specifies that the property corresponding to this xsd:element MUST be displayed as a percentage in

the property-editing UI, and if not present or set to any other value, specifies that the property corresponding to this xsd:element MUST NOT be displayed as a percentage in the property-editing UI.

If applied to any other element, this attribute MUST be ignored.

readOnly : When applied to an **xsd:element** in a schema other than the root schema or core schema, if set to **true**, specifies that the property corresponding to this **xsd:element** MUST NOT be presented for editing.

If applied to any other element, this attribute MUST be ignored.

requiredMultiChoice : When applied to an **xsd:element** in a schema other than the root schema or core schema whose content model is an xsd:restriction based on {http://schemas.microsoft.com/office/2006/documentManagement/types}MultiChoice, {http://schemas.microsoft.com/office/2006/documentManagement/types}MultiChoiceFillIn, or {http://schemas.microsoft.com/office/2006/documentManagement/types}MultiChoiceLookup, if set to "true", choices MUST be selected for the property corresponding to this **xsd:element**.

If applied to any other element, this attribute MUST be ignored.

root : MUST be set to **true** on the root schema element specified in the **contentTypeSchema** (section 2.3.6.1.3.7). It MUST NOT be applied to any other element.

showField : Specifies the text displayed in the property-editing UI for the lookup options of the property corresponding to this **xsd:element**.

MUST be present on **xsd:elements** for which neither the **hidden** nor **readOnly** attributes are set to **true** in a schema other than the root schema or core schema whose content model is an **xsd:restriction** based on {http://schemas.microsoft.com/office/2006/documentManagement/types}Lookup or {http://schemas.microsoft.com/office/2006/documentManagement/types}MultiLookup.

If applied to any other element, this attribute MUST be ignored.

sspId : Specifies the term store identifier used in the property-editing UI for the property corresponding to this **xsd:element**.

MUST be present on **xsd:elements** for which the **taxonomy** attribute is set to **true**, and MUST be set to 00000000-0000-0000-0000-000000000000 on **xsd:elements** for which the **isKeyword** attribute is set to **true**.

If applied to any other element, this attribute MUST be ignored.

taxonomy : MUST be set to **true** on **xsd:elements** in a schema other than the root schema or core schema whose content model has the following structure.

```
<xsd:complexType>
   <xsd:sequence>
      <xsd:element ref="pc:Terms" minOccurs="0" maxOccurs="1">
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>
```

It MUST NOT be applied to any other element.

taxonomyFieldName : MUST be present on **xsd:elements** for which the **taxonomy** attribute is set to **true**.

If applied to any other element, this attribute MUST be ignored.

taxonomyMulti : If set to **false**, specifies that only single values are displayed as valid in the property-editing UI for the property corresponding to this **xsd:element**.

MUST be present on **xsd:elements** for which the **taxonomy** attribute is set to **true**.

If applied to any other element, this attribute MUST be ignored.

termSetId : Specifies the term set used in the property-editing UI for the property corresponding to this xsd:element.

MUST be present on xsd:elements for which the **taxonomy** attribute is set to "true", and MUST be set to 00000000-0000-0000-0000-000000000000 on **xsd:elements** for which the **isKeyword** attribute is set to **true**.

If applied to any other element, this attribute MUST be ignored.

web : When applied to an **xsd:element** for which neither the **hidden** nor **readOnly** attributes are set to **true** in a schema other than the root schema or core schema whose content model is an xsd:restriction based on {http://schemas.microsoft.com/office/2006/documentManagement/types}Lookup or {http://schemas.microsoft.com/office/2006/documentManagement/types}MultiLookup, specifies the location of the website containing the list from which data is obtained to display the property corresponding to this xsd:element in the property-editing UI.

If applied to any other element, this attribute MUST be ignored.

```
<xsd:complexType name="CT_Dummy">
  <xsd:attribute name="anchorId" type="UniqueIdentifierWithoutBraces"/>
  <xsd:attribute name="decimals" type="xsd:string"/>
  <xsd:attribute name="default" type="xsd:string"/>
  <xsd:attribute name="description" type="xsd:string"/>
  <xsd:attribute name="displayName" type="xsd:string"/>
  <xsd:attribute name="fieldId" type="UniqueIdentifier"/>
  <xsd:attribute name="fieldsID" type="xsd:string"/>
  <xsd:attribute name="format" type="xsd:string"/>
  <xsd:attribute name="hidden" type="xsd:string"/>
  <xsd:attribute name="index" type="IntNonNegative"/>
  <xsd:attribute name="indexed" type="xsd:string"/>
  <xsd:attribute name="internalName" type="xsd:string"/>
  <xsd:attribute name="isKeyword" type="xsd:string"/>
  <xsd:attribute name="LCID" type="xsd:int"/>
  <xsd:attribute name="list" type="xsd:string"/>
  <xsd:attribute name="open" type="xsd:string"/>
  <xsd:attribute name="percentage" type="xsd:string"/>
  <xsd:attribute name="readOnly" type="xsd:string"/>
  <xsd:attribute name="requiredMultiChoice" type="xsd:string"/>
  <xsd:attribute name="root" type="xsd:boolean"/>
  <xsd:attribute name="showField" type="xsd:string"/>
  <xsd:attribute name="sspId" type="UniqueIdentifierWithoutBraces"/>
  <xsd:attribute name="taxonomy" type="xsd:boolean"/>
  <xsd:attribute name="taxonomyFieldName" type="xsd:string"/>
  <xsd:attribute name="taxonomyMulti" type="xsd:boolean"/>
  <xsd:attribute name="termSetId" type="UniqueIdentifierWithoutBraces"/>
  <xsd:attribute name="web" type="UniqueIdentifierWithoutBraces"/>
</xsd:complexType>
```

2.3.6.1.3.6 schema

This element specifies an **XML schema**.

Parent element: contentTypeSchema (section 2.3.6.1.3.7)

```
<xsd:element name="schema"/>
```

2.3.6.1.3.7 contentTypeSchema

The attributes and child elements of this element collectively specify a schema for the custom XML data item that specifies properties of the content type associated with the document. Each child of this element is an **xsd:schema** (section 2.3.6.1.3.6) element (as specified by [W3C-XSD]) that specifies one or more elements that appear in the custom XML data item. These xsd:schema elements are decorated with attributes from the http://schemas.microsoft.com/office/2006/metadata/properties/metaAttributes namespace as specified in section 2.3.6.1.3.5.

This element MUST contain an xsd:schema child element that has the **ma:root** Root (**DummyContentTypeElement.DummyContentType@root** (section 2.3.6.1.3.5)) attribute with value **true**. This **xsd:schema** specifies the structure of the root element of this Custom XML Data Storage item. This **xsd:schema** MUST have its **targetNamespace** attribute set to http://schemas.microsoft.com/office/2006/metadata/properties. It MUST declare and import any namespaces used by its child elements that are specified by other **xsd:schema** child elements. This xsd:schema element MUST have a **ma:fieldsID** Field IDs Hash (**DummyContentTypeElement.DummyContentType@fieldsID** (section 2.3.6.1.3.5)) attribute.

Additionally, this **xsd:schema** element MUST declare an element called **properties** with the following structure. The child elements of the **xsd:all** are not shown, but are instead specified following this XML markup.

```
<xsd:element name="properties">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element name="documentManagement">
        <xsd:complexType>
          <xsd:all>
            ...
          </xsd:all>
        </xsd:complexType>
      </xsd:element>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

In the following description, the term "content type" refers specifically to the root element of the Content Type Definition Type as specified in Collaborative Application Markup Language (CAML) Structure ([MS-WSSCAML] section 2.4.1.1).

Let the term "fields" denote the list of all Field elements that are the fields used in the content type, as specified in Collaborative Application Markup Language (CAML) Structure ([MS-WSSCAML] section 2.3.2.6). Furthermore, let the term "filtered fields" denote the list of all filtered fields to be those Field elements from the fields that satisfy all of the following conditions (the quotes are not included):

- The value of the **Type** attribute is not "Computed", "Calculated", "WorkflowStatus", "TaxonomyFieldType", or "TaxonomyFieldTypeMulti".
- The value of the **Name** attribute is not "Modified_x0020_By", "Created_x0020_By", "Modified", "Created", or "FileLeafRef".
- The value of the **StaticName** attribute is not "Title", "Subject", "_Author", "Keywords", "_Comments", "_Category", "Slides", "ContentType", or "Status".

The xsd:all element from the structure described previously MUST contain an xsd:element element for each Field element in the filtered fields set. This xsd:element MUST have a **ref** attribute. The value of the **ref** attribute MUST be a qualified reference to a type whose name is the value of the **StaticName** attribute of the Field element, and which resides in the namespace identified by the **SourceID** attribute of the Field element. Furthermore, if the **Required** attribute is not present on the Field

element, or if the attribute is present but its value is not **true**, or if the value of the **Type** attribute is "MultiChoice" or "LookupMulti", then a **minOccurs** attribute with value of 0 MUST be present on the xsd:element element.

For each unique string that appears as the value of the **SourceID** attributes on any Field element in fields, the contentTypeSchema element MUST contain an xsd:schema child whose **targetNamespace** attribute has that same value. Each **xsd:schema** MUST contain one xsd:element element for each Field whose **SourceID** attribute is the same as the **targetNamespace** of the xsd:schema.

Each xsd:element element MUST have a **Name** attribute whose value is the value of the **StaticName** attribute on the corresponding Field element. Each **xsd:element** element MUST have a **ma:index** Index (**DummyContentTypeElement.DummyContentType@index** (section 2.3.6.1.3.5)) attribute whose value is the number of Field elements that appear before the corresponding Field element within the Fields element in the content type. If the **Required** attribute is not present on the corresponding Field element, or if the attribute is present but its value is not **true**, or if the value of the **Type** attribute is "MultiChoice" or "LookupMulti", a **nullable** attribute ([W3C-XSD]) with value of **true** MUST be present on the **xsd:element** element.

The following attributes can appear on each **xsd:element** element:

- **ma:displayName** Display Name (**DummyContentTypeElement.DummyContentType@displayName** (section 2.3.6.1.3.5))
- **ma:decimals** Decimals (**DummyContentTypeElement.DummyContentType@decimals** (section 2.3.6.1.3.5))
- **ma:default** Default (**DummyContentTypeElement.DummyContentType@default** (section 2.3.6.1.3.5))
- **ma:description** Description (**DummyContentTypeElement.DummyContentType@description** (section 2.3.6.1.3.5)); **ma:format** Format (**DummyContentTypeElement.DummyContentType@format** (section 2.3.6.1.3.5))
- **ma:hidden** Hidden (**DummyContentTypeElement.DummyContentType@hidden** (section 2.3.6.1.3.5))
- **ma:LCID** LCID (**DummyContentTypeElement.DummyContentType@LCID** (section 2.3.6.1.3.5))
- **ma:list** List (**DummyContentTypeElement.DummyContentType@list** (section 2.3.6.1.3.5))
- **ma:internalName** Internal Name (**DummyContentTypeElement.DummyContentType@internalName** (section 2.3.6.1.3.5))
- **ma:readOnly** Read Only (**DummyContentTypeElement.DummyContentType@readOnly** (section 2.3.6.1.3.5))
- **ma:showField** Show Field (**DummyContentTypeElement.DummyContentType@showField** (section 2.3.6.1.3.5))
- **ma:percentage** Percentage (**DummyContentTypeElement.DummyContentType@percentage** (section 2.3.6.1.3.5))
- **ma:web** Web (**DummyContentTypeElement.DummyContentType@web** (section 2.3.6.1.3.5))
- **ma:requiredMultiChoice** Required Multiple Choice (**DummyContentTypeElement.DummyContentType@requiredMultiChoice** (section 2.3.6.1.3.5))

The **contentTypeSchema** MUST have an **xsd:schema** child element whose **targetNamespace** attribute has the value "http://schemas.openxmlformats.org/package/2006/metadata/core-properties".

Child element: schema (section 2.3.6.1.3.6)

Attributes:

contentTypeDescription : MUST be the same value as the **Description** attribute of the ContentType element (as specified in Collaborative Application Markup Language (CAML) Structure ([MS-WSSCAML] section 2.4.1.1) for the content type associated with the document.

contentTypeID : Identifies the content type associated with the document.

contentTypeName : MUST be the same value as the **Name** attribute of the ContentType element as specified in Collaborative Application Markup Language (CAML) Structure ([MS-WSSCAML] section 2.4.1.1) for the content type associated with the document.

contentTypeScope : MUST be empty and MUST be ignored.

contentTypeVersion : MUST be the same value as the **Version** attribute of the ContentType element as specified in Collaborative Application Markup Language (CAML) Structure ([MS-WSSCAML] section 2.4.1.1) for the content type associated with the document.

versionID : Specifies the string hash used to determine if the definition of the content type associated with the document has changed since the last time the content type was applied to the document. The value MUST be computed as an MD5 hash (as specified in [RFC1321]) of the substring that represents the **ContentType** element (beginning with "<ContentType" and ending with "</ContentType>", both strings included) within the SOAP result of the **GetListContentType** web service request ([MS-LISTSWS] section 3.1.4.17).

```
<xsd:complexType name="CT_ContentTypeSchema">
  <xsd:sequence>
    <xsd:element ref="xsd:schema" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="contentTypeName" type="xsd:string" />
  <xsd:attribute name="contentTypeID" type="ContentTypeId" />
  <xsd:attribute name="contentTypeVersion" type="IntNonNegative" />
  <xsd:attribute name="contentTypeDescription" type="xsd:string" />
  <xsd:attribute name="contentTypeScope" type="xsd:string" />
  <xsd:attribute name="versionID" type="xsd:string" />
</xsd:complexType>
```

2.3.6.1.4 Cover Page Properties

This section specifies the **XML schema** for specifying properties used for the document cover page<73>.

2.3.6.1.4.1 ST_PublishDate

This simple type specifies the document publish date.

```
<xsd:simpleType name="ST_PublishDate">
  <xsd:union memberTypes="xsd:date xsd:dateTime xsd:string"/>
</xsd:simpleType>
```

2.3.6.1.4.2 PublishDate

Specifies the date the document was published.

Parent element: CoverPageProperties (section 2.3.6.1.4.8)

```
<xsd:element name="PublishDate" type="ST_PublishDate"/>
```

2.3.6.1.4.3 Abstract

Specifies the abstract of the document's content.

Parent element: CoverPageProperties (section 2.3.6.1.4.8)

```
<xsd:element name="Abstract" type="xsd:string"/>
```

2.3.6.1.4.4 CompanyAddress

Specifies the address of the company to be used in the document.

Parent element: CoverPageProperties (section 2.3.6.1.4.8)

```
<xsd:element name="CompanyAddress" type="xsd:string"/>
```

2.3.6.1.4.5 CompanyPhone

Specifies the phone number of the company to be used in the document.

Parent element: CoverPageProperties (section 2.3.6.1.4.8)

```
<xsd:element name="CompanyPhone" type="xsd:string"/>
```

2.3.6.1.4.6 CompanyFax

Specifies the fax number of the company to be used in the document.

Parent element: CoverPageProperties (section 2.3.6.1.4.8)

```
<xsd:element name="CompanyFax" type="xsd:string"/>
```

2.3.6.1.4.7 CompanyEmail

Specifies the email address of the company to be used in the document.

Parent element: CoverPageProperties (section 2.3.6.1.4.8)

```
<xsd:element name="CompanyEmail" type="xsd:string"/>
```

2.3.6.1.4.8 CoverPageProperties

Specifies the parent element for properties used for the document cover page.

Child elements:

Abstract (section 2.3.6.1.4.3)

CompanyAddress (section 2.3.6.1.4.4)

CompanyEmail (section 2.3.6.1.4.7)

CompanyFax (section 2.3.6.1.4.6)

CompanyPhone (section 2.3.6.1.4.5)

PublishDate (section 2.3.6.1.4.2)

```
<xsd:complexType name="CT_CoverPageProperties">
  <xsd:sequence>
    <xsd:element name="PublishDate" type="ST_PublishDate"/>
    <xsd:element name="Abstract" type="xsd:string"/>
    <xsd:element name="CompanyAddress" type="xsd:string"/>
    <xsd:element name="CompanyPhone" type="xsd:string"/>
    <xsd:element name="CompanyFax" type="xsd:string"/>
    <xsd:element name="CompanyEmail" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.6.1.5 Long Properties

This section specifies the **XML schema** for specifying the full content of the User Defined Properties (section 2.3.3.2.3) whose values exceed 255 characters.

2.3.6.1.5.1 LongProp

This element specifies the full value of a user-defined property that exceeds 255 characters.

Parent element: LongProperties (section 2.3.6.1.5.2)

Attributes:

name : Specifies the name of the user-defined property.

```
<xsd:element name="LongProp" minOccurs="0" maxOccurs="unbounded">
  <xsd:complexType>
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="name" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:element>
```

2.3.6.1.5.2 LongProperties

This element specifies the full content of user defined properties whose values exceed 255 characters.

Child element: LongProp (section 2.3.6.1.5.1)

```
<xsd:complexType name="CT_LongProperties">
  <xsd:sequence>
    <xsd:element name="LongProp" minOccurs="0" maxOccurs="unbounded">
      <xsd:complexType>
        <xsd:simpleContent>
          <xsd:extension base="xsd:string">
            <xsd:attribute name="name" type="xsd:string"/>
          </xsd:extension>
        </xsd:simpleContent>
      </xsd:complexType>
    </xsd:element>
  </xsd:sequence>
</xsd:complexType>
```

2.3.6.2 Custom XML Data Storage Properties

The name of this **stream** MUST be "Properties". An instance of this stream contains an XML fragment conforming to the namespace http://schemas.openxmlformats.org/officeDocument/2006/customXml. It contains the set of properties which are specified for the corresponding custom XML data item.

These properties consist of a unique ID for the custom XML data item and information about the set of **XML schemas** used by the custom XML data item.

2.3.6.2.1 ST_Guid

This simple type specifies the restriction for specifying a **GUID**.

Referenced by: *datastoreItem.CustomXmlDataProperties@itemID*

```
<xsd:simpleType name="ST_Guid">
  <xsd:restriction base="xsd:token">
    <xsd:pattern value="\{[0-9A-F]{8}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{4}-[0-9A-F]{12}\}"/>
  </xsd:restriction>
</xsd:simpleType>
```

2.3.6.2.2 schemaRef

This element specifies a single **XML schema** that is associated with the custom XML data item. This XML schema is identified using its target namespace.

Parent element: schemaRefs (section 2.3.6.2.3)

Attributes:

uri : Specifies the target namespace for the XML Schema associated with this schema reference.

```
<xsd:complexType name="CT_DatastoreSchemaRef">
  <xsd:attribute name="uri" type="xsd:string" use="required"/>
</xsd:complexType>
```

2.3.6.2.3 schemaRefs

This element specifies the set of **XML schemas** that are associated with the corresponding custom XML data item. If this element is present, then the set of XML schemas provided within MUST be used to validate the contents of the corresponding custom XML data item. If no child **schemaRef** (section 2.3.6.2.2) elements exist then XML schema validation MUST NOT be performed on the custom XML data item.

Parent element: datastoreItem (section 2.3.6.2.4)

Child element: schemaRef (section 2.3.6.2.2)

```
<xsd:complexType name="CT_DatastoreSchemaRefs">
  <xsd:sequence>
    <xsd:element name="schemaRef" type="CT_DatastoreSchemaRef" minOccurs="0"
maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
```

2.3.6.2.4 dataStoreItem

This element specifies the properties for the corresponding custom XML data item. The set of properties specified within this element is attached to the custom XML data item.

Child element: schemaRefs (section 2.3.6.2.3)

Attributes:

itemID : Specifies a **GUID** that uniquely identifies a single custom XML data item within the document. Each **itemID** value MUST be unique among all custom XML data items in the document.

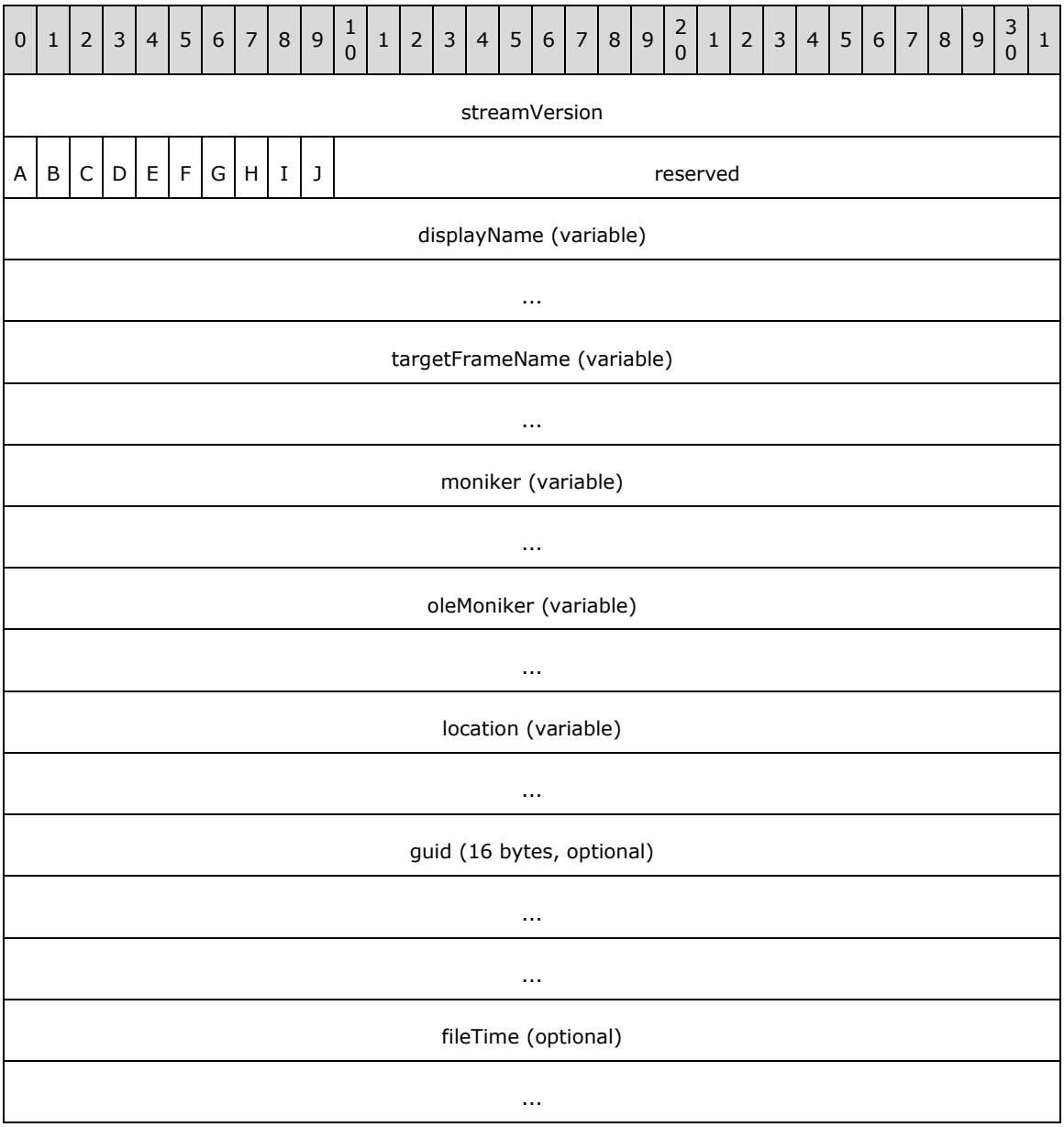
```
<xsd:complexType name="CT_DatastoreItem">
  <xsd:sequence>
    <xsd:element name="schemaRefs" type="CT_DatastoreSchemaRefs" minOccurs="0"/>
  </xsd:sequence>
  <xsd:attribute name="itemID" type="ST_Guid" use="required"/>
</xsd:complexType>
```

2.3.7 Hyperlinks

This section specifies **hyperlink**-related objects.

2.3.7.1 Hyperlink Object

This structure specifies a **hyperlink** and hyperlink-related information.



streamVersion (4 bytes): An unsigned integer that specifies the version number of the serialization implementation used to save this structure. This value MUST equal 2.

A - hlstmfHasMoniker (1 bit): A bit that specifies whether this structure contains a **moniker**. If **hlstmfMonikerSavedAsStr** equals 1, this value MUST equal 1.

B - hlstmfIsAbsolute (1 bit): A bit that specifies whether this hyperlink is an **absolute path** or **relative path**.

Value	Meaning
0	This hyperlink is a relative path.
1	This hyperlink is an absolute path.

C - hlstmfSiteGaveDisplayName (1 bit): A bit that specifies whether the creator of the hyperlink specified a display name.

D - hlstmfHasLocationStr (1 bit): A bit that specifies whether this structure contains a **hyperlink location**.

E - hlstmfHasDisplayName (1 bit): A bit that specifies whether this structure contains a display name.

F - hlstmfHasGUID (1 bit): A bit that specifies whether this structure contains a **GUID** as specified by [MS-DTYP].

G - hlstmfHasCreationTime (1 bit): A bit that specifies whether this structure contains the creation time of the file that contains the hyperlink.

H - hlstmfHasFrameName (1 bit): A bit that specifies whether this structure contains a **target frame** name.

I - hlstmfMonikerSavedAsStr (1 bit): A bit that specifies whether the moniker was saved as a string.

J - hlstmfAbsFromGetdataRel (1 bit): A bit that specifies whether the hyperlink specified by this structure is an absolute path generated from a relative path.

reserved (22 bits): MUST be zero and MUST be ignored.

displayName (variable): An optional **HyperlinkString** (section 2.3.7.9) that specifies the display name for the hyperlink. MUST exist if and only if **hlstmfHasDisplayName** equals 1.

targetFrameName (variable): An optional **HyperlinkString** (section 2.3.7.9) that specifies the target frame. MUST exist if and only if **hlstmfHasFrameName** equals 1.

moniker (variable): An optional **HyperlinkString** (section 2.3.7.9) that specifies the hyperlink moniker. MUST exist if and only if **hlstmfHasMoniker** equals 1 and **hlstmfMonikerSavedAsStr** equals 1.

oleMoniker (variable): An optional **HyperlinkMoniker** (section 2.3.7.2) that specifies the hyperlink moniker. MUST exist if and only if **hlstmfHasMoniker** equals 1 and **hlstmfMonikerSavedAsStr** equals 0.

location (variable): An optional **HyperlinkString** (section 2.3.7.9) that specifies the hyperlink location. MUST exist if and only if **hlstmfHasLocationStr** equals 1.

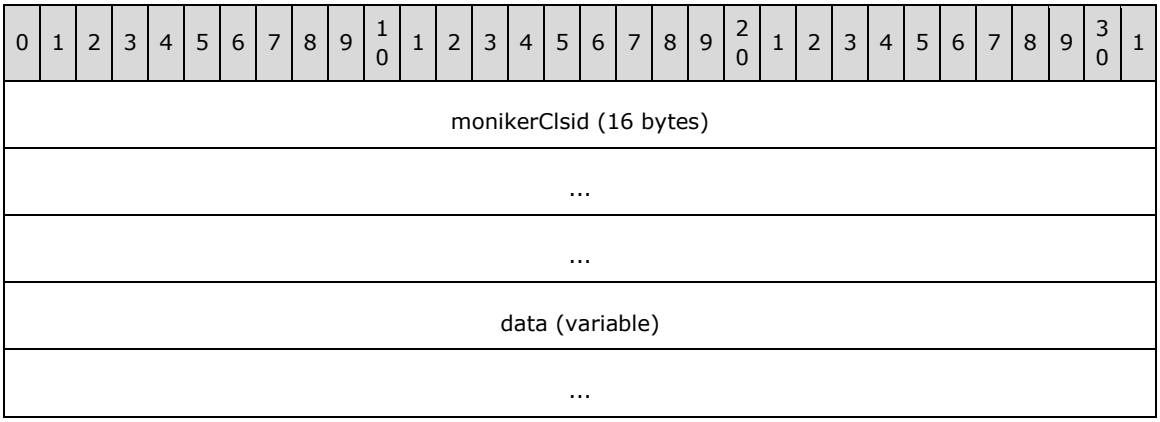
guid (16 bytes): An optional GUID as specified by [MS-DTYP] that identifies this hyperlink. MUST exist if and only if **hlstmfHasGUID** equals 1.

fileTime (8 bytes): An optional **FileTime** structure as specified by [MS-DTYP] that specifies the **UTC** file creation time. MUST exist if and only if **hlstmfHasCreationTime** equals 1.

2.3.7.2 HyperlinkMoniker

Referenced by: CompositeMoniker, Hyperlink Object

This structure specifies a **hyperlink moniker**.



monikerClsid (16 bytes): A **class identifier (CLSID)** that specifies the **Component Object Model (COM)** component that saved this structure. MUST be a value from the following table:

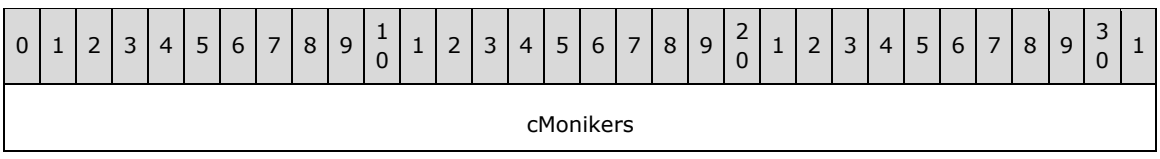
Value	Meaning
{0x79EAC9E0, 0xBAF9, 0x11CE, 0x8C, 0x82, 0x00, 0xAA, 0x00, 0x4B, 0xA9, 0x0B}	**Data** field contains a **URLMoniker** (section 2.3.7.6).
{0x00000303, 0x0000, 0x0000, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46}	**Data** field contains a **FileMoniker** (section 2.3.7.8).
{0x00000309, 0x0000, 0x0000, 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46}	**Data** field contains a **CompositeMoniker** (section 2.3.7.3).
{0x00000305, 0x0000, 0x0000, { 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46 }	**Data** field contains an **AntiMoniker** (section 2.3.7.4).
{0x00000304, 0x0000, 0x0000, { 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46 }	**Data** field contains an **ItemMoniker** (section 2.3.7.5).

data (variable): A moniker of the type specified by **monikerClsid**.

2.3.7.3 CompositeMoniker

Referenced by: HyperlinkMoniker

This structure specifies a **composite moniker**. A composite moniker is a collection of arbitrary **monikers**. For more information about composite monikers see [MSDN-IMGCMI].



monikerArray (variable)
...

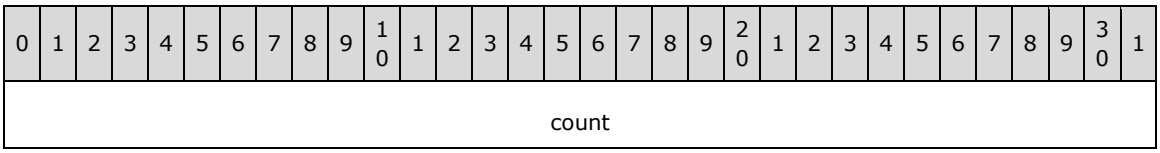
cMonikers (4 bytes): An unsigned integer that specifies the count of monikers in **monikerArray**.

monikerArray (variable): An array of **HyperlinkMonikers** (section 2.3.7.2). Each array element specifies a moniker of arbitrary type.

2.3.7.4 AntiMoniker

Referenced by: HyperlinkMoniker

This structure specifies an **anti-moniker**. An anti-moniker acts as the inverse of any moniker it is composed onto, effectively canceling out that moniker. In a **composite moniker**, anti-monikers are used to cancel out existing moniker elements, because monikers cannot be removed from a composite moniker. For more information about anti-monikers, see [MSDN-IMAMI].

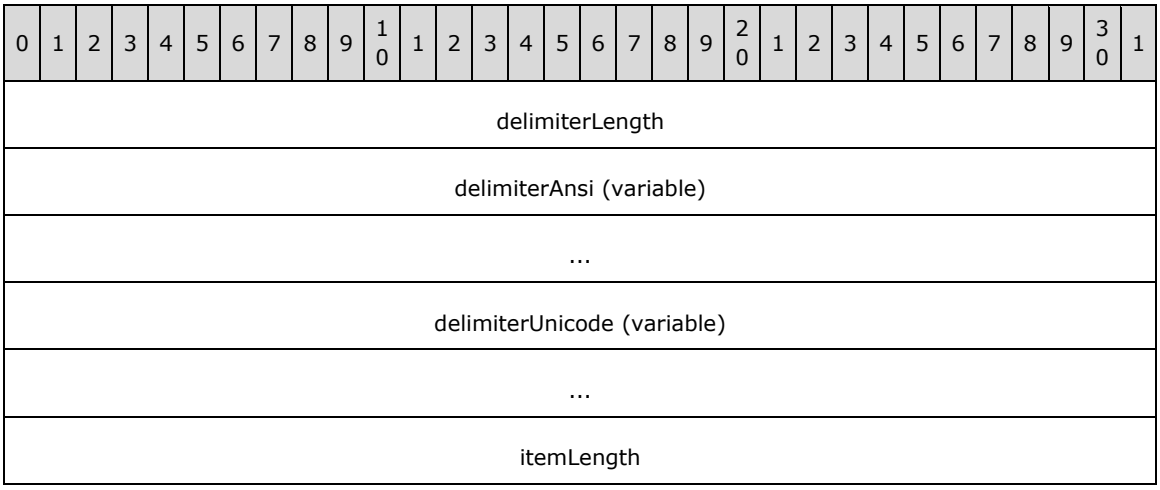


count (4 bytes): An unsigned integer that specifies the number of anti-monikers that have been composed together to create this instance. When an anti-moniker is composed with another anti-moniker, the resulting composition would have a **count** field equaling the sum of the two **count** fields of the composed anti-monikers. This value MUST be less than or equal to 1048576.

2.3.7.5 ItemMoniker

Referenced by: HyperlinkMoniker

This structure specifies an **item moniker**. Item monikers are used to identify objects within containers, such as a portion of a document, an embedded object within a compound document, or a range of cells within a spreadsheet. For more information about item monikers, see [MSDN-IMCOM].



itemAnsi (variable)
...
itemUnicode (variable)
...

delimiterLength (4 bytes): An unsigned integer that specifies the sum of the count of bytes in the **delimiterAnsi** and **delimiterUnicode** fields.

delimiterAnsi (variable): A null-terminated array of ANSI characters that specifies a delimiter for this **moniker**. Delimiters are used to separate monikers that are part of a collection of monikers in a **composite moniker**. The number of characters in the array is determined by the position of the terminating NULL character.

delimiterUnicode (variable): An optional array of **Unicode** characters that specifies a delimiter for this moniker if the delimiter cannot be completely specified in ANSI characters. This field MUST exist if and only if **delimiterLength** is greater than the size of **delimiterAnsi** in bytes. The number of characters in the array is determined `by (delimiterLength - (size of delimiterAnsi in bytes)) / 2`.

itemLength (4 bytes): An unsigned integer that specifies the count of bytes in the **itemAnsi** and **itemUnicode** fields.

itemAnsi (variable): A null-terminated array of ANSI characters that specifies the string used to identify this item in a collection of items. The number of characters in this array is specified by **itemLength**.

itemUnicode (variable): An optional array of Unicode characters that specifies the string used to identify this item in a collection of items, if the string cannot be completely specified in ANSI characters. This field MUST exist if and only if **itemLength** is greater than the size of **itemAnsi** in bytes. The number of characters in the array is determined by `(itemLength - (size of itemAnsi field in bytes)) / 2`.

2.3.7.6 URLMoniker

Referenced by: HyperlinkMoniker

This structure specifies a **URL moniker**. For more information about URL monikers, see [MSDN-URLM].

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
length																															
url (variable)																															
...																															
serialGUID (16 bytes, optional)																															
...																															

...
serialVersion (optional)
uriFlags (optional)

length (4 bytes): An unsigned integer that specifies the size of this structure in bytes, excluding the size of the **length** field.

The value of this field MUST be either the byte size of the **url** field (including the terminating NULL character) or the byte size of the **url** field plus 24.

If the value of this field is set to the byte size of the **url** field, then the **serialGUID**, **serialVersion**, and **uriFlags** fields MUST NOT be present.

If the value of this field is set to the byte size of the **url** field plus 24, then the **serialGUID**, **serialVersion**, and **uriFlags** fields MUST be present.

url (variable): A null-terminated array of **Unicode** characters that specifies the **URL**. The number of characters in the array is determined by the position of the terminating NULL character.

serialGUID (16 bytes): An optional **GUID** as specified by [MS-DTYP] for this implementation of the URL moniker serialization. This field MUST equal {0xF4815879, 0x1D3B, 0x487F, 0xAF, 0x2C, 0x82, 0x5D, 0xC4, 0x85, 0x27, 0x63} if present.

serialVersion (4 bytes): An optional unsigned integer that specifies the version number of this implementation of the URL moniker serialization. This field MUST equal 0 if present.

uriFlags (4 bytes): An optional **URICreateFlags** structure (section 2.3.7.7) that specifies creation flags for an [RFC3986] compliant URI.

2.3.7.7 URICreateFlags

Referenced by: URLMoniker

This structure specifies creation flags for an [RFC3986] compliant **URI**. For more information about URI creation flags, see [MSDN-CreateUri].

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	reserved															

A - createAllowRelative (1 bit): A bit that specifies that if the **URI scheme** is unspecified and not implicitly "file," a relative scheme is assumed during creation of the URI.

B - createAllowImplicitWildcardScheme (1 bit): A bit that specifies that if the URI scheme is unspecified and not implicitly "file," a wildcard scheme is assumed during creation of the URI.

C - createAllowImplicitFileScheme (1 bit): A bit that specifies that if the URI scheme is unspecified and the URI begins with a drive letter or a **UNC** path, a file scheme is assumed during creation of the URI.

D - createNoFrag (1 bit): A bit that specifies that if a **URI query** string is present, the **URI fragment** is not looked for during creation of the URI.

E - createNoCanonicalize (1 bit): A bit that specifies that the scheme, host, authority, path, and fragment will not be canonicalized during creation of the URI. This value MUST be 0 if **createCanonicalize** equals 1.

F - createCanonicalize (1 bit): A bit that specifies that the scheme, host, authority, path, and fragment will be canonicalized during creation of the URI. This value MUST be 0 if **createNoCanonicalize** equals 1.

G - createFileUseDosPath (1 bit): A bit that specifies that **MS-DOS path compatibility mode** will be used during creation of file URIs.

H - createDecodeExtraInfo (1 bit): A bit that specifies that percent encoding and percent decoding canonicalizations will be performed on the URI query and URI fragment during creation of the URI. This field takes precedence over the **createNoCanonicalize** field. This value MUST be 0 if **createNoDecodeExtraInfo** equals 1. The value 1 can also be saved. This will cause a return value of E_INVALIDARG from CreateUri().

I - createNoDecodeExtraInfo (1 bit): A bit that specifies that percent encoding and percent decoding canonicalizations will not be performed on the URI query and URI fragment during creation of the URI. This field takes precedence over the **createCanonicalize** field. This value MUST be 0 if **createDecodeExtraInfo** equals 1. The value 1 can also be saved. This will cause a return value of E_INVALIDARG from CreateUri().

J - createCrackUnknownSchemes (1 bit): A bit that specifies that hierarchical URIs with unrecognized URI schemes will be treated like hierarchical URIs during creation of the URI. This value MUST be 0 if **createNoCrackUnknownSchemes** equals 1.

K - createNoCrackUnknownSchemes (1 bit): A bit that specifies that hierarchical URIs with unrecognized URI schemes will be treated like opaque URIs during creation of the URI. This value MUST be 0 if **createCrackUnknownSchemes** equals 1.

L - createPreProcessHtmlUri (1 bit): A bit that specifies that preprocessing will be performed on the URI to remove control characters and white space during creation of the URI. This value MUST be 0 if **createNoPreProcessHtmlUri** equals 1.

M - createNoPreProcessHtmlUri (1 bit): A bit that specifies that preprocessing will not be performed on the URI to remove control characters and white space during creation of the URI. This value MUST be 0 if **createPreProcessHtmlUri** equals 1.

N - createIESettings (1 bit): A bit that specifies that registry settings will be used to determine default **URL** parsing behavior during creation of the URI. This value MUST be 0 if **createNoIESettings** equals 1.

O - createNoIESettings (1 bit): A bit that specifies that registry settings will not be used to determine default URL parsing behavior during creation of the URI. This value MUST be 0 if **createIESettings** equals 1.

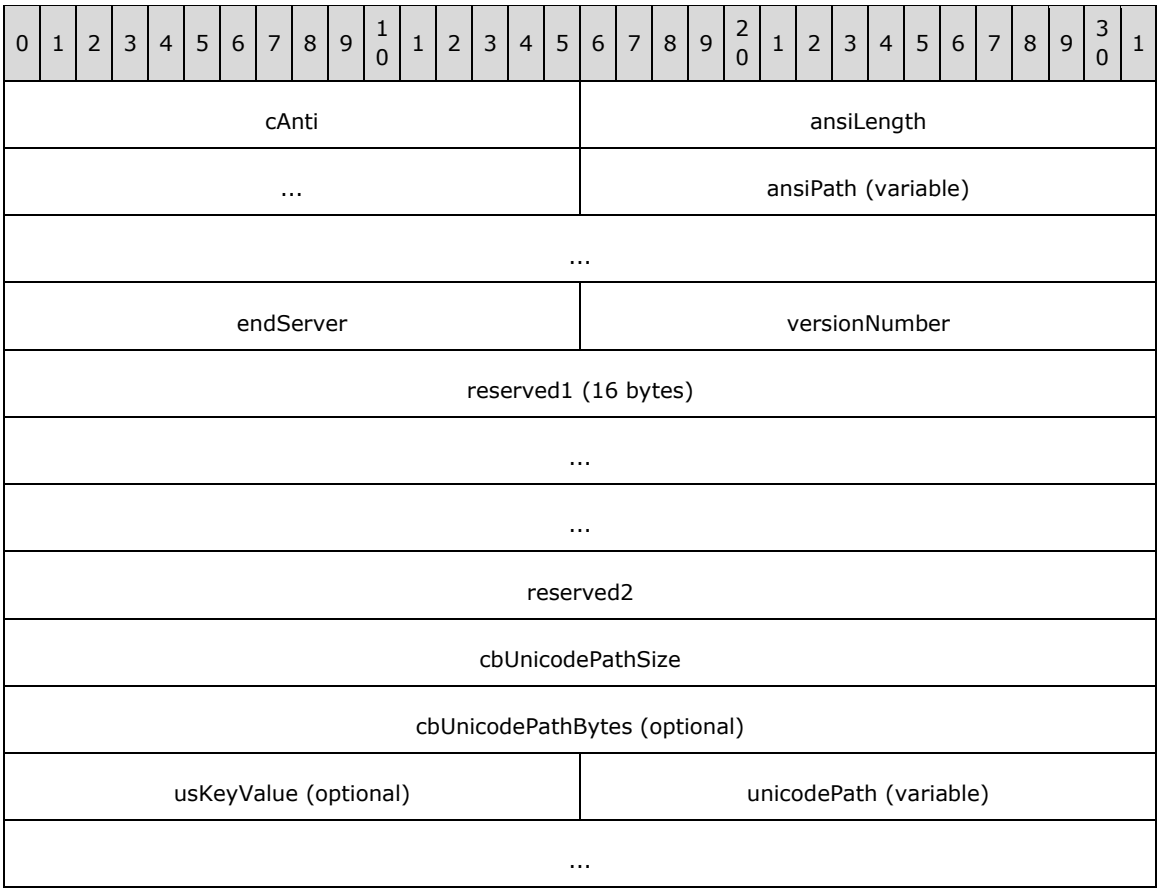
P - createNoEncodeForbiddenCharacters (1 bit): A bit that specifies that URI characters forbidden in [RFC3986] will not be percent-encoded during creation of the URI.

reserved (16 bits): MUST be zero and MUST be ignored.

2.3.7.8 FileMoniker

Referenced by: HyperlinkMoniker

This structure specifies a **file moniker**. For more information about file monikers, see [MSDN-FM].



cAnti (2 bytes): An unsigned integer that specifies the number of **parent directory indicators** at the beginning of the **ansiPath** field.

ansiLength (4 bytes): An unsigned integer that specifies the number of ANSI characters in **ansiPath**, including the terminating NULL character. This value MUST be less than or equal to 32767.

ansiPath (variable): A null-terminated array of ANSI characters that specifies the file path. The number of characters in the array is specified by **ansiLength**.

endServer (2 bytes): An unsigned integer that specifies the number of **Unicode** characters used to specify the server portion of the path if the path is a **UNC** path (including the leading "\\"). If the path is not a UNC path, this field MUST equal 0xFFFF.

versionNumber (2 bytes): An unsigned integer that specifies the version number of this file moniker serialization implementation. MUST equal 0xDEAD.

reserved1 (16 bytes): MUST be zero and MUST be ignored.

reserved2 (4 bytes): MUST be zero and MUST be ignored.

cbUnicodePathSize (4 bytes): An unsigned integer that specifies the size, in bytes, of **cbUnicodePathBytes**, **usKeyValue**, and **unicodePath**.

If the file path specified in **ansiPath** cannot be completely specified by ANSI characters, the value of this field MUST be equal to the size, in bytes, of the path as a Unicode string (without a terminating NULL character) + 6. If the path can be fully specified in ANSI characters then the value of this field MUST be set to zero.

If the value of this field is greater than zero, then the **cbUnicodePathBytes, usKeyValue** and **unicodePath** fields MUST exist.

If the value of this field is zero, then the **cbUnicodePathBytes**, **usKeyValue**, and **unicodePath** fields MUST NOT exist.

cbUnicodePathBytes (4 bytes): An optional unsigned integer that specifies the size, in bytes, of the **unicodePath** field. This field exists if and only if **cbUnicodePathSize** is greater than zero.

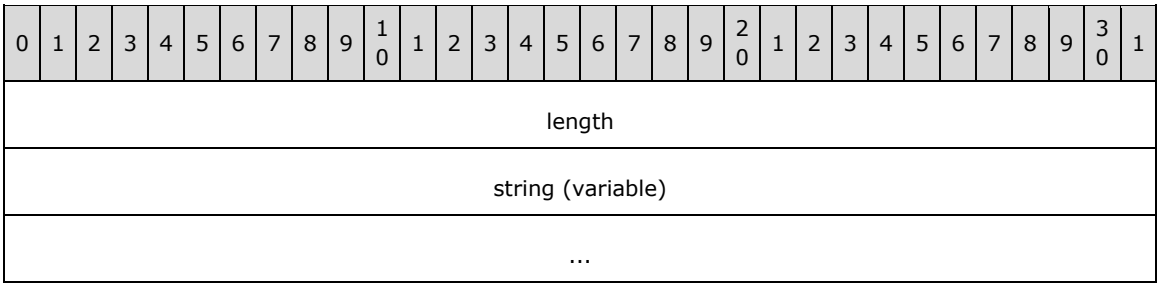
usKeyValue (2 bytes): An optional unsigned integer that MUST be 3 if present. This field exists if and only if **cbUnicodePathSize** is greater than zero.

unicodePath (variable): An optional array of Unicode characters that specifies the complete file path. This path MUST be the complete Unicode version of the file path specified in **ansiPath** and MUST include additional Unicode characters that cannot be completely specified in ANSI characters. The number of characters in this array is specified by **cbUnicodePathBytes**/2. This array MUST NOT include a terminating NULL character. This field exists if and only if **cbUnicodePathSize** is greater than zero.

2.3.7.9 HyperlinkString

Referenced by: Hyperlink Object

This structure specifies a string for a **hyperlink**.



length (4 bytes): An unsigned integer that specifies the number of **Unicode** characters in the **string** field, including the null-terminating character.

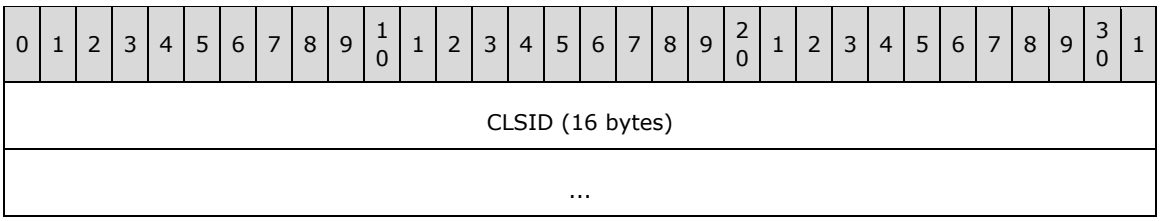
string (variable): A null-terminated array of Unicode characters. The number of characters in the array is specified by the **length** field.

2.3.8 MsoEnvelope

This section specifies structures related to the **MsoEnvelope** structure (section 2.3.8.2).

2.3.8.1 MsoEnvelopeCLSID

A structure that specifies the type of data in **EnvelopeData** based on the value of **CLSID**.



...
EnvelopeData (variable)
...

CLSID (16 bytes): A **GUID**, as specified by [MS-DTYP], that specifies the type of data in **EnvelopeData**. If this GUID equals { 0x0006F01A, 0x0000, 0x0000, { 0xC0, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x46 } }, then the data in **EnvelopeData** is specified by **MsoEnvelope** structure (section 2.3.8.2). If not, then the data in **EnvelopeData** is out of scope for this document.

EnvelopeData (variable): An array of bytes that is either specified by **MsoEnvelope** structure (section 2.3.8.2) or is out of scope, depending on the value of CLSID.

2.3.8.2 MsoEnvelope

A structure that contains information about the email message being sent.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Ver																															
LastSentTime																															
FlagStatus																															
ReplyTime																															
RequestStrSize																	RequestStr (variable)														
...																															
SentRepresentingEntryIdSize																															
SentRepresentingEntryId (variable)																															
...																															
SentRepresentingNameSize																	SentRepresentingName (variable)														
...																															
InetAcctStampSize																	InetAcctStamp (variable)														
...																															
InetAcctNameSize																	InetAcctName (variable)														

...	
ExpiryTime	
DeferredDeliveryTime	
DeleteAfterSubmit	
SecurityFlags	
OriginatorDeliveryReportRequested	
ReadReceiptRequested	
CategoriesStrSize	CategoriesStr (variable)
...	
Sensitivity	
Importance	
SubjectSize	Subject (variable)
...	
VotingOptionsSize	VotingOptions (variable)
...	
ReplyRecipients (variable)	
...	
ContactLinkRecipients (variable)	
...	
Recipients (variable)	
...	
Attachments (variable)	
...	
IntroText (variable)	
...	

Ver (4 bytes): An unsigned integer that specifies the version of this envelope.

LastSentTime (4 bytes): A signed integer that specifies the last time this email message was sent. Time is represented as the number of minutes since 12:00 AM January 1, 1601. The value MUST be greater than or equal to 0 and less than or equal to 0x5AE980E0. The default value is 0x5AE980E0, which means no time was specified.

FlagStatus (4 bytes): An unsigned integer that specifies the follow-up status of this email message. Values are zero for not flagged for follow-up, 1 for flagged for follow-up, and 2 for completed follow-up.

ReplyTime (4 bytes): A signed integer that specifies when the last reply was received for this email message. The time is represented as the number of minutes since 12:00 AM January 1, 1601. The value MUST be greater than or equal to zero and less than or equal to 0x5AE980E0. The default value is 0x5AE980E0, which means no time was specified.

RequestStrSize (2 bytes): An unsigned integer that contains the size, in characters, of **RequestStr**.

RequestStr (variable): A character array that contains the request string. For example, if the sender has set a flag on this email message with "No Response Necessary", this string contains "No Response Necessary". If **Ver** is 6, this is an **ANSI character set** array. If **Ver** is 8, this is a **Unicode** character array.

SentRepresentingEntryIdSize (4 bytes): An unsigned integer that contains the size, in bytes, of **SentRepresentingEntryId**.

SentRepresentingEntryId (variable): An array of bytes that specifies the PR_SENT_REPRESENTING_ENTRYID as specified in [MS-OXPROPS].

SentRepresentingNameSize (2 bytes): An unsigned integer that contains the size, in characters, of **SentRepresentingName**.

SentRepresentingName (variable): A character array that specifies the PR_SENT_REPRESENTING_NAME as specified in [MS-OXPROPS]. If **Ver** is 6, this is an ANSI character set array. If **Ver** is 8, this is a Unicode character array.

InetAcctStampSize (2 bytes): An unsigned integer that contains the size, in characters, of **InetAcctStamp**.

InetAcctStamp (variable): A character array that specifies the **dispidInetAcctStamp** as specified in [MS-OXPROPS]. If **Ver** is 6, this is an ANSI character set array. If **Ver** is 8, this is a Unicode character array.

InetAcctNameSize (2 bytes): An unsigned integer that contains the size, in characters, of **InetAcctName**.

InetAcctName (variable): A character array that contains the **dispidInetAcctName** as specified in [MS-OXPROPS]. If **Ver** is 6, this is an ANSI character set array. If **Ver** is 8, this is a Unicode character array.

ExpiryTime (4 bytes): A signed integer that specifies the time this email message expires. The time is represented as the number of minutes since 12:00 AM January 1, 1601. The value MUST be greater than or equal to 0 and less than or equal to 0x5AE980E0. The default value is 0x5AE980E0, which means no time was specified.

DeferredDeliveryTime (4 bytes): A signed integer that specifies the time to send this email message. The time is represented as the number of minutes since 12:00 AM January 1, 1601. The value MUST be greater than or equal to 0 and less than or equal to 0x5AE980E0. The default value is 0x5AE980E0, which means no time was specified.

DeleteAfterSubmit (4 bytes): An unsigned integer that specifies whether to delete the message after a recipient has submitted a reply. Values are zero for no, and 1 for yes. MUST be zero or 1.

SecurityFlags (4 bytes): A bit field that specifies the security settings for this email message. The least significant bit specifies whether the email message is signed, and the next bit specifies whether the email message is encrypted. All other bits MUST be zero.

OriginatorDeliveryReportRequested (4 bytes): An unsigned integer that specifies whether to send a delivery receipt to the sender. Values are zero for no and 1 for yes. MUST be zero or 1.

ReadReceiptRequested (4 bytes): An unsigned integer that specifies whether to send a read receipt to the sender. Values are zero for no and 1 for yes. MUST be zero or 1.

CategoriesStrSize (2 bytes): An unsigned integer that contains the size, in characters, of **CategoriesStr**.

CategoriesStr (variable): A character array that contains the category for this email message. For example, "Business" or "Favorites". If **Ver** is 6, this is an ANSI character set array. If **Ver** is 8, this is a Unicode character array.

Sensitivity (4 bytes): An unsigned integer that specifies the sensitivity of the email message. Values are zero for Normal, 1 for Personal, 2 for Private, and 3 for Confidential. MUST be less than 4.

Importance (4 bytes): An unsigned integer that specifies the importance flag of the email message. Values are 0 for Low, 1 for Normal, and 2 for High. MUST be less than 3.

SubjectSize (2 bytes): An unsigned integer that contains the size, in characters, of **Subject**.

Subject (variable): A character array that contains the subject of the email message. If **Ver** is 6, this is an ANSI character set array. If **Ver** is 8, this is a Unicode character array.

VotingOptionsSize (2 bytes): An unsigned integer that contains the size, in characters, of the **VotingOptions**.

VotingOptions (variable): An ANSI character array that contains the voting option choices separated by a semicolon. For example, "Yes;No;Maybe".

ReplyRecipients (variable): An **EnvRecipientCollection** (section 2.3.8.3) that contains the recipients of replies to this email message.

ContactLinkRecipients (variable): An **EnvRecipientCollection** (section 2.3.8.3) that contains the sender contacts to whom the email message is to be sent. This is not present if **Ver** does not equal 8.

Recipients (variable): An **EnvRecipientCollection** (section 2.3.8.3) that contains the recipients to receive this email message.

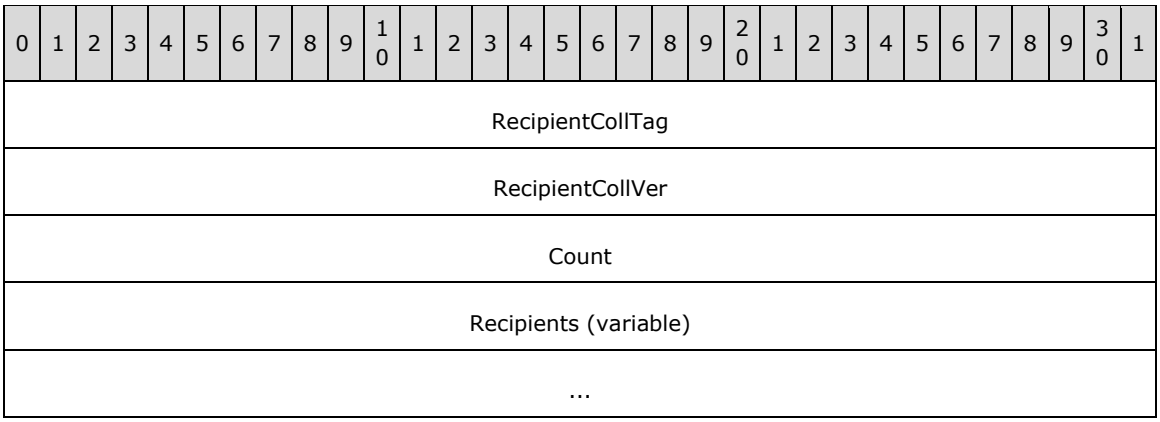
Attachments (variable): An **EnvAttachmentCollection** (section 2.3.8.17) that contains information about attachments for this email message.

IntroText (variable): If **Ver** equals 8, this is an **IntroText** (section 2.3.8.19). This is not present if **Ver** does not equal 8.

2.3.8.3 EnvRecipientCollection

Referenced by: MsoEnvelope

A collection that contains information about recipients for an email message.



RecipientCollTag (4 bytes): An unsigned integer that contains a version tag for the recipient collection. MUST be 0xDCCA0123.

RecipientCollVer (4 bytes): An unsigned integer that contains a version for the recipient collection. MUST be 1.

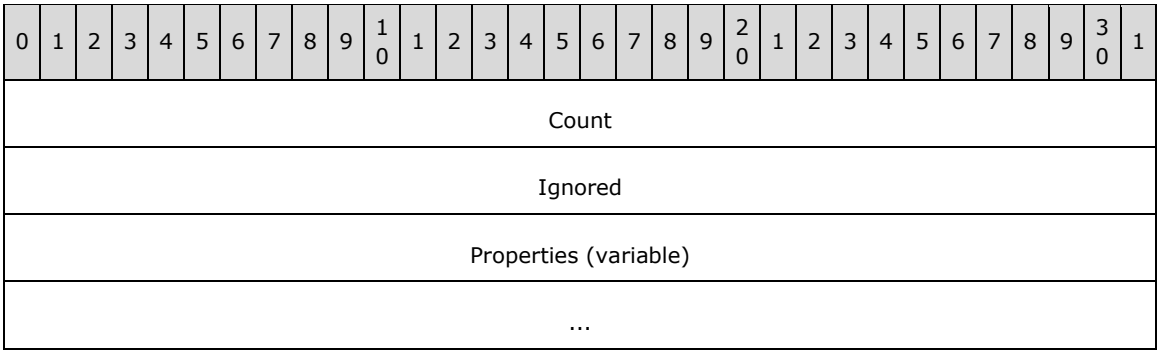
Count (4 bytes): An unsigned integer that contains the number of recipients in the **Recipients** array.

Recipients (variable): An array of **EnvRecipientProperties** (section 2.3.8.4) that contains the properties for each recipient.

2.3.8.4 EnvRecipientProperties

Referenced by: EnvRecipientCollection

An **EnvRecipientProperties** structure that contains the properties of a single email message recipient.



Count (4 bytes): An unsigned integer that contains the number of properties for this recipient.

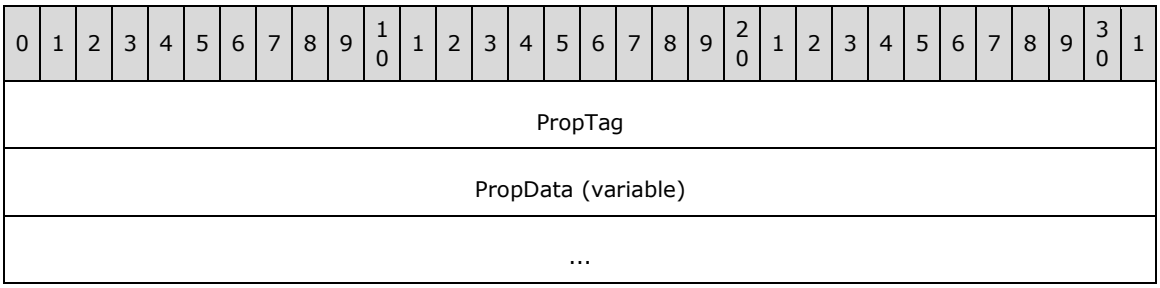
Ignored (4 bytes): MUST be ignored.

Properties (variable): An array of **EnvRecipientProperty** (section 2.3.8.5) that has the size of **Count**.

2.3.8.5 EnvRecipientProperty

Referenced by: EnvRecipientProperties

A single property for an email message recipient. These properties are specified in [MS-OXCDATA]. Only the sizes of the properties are described here.



PropTag (4 bytes): An unsigned integer that contains the property identifier tag.

PropData (variable): An **EnvRecipientPropertyBlob** (section 2.3.8.6), the type of which is specified by **PropTag**.

2.3.8.6 EnvRecipientPropertyBlob

Referenced by: EnvRecipientProperty

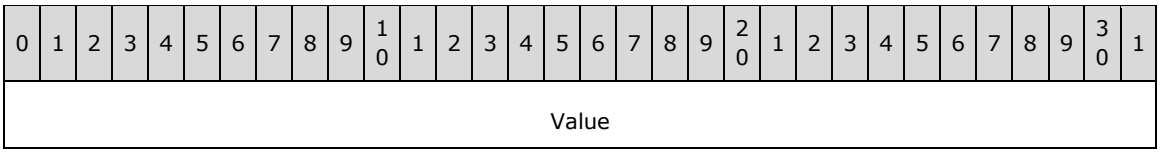
A structure that describes the size of the data for a single recipient property. The type of this structure is specified by the **PropTag** of a **EnvRecipientProperty** (section 2.3.8.5).

Value	Meaning
0x0003	The property data is a **PT_LONG** (section 2.3.8.7).
0x0001	The property data is a **PT_NULL** (section 2.3.8.8).
0x000B	The property data is a **PT_BOOLEAN** (section 2.3.8.9).
0x0040	The property data is a **PT_SYSTIME** (section 2.3.8.10).
0x000A	The property data is a **PT_ERROR** (section 2.3.8.11).
0x001E	The property data is a **PT_STRING8** section 2.3.8.12).
0x001F	The property data is a **PT_UNICODE** (section 2.3.8.13).
0x0102	The property data is a **PT_BINARY** (section 2.3.8.14).
0x101E	The property data is a **PT_MV_STRING8** (section 2.3.8.15).
0x1102	The property data is a **PT_MV_BINARY** (section 2.3.8.16).

2.3.8.7 PT_LONG

Referenced by: EnvRecipientPropertyBlob

A recipient property that is an unsigned integer as specified in [MS-OXCDATA].

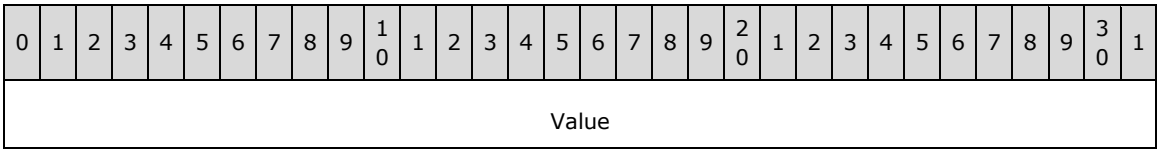


Value (4 bytes): The value for this property as specified in [MS-OXCDATA].

2.3.8.8 PT_NULL

Referenced by: EnvRecipientPropertyBlob

A recipient property that is an unsigned integer as specified in [MS-OXCDATA].

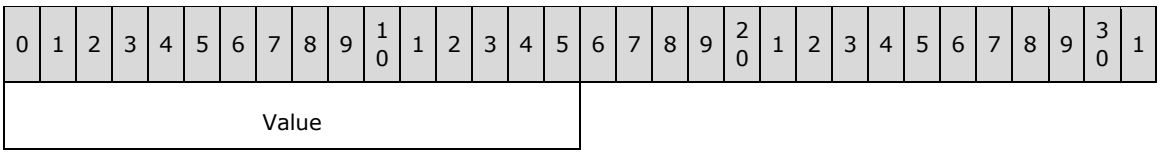


Value (4 bytes): The value for this property as specified in [MS-OXCDATA].

2.3.8.9 PT_BOOLEAN

Referenced by: EnvRecipientPropertyBlob

A recipient property that is an unsigned integer as specified in [MS-OXCDATA].

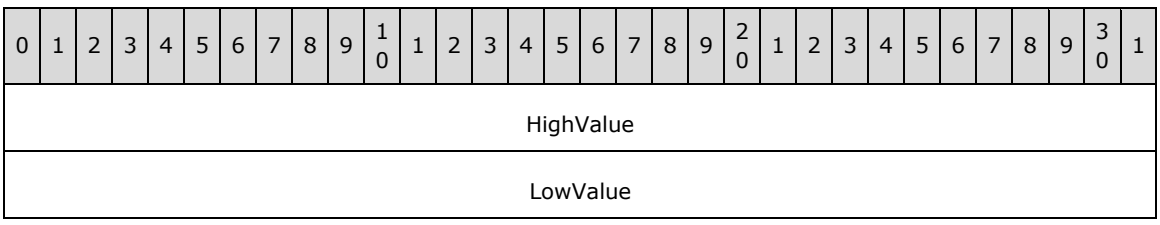


Value (2 bytes): The value of the property as specified in [MS-OXCDATA].

2.3.8.10 PT_SYSTIME

Referenced by: EnvRecipientPropertyBlob

A recipient property that is a **FILETIME** value as specified in [MS-OXCDATA].



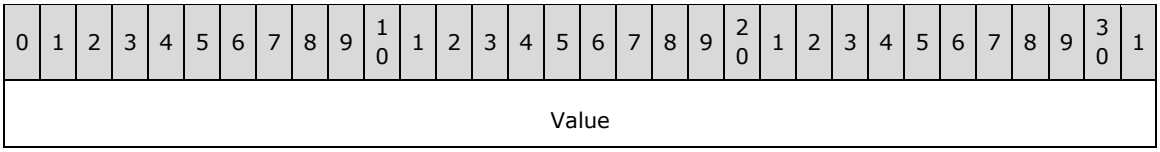
HighValue (4 bytes): The first part of the property's value.

LowValue (4 bytes): The second part of the property's value.

2.3.8.11 PT_ERROR

Referenced by: EnvRecipientPropertyBlob

A recipient property that is an unsigned integer as specified in [MS-OXCDATA].

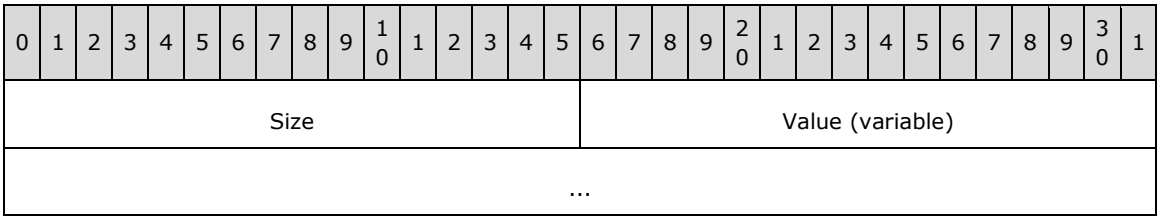


Value (4 bytes): The value of the property as specified in [MS-OXCDATA].

2.3.8.12 PT_STRING8

Referenced by: EnvRecipientPropertyBlob, PT_MV_STRING8

A recipient property that is an **ANSI character set** string as specified in [MS-OXCDATA].



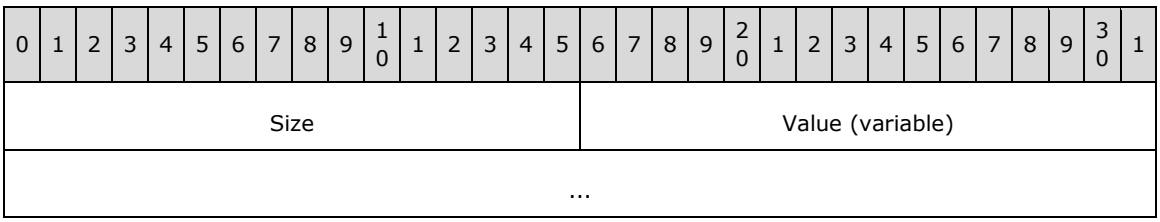
Size (2 bytes): An unsigned integer that specifies the size, in bytes, of **Value**.

Value (variable): An array of characters from the ANSI character set as specified in [MS-OXCDATA].

2.3.8.13 PT_UNICODE

Referenced by: EnvRecipientPropertyBlob

A recipient property that is a **Unicode** string as specified in [MS-OXCDATA].



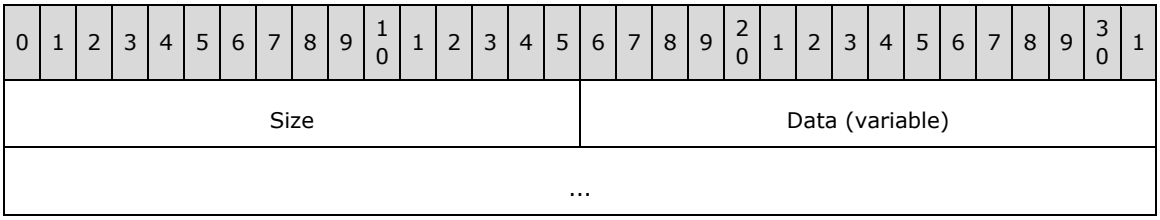
Size (2 bytes): An unsigned integer that specifies the size, in bytes, of **Value**.

Value (variable): A Unicode string as specified in [MS-OXCDATA].

2.3.8.14 PT_BINARY

Referenced by: EnvRecipientPropertyBlob, PT_MV_BINARY

A recipient property that is an array of bytes as specified in [MS-OXCDATA].



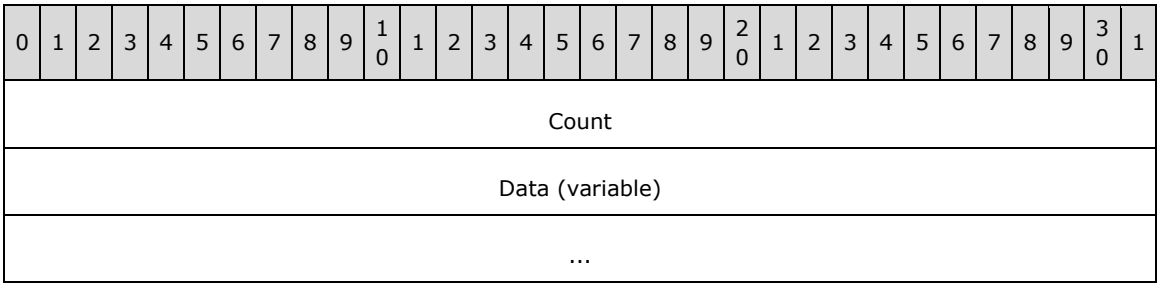
Size (2 bytes): An unsigned integer that contains the size, in bytes, of **Data**.

Data (variable): An array of bytes as specified in [MS-OXCDATA].

2.3.8.15 PT_MV_STRING8

Referenced by: EnvRecipientPropertyBlob

A recipient property that consists of multiple **ANSI character set** strings as specified in [MS-OXCDATA].



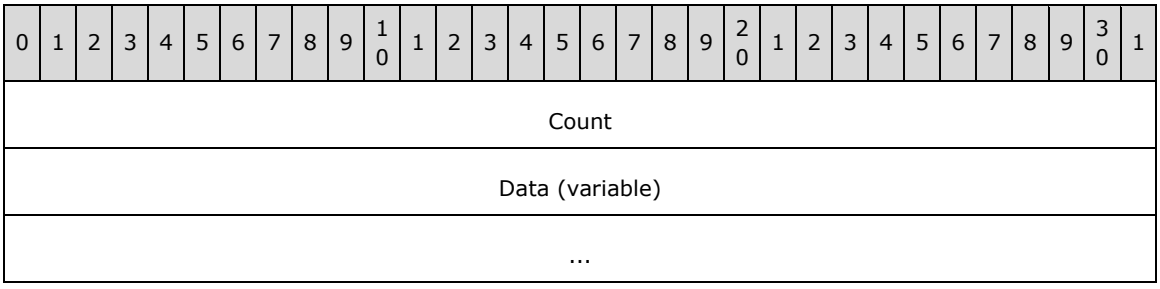
Count (4 bytes): An unsigned integer that specifies the number of elements in **Data**.

Data (variable): An array of **PT_STRING8** (section 2.3.8.12) as specified in [MS-OXCDATA].

2.3.8.16 PT_MV_BINARY

Referenced by: EnvRecipientPropertyBlob

A recipient property that consists of multiple byte arrays as specified in [MS-OXCDATA].



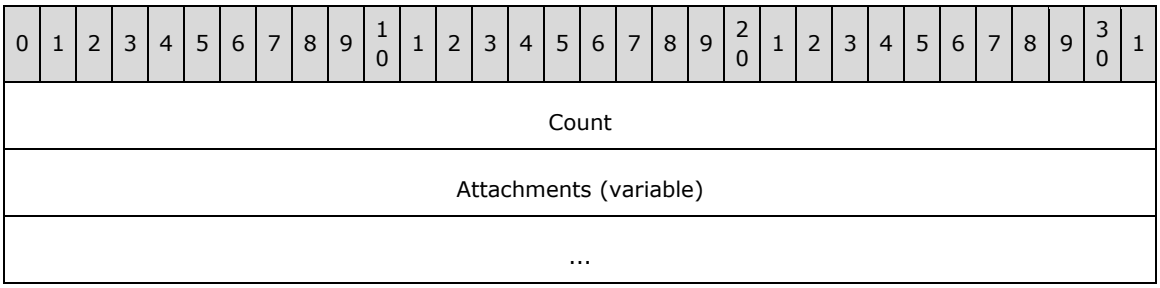
Count (4 bytes): An unsigned integer that specifies the number of elements in **Data**.

Data (variable): An array of **PT_BINARY** (section 2.3.8.14) as specified in [MS-OXCDATA].

2.3.8.17 EnvAttachmentCollection

Referenced by: MsoEnvelope

A collection that contains the data for each of the attachments for an email message.



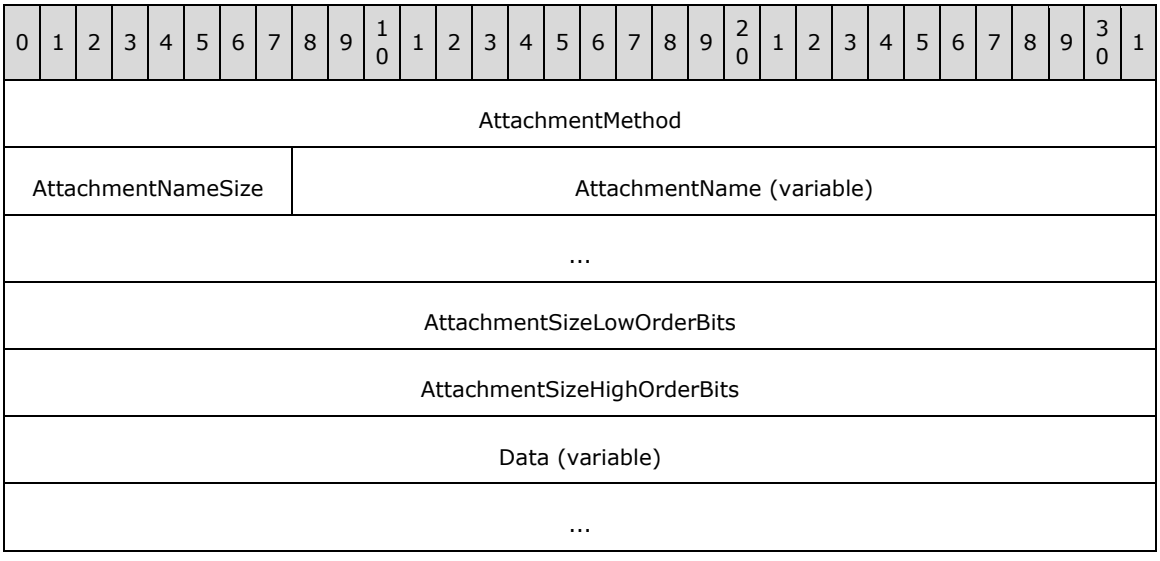
Count (4 bytes): An unsigned integer that contains the number of attachments for an email message.

Attachments (variable): An array of **EnvAttachment** (section 2.3.8.18) of size **Count** that contains the data for each attachment.

2.3.8.18 EnvAttachment

Referenced by: EnvAttachmentCollection

A structure that contains the data for a single email message attachment.



AttachmentMethod (4 bytes): An unsigned integer that represents the MAPI property **PR_ATTACH_METHOD**, as specified in [MS-OXPROPS].

AttachmentNameSize (1 byte): A byte that contains the size, in characters, of **AttachmentName**.

AttachmentName (variable): A **Unicode** character array that contains the file name of the attachment.

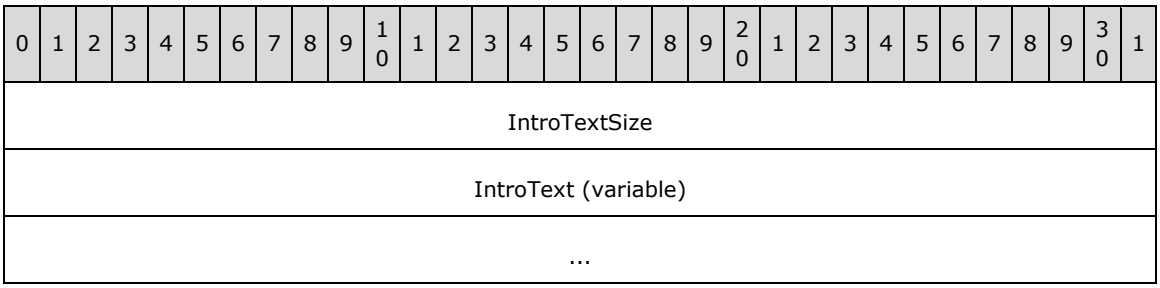
AttachmentSizeLowOrderBits (4 bytes): An unsigned integer that contains the low-order bits for the file size of the attachment.

AttachmentSizeHighOrderBits (4 bytes): An unsigned integer that contains the high-order bits for the file size of the attachment.

Data (variable): An array of bytes that contains the content of the attachment.

2.3.8.19 IntroText

Specifies the text of the introduction for the document when sent as an email message.



IntroTextSize (4 bytes): An unsigned integer that specifies the size, in bytes, of **IntroText**. MUST be evenly divisible by two.

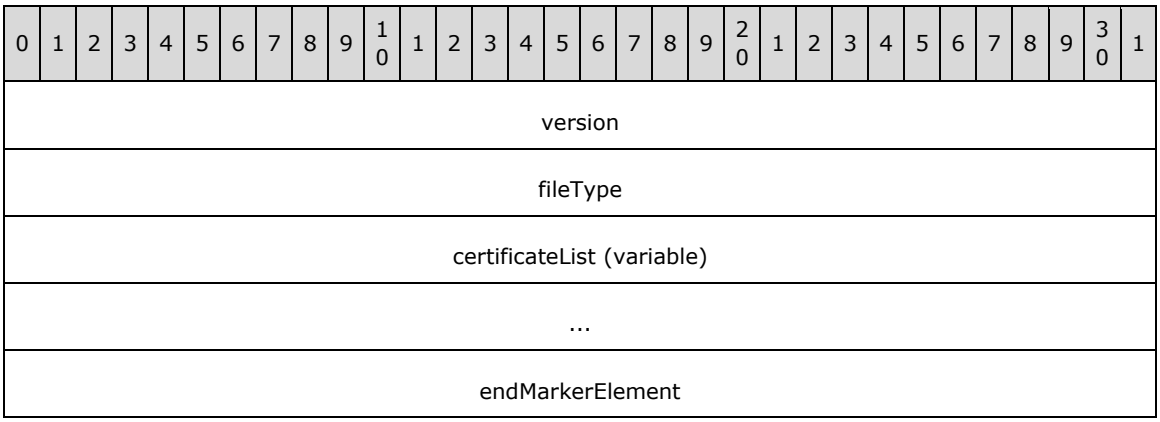
IntroText (variable): A **Unicode** character array that specifies the introduction text.

2.3.9 Document Signature Serialized Certificate Store Structure

This section specifies a serialized **digital certificate store** that is persisted as part of a collection of document signatures, as specified in [MS-OFFCRYPTO] section 2.5.

2.3.9.1 DocSigSerializedCertStore

The serialized **digital certificate store** specifies structures for storing a digital certificate store containing zero or more **digital certificates** and, optionally, a list of properties associated with each certificate.



...
...

version (4 bytes): An unsigned integer identifying the version of the structure. MUST be 0x00000000.

fileType (4 bytes): An unsigned integer that specifies the type of data contained in the structure. MUST be the value 0x54524543.

certificateList (variable): An array of **CertStoreCertificateGroup** (section 2.3.2.5.4). This array can contain zero or more elements. Each element of the array specifies a digital certificate and a collection of optional properties associated with the certificate. Elements of this array are read and processed until a **CertStoreCertificateGroup.SerializedPropertyEntry.id** is read with the unsigned integer value 0x00000000, which specifies the end of this array and the beginning of the **endMarkerElement** field. The terminating **CertStoreCertificateGroup.SerializedPropertyEntry.id** value read is actually the **endMarkerElement.id** field and not part of this field.

endMarkerElement (12 bytes): An **EndElementMarkerEntry** structure (section 2.3.2.5.2) specifying the end of the structure.

2.4 Common Algorithms

2.4.1 Unicode String to Unsigned Integer Hash

Given a null-terminated **Unicode** string, a 4-byte unsigned integer **hash** value can be obtained by performing the following algorithm:

1. Set a 4-byte unsigned integer value **dwHash** equal to 0x00000000.
2. Truncate the **input string** to 255 characters plus the null-terminating character.
3. Convert the **input string** to lower case.
4. For each set of 2 characters (**Char1**, **Char2**) from position 0 to the end of the **input string** (for example, [0,1], [2,3], [4,5], …, [255, <null>]) perform the following steps.

   Example:

   ```
   for (i=0; i<wcslen(inputString); i+=2)
       Char1 = inputString[i];
       Char2 = inputString[i+1];
   ```

   1. Set an unsigned 4-byte integer value, **dwNext**, equal to the bitwise OR of **Char2** shifted 16 bits to the high order and **Char1**.

      Example: dwNext = (Char2 << 16) | Char1;

   2. Set **dwHash** equal to the exclusive **OR** between **dwHash** and **dwNext**.

      Example: dwHash = dwHash ^ dwNext;

5. The result of the algorithm is the value **dwHash**.

2.4.2 Hyperlink Hash

Given the two null-terminated **Unicode** hyperlink strings, hlink1 (**hyperlink target**) and hlink2 (**hyperlink location**), an unsigned 4-byte integer **hash** can be obtained by performing the following algorithm:

1. Hash **hlink1** as specified in Unicode String to Unsigned LONG Hash (section 2.4.1), and store the resulting unsigned 4-byte integer hash in **dwHash1**.

2. Hash **hlink2** as specified in Unicode String to Unsigned LONG Hash (section 2.4.1), and store the resulting unsigned 4-byte integer hash in **dwHash2**.

3. The result of the algorithm is the value obtained by the exclusive **OR** of **dwHash1** and **dwHash2**.

    Example: dwHash = dwHash1 ^ dwHash2;

2.4.3 MsoCrc32Compute

The purpose of this algorithm is to calculate a **cyclic redundancy check (CRC)** checksum. Given a data stream, this function computes a 32-bit checksum using the polynomial $x^{32}+x^7+x^5+x^3+x2+x+1$. The algorithm uses **big-endian** bit ordering in a byte (most significant bit first). Normal representation for a polynomial is xAF.

2.4.3.1 Caching Algorithm

The caching algorithm is run once and its purpose is to initiate **Cache** – an array of 256 integers.

```
SUBROUTINE InitCrcCache()
    IF Called once
        RETURN
    END IF
    FOR each Index from 0 to 255
        SET Value to Index
        SHIFT Value left by 24 bits
        FOR Bit from 0 to 7
            IF highest (31st) bit of Value is set

                SHIFT Value left by 1 bit
                Bitwise exclusive-OR Value with xAF
            ELSE
                SHIFT Value left by 1 bit
            END IF
        END FOR
        AND Value with 0xFFFF
        SET Cache[Index] to Value
    END LOOP
END SUBROUTINE
```

2.4.3.2 CRC Computation

Given a **stream** of data, or an array of bytes, this function computes a 32-bit checksum. The caller specifies the initial value of the checksum (CrcValue). Note that the caller can split the stream of data into substreams and call this subroutine for each of the substreams, passing in the result of the **CRC** computation from the previous substream as CrcValue.

```
SUBROUTINE CRC(CrcValue, Array)
    CALL InitCrcCache()
    FOR each Byte from Array
        SET Index to CrcValue
        SHIFT Index right 24 bits
```

```
            Bitwise exclusive-OR Index with Byte
            SHIFT CrcValue left 8 bits
            Bitwise exclusive-OR CrcValue with Cache[Index]
        END LOOP
        RETURN CrcValue
    END SUBROUTINE
```

2.4.4 Date/Time Format from Format Index

This section specifies the algorithm that produces a date/time format given a format index and an **LCID** that specifies a **locale**.

Note: In this section, a string *s*, which consists of a sequence of the same symbol *x*, is defined to occur in a string *t* where and only where *t* contains a sequence of *x* that is the exact same length as *s*. For example, the substring "dd" occurs in "MM/dd/yyyy" but not in "MM/dddd/yyyy".

2.4.4.1 Format Indices

This section lists and describes the format indices that can be used as inputs to the algorithm. The descriptions given are generalized; the actual format produced can vary from the description, depending on the input **locale**.

Format index	Format description
0	Short date.
1	Long date.
2	Long date without weekday.
3	Alternate short date.
4	ISO standard date.
5	Short date with abbreviated month.
6	Short date with slashes.
7	Alternate short date with abbreviated month.
8	English date.
9	Month and year.
10	Abbreviated month and year.
11	Date and 12-hour time.
12	Date and 12-hour time with seconds.
13	12-hour time.
14	12-hour time with seconds.
15	24-hour time.
16	24-hour time with seconds.
17	Used for Japanese and Chinese only.
18	Used for Chinese only.
19	Used for Chinese only.

2.4.4.2 Base Format Strings

This section specifies a set of base strings which are used to generate some formats. The value of each string is determined by the system's **locale settings** for the input **locale**.

String name	Meaning
Base short date	Specifies a date format which can be used in a context where a short string representation of a date is desired.
Base long date	Specifies a date format which can be used in a context where a detailed string representation of a date is desired.
Base time	Specifies a time format.
Time separator	Specifies the symbols which separate minutes from hours or seconds from minutes in a string representation of a time.

2.4.4.3 Retrieve Format

This section specifies the algorithms used for retrieving the format based on the format index. This section is divided into sections that specify the algorithm for a specific set of input **locales**. Note that the format retrieved using an algorithm in this section is not the final format in every case; the format is still subject to format exceptions (section 2.4.4.4).

2.4.4.3.1 Chinese Formats

When the input **locale** is Chinese, the format is retrieved from the following lookup table.

Format index	Format
0	Base short date
1	Base long date
2	yyyy年M月d日星期W
3	yyyy年M月d日
4	yyyy/M/d
5	yy.M.d
6	yyyy年M月d日星期W
7	yyyy年M月d日
8	yyyy年M月d日星期W
9	yyyy年M月
10	yyyy年M月
11	h时m分s秒
12	h时m分
13	H时m分
14	h时m分
15	AMPMh时m分
16	AMPMh时m分
17	EEEE年O月A日
18	EEEE年O月A日星期W
19	EEEE年O月

2.4.4.3.2 Hindi Formats

When the input **locale** is Hindi, the format is retrieved from the following lookup table.

Format index	Format
0	aa/o/g

Format index	Format
1	dddd, d MMMM yyyy
2	dd MMMM yyyy
3	aa/o/gg
4	gg-o-aa
5	d-MMMM-yyyy
6	aa.o.g
7	aa MMMM. gg
8	aa MMMM gg
9	MMMM YY
10	MMMM-g
11	aa/o/g r:i
12	aa/o/g r:i:c
13	r:i am/pm
14	r:i:c am/pm
15	r:i
16	r:i:c

2.4.4.3.3 Japanese Formats

When the input **locale** is Japanese, the format is retrieved from the following lookup table.

Format index	Format
0	Base short date
1	Base long date
2	ggge年M月d日(aaa)
3	yyyy年M月d日
4	yyyy/M/d
5	gggE年O月A日
6	EE年O月A日(aaa)
7	ggge年M月d日
8	yyyy年M月d日(aaa)
9	ggge年M月
10	yyyy年M月
11	yy/M/d H時m分
12	yy/M/d H時m分s秒
13	AMPMh時m分
14	AMPMh時m分s秒
15	H時m分
16	H時m分s秒
17	ggge年M月d日 aaa曜日

2.4.4.3.4 Korean Formats

When the input **locale** is Korean, the format is retrieved from the following lookup table.

Format index	Format
0	Base short date
1	Base long date
2	yyyy년 M월 d일 aaa요일
3	yyyy년 M월 d일
4	yyyy/M/d
5	yyMMdd
6	gg yyyy년 M월 d일
7	gg yyyy년 M월
8	gg yyyy년 M월 d일
9	gg yyyy년
10	yyyy년 M월
11	yyyy년 M월 d일 AMPM h시 m분
12	yy년 M월 d일 H시 m분 s초
13	AMPM h시 m분
14	AMPM h시 m분 s초
15	H시 m분
16	H시 m분 S초

2.4.4.3.5 Taiwanese Formats

When the input **locale** is Taiwanese, the format is retrieved from the following lookup table.

Format index	Format
0	Base short date
1	Base long date
2	E年O月A日
3	e年M月d日
4	EEEE年O月A日
5	yyyy年M月d日
6	E年O月A日星期W
7	e年M月d日星期W
8	E年O月A日星期W
9	EE年O月A日星期W
10	EEE年O月A日星期W
11	EEEE年O月A日星期W
12	yyyy年M月d日星期W
13	yyyy年M月d日星期W
14	yyyy年M月d日星期W
15	EE年O月A日
16	EEE年O月A日

2.4.4.3.6 Thai Formats

When the input **locale** is Thai, the format is retrieved from the following lookup table.

Format index	Format
0	dd/MM/bb
1	"□□□"□□□□"□□□" d ด ด ด ด "พ .ศ ." bbbb
2	d ด ด ด ด bbbb
3	dd ด ด ด bb
4	dd/MM/bbbb"
5	ด ด ด ด bb
6	dd/MM/bb HH:mm "น ."
7	dd/MM/bb HH:mm:ss "น ."
8	ว ว /ด ด /ป ป
9	"□□□"□□□□"□□□" ว ด ด ด ด "พ .ศ ." ป ป ป ป
10	ว ด ด ด ด ป ป ป ป
11	ด ด ด ป ป
12	ด ด ด ด ป ป
13	ว ว /ด ด /ป ป ม ม :น น "น ."
14	ว ว /ด ด /ป ป ป ป ม ม :น น :ท ท 'น .'
15	ม ม :น น "น ."
16	ม ม :น น :ท ท "น ."

2.4.4.3.7 Yi Formats

When the input **locale** is Yi, the format is retrieved from the following lookup table.

Format index	Format
0	Base short date
1	yyyy[0xA20E]M[0xA1AA]d[0xA44D], w
2	yyyy[0xA20E]M[0xA1AA]d[0xA44D]
3	yyyy.MM.dd
4	yyyy/M/d
5	yy.M.d
6	yyyy/MM/dd
7	yyyy-MM-dd
8	yyyy[0xA20E]M[0xA1AA]d[0xA44D]
9	yyyy[0xA20E]M[0xA1AA]
10	yyyy[0xA20E]M[0xA1AA]
11	yyyy/M/d HH:mm
12	yyyy/M/d HH:mm:ss
13	h:mm AMPM
14	h:mm:ss AMPM
15	HH:mm
16	HH:mm:ss

2.4.4.3.8 Formats for all other Locales

When the input **locale** is any other locale, the format is retrieved by following the steps from the row in the following table corresponding to the input format index.

Format index	Format
0	Base short date.
1	Base long date.
2	Do the following to base long date: ▪ Remove occurrences of "dddd". ▪ Remove the comma symbol (0x002C) and space following "dddd" if present. ▪ Change occurrences of "dd" to "d".
3	Do the following to base short date: ▪ Change occurrences of "yyyy" to "yy". ▪ Change occurrences of "yy" to "yyyy".
4	yyyy-MM-dd
5	If the symbol "y" occurs before the symbol "M" occurs in the base short date, the format is "yy-MMM-d". Otherwise, the format is "d-MMM-yy".
6	If the forward slash symbol (0x002F) occurs in the base short date, the slash symbol is the period symbol (0x002E). Otherwise, the slash symbol is the forward slash (0x002F). A group is an uninterrupted sequence of qualified symbols where a qualified symbol is "d", "M", or "Y". Identify the first three groups that occur in the base short date. The format is formed by appending the three groups together with single slash symbols separating the groups.
7	Do the following to base long date: ▪ Remove occurrences of "dddd". ▪ Remove the comma symbol (0x002C) and space following "dddd" if present. ▪ Change occurrences of "dd" to "d". ▪ For all right-to-left locales and Lao, change a sequence of any length of "M" to "MMM". ▪ For all other locales, change a sequence of any length of "M" to "MMM". ▪ Change occurrences of "yyyy" to "yy".
8	For American English and Arabic, the format is "d MMMM yyyy". For Hebrew, the format is "d MMMM, yyyy". For all other locales, the format is the same as format 6 with the following additional step: Change occurrences of "yyyy" to "yy".
9	Do the following to base long date: ▪ Remove all symbols that occur before the first occurrence of either the "y" symbol or the "M" symbol. ▪ Remove all "d" symbols. ▪ For all locales except Lithuanian, remove all period symbols (0x002E). ▪ Remove all comma symbols (0x002C). ▪ Change occurrences of "yyyy" to "yy".
10	"MMM-yy".
11	Base short date followed by a space, followed by base time with seconds removed. Seconds are removed by removing all "s" symbols and any symbol that directly precedes an "s" symbol that is not an "h" or "m" symbol.
12	Base short date followed by a space, followed by base time.
13	For Hungarian, the format is "am/pm h:mm". For all other locales, the format is "h:mm am/pm". In both cases, replace occurrences of the colon symbol (0x003A) with the time separator.
14	For Hungarian, the format is "am/pm h:mm:ss". For all other locales, the format is "h:mm:ss am/pm". In both cases, replace occurrences of the colon symbol (0x003A) with the

Format index	Format
	time separator.
15	"HH" followed by the time separator, followed by "mm".
16	"HH" followed by the time separator, followed by "mm", followed by the time separator followed by "ss".

2.4.4.4 Apply Format Exceptions

This section specifies any exception operations that are performed on a format after it is retrieved. The table in each subsection lists all format indices for which an exception operation is to be performed for the **locale** specified by that section.

2.4.4.4.1 Bokmål (Norwegian)

Format index	Exception operation
5	Set format to "d. MMM. yyyy".
6	Set format to "d/m yyyy".
7	Set format to "MMM. yy".
8	Set format to "yyyy.mm.dd".
10	Set format to "d. MMM.".

2.4.4.4.2 Czech

Format index	Exception operation
9	Insert "'[0x2019]'" before the last sequence of "y" symbols.

2.4.4.4.3 Danish

Format index	Exception operation
1	Set format to "d. MMMM yyyy".
2	Set format to "yy-MM-dd".
3	Set format to "yyyy.MM.dd".
5	Set format to "MMMM yyyy".
6	Set format to "d.M.yy".
7	Set format to "d/M yyyy".
8	Set format to "dd.MM.yyyy".
9	Set format to "d.M.yyyy".
10	Set format to "dd/MM yyyy".

2.4.4.4.4 Dutch

Format index	Exception operation
9	Insert "'[0x2019]'" before the last sequence of "y" symbols.

2.4.4.4.5 Finnish

Format index	Exception operation
5, 7, 10	Remove any occurrence of "ta". Insert the symbol "k" just after the last occurrence of the symbol "M".

2.4.4.4.6 French Canadian

Format index	Exception operation
3	Set format to "yy MM dd".

2.4.4.4.7 German

Format index	Exception operation
5	Set format to "yy-MM-dd".
7	Set format to "dd. MMM. yyyy".

2.4.4.4.8 Hungarian

Format index	Exception operation
7	Set format to "yy. MMM. dd.".
8	Set format to "'[0x2019]'yy MMM.".
9	Insert "'[0x2019]'" before the last sequence of "y" symbols.

2.4.4.4.9 Italian

Format index	Exception operation
5, 7	If there is no period symbol (0x002E) following the first sequence of "M" symbols, insert a period symbol after the sequence.
8	Set format to "MMM. '[0x2019]'yy".
9	Insert "'[0x2019]'" before the last sequence of "y" symbols.

2.4.4.4.10 Japanese

Format index	Exception operation
1	Replace any occurrence of "dddd" with "aaa曜日".
All	Replace any occurrence of "gg yy" with "gggee". Replace any occurrence of "gg y" with "ggge".

2.4.4.4.11 Kazakh

Format index	Exception operation
1	If there are less than four occurrences of the "d" symbol, append ", dddd".

2.4.4.4.12 Khmer

Format index	Exception operation
1	If there are less than four occurrences of the "d" symbol, add the prefix "dddd" followed by a space.

2.4.4.4.13 Korean

Format index	Exception operation
1	Replace any occurrence of "dddd" with "aaa요일".

2.4.4.4.14 Lao

Format index	Exception operation
1	If there are less than four occurrences of the "d" symbol, add the prefix "dddd'[0x0E97][0x0EB5][0x0EC8]'" followed by a space.

2.4.4.4.15 Lithuanian

Format index	Exception operation
9	Remove all symbols that occur after the last "M" symbol.

2.4.4.4.16 Polish

Format index	Exception operation
7	Set format to "dd MM yy".

2.4.4.4.17 Portuguese

Format index	Exception operation
7	Replace the word "de" along with any surrounding **whitespace** and surrounding single quote symbols (0x0027) with a single space.

2.4.4.4.18 Russian

Format index	Exception operation
9	Remove all symbols that occur after the last "y" symbol.

2.4.4.4.19 Spanish

Format index	Exception operation
7	Replace the word "de" along with any surrounding **whitespace** and surrounding single quote symbols (0x0027) with a single space.
9	If "yy" occurs at the end of the format, replace the "yy" with "yyyy".

2.4.4.4.20 Swedish

Format index	Exception operation
1	Set format to "dddd 'den 'd MMMM yyyy".
4	Set format to "yyMMdd".
5	Set format to "d MMM yy".
6	Set format to "d/M yyyy".
7	Set format to "d MMM –yy".
8	Set format to "M/d/yy".
9	Set format to "MMMM yyyy".
10	Set format to "'den' d MMMM yyyy".
12	Set format to "yy-MM-dd hh:mm".
13	Set format to "h.mm AM/PM".
14	Set format to "h.mm".

2.4.4.4.21 Tibetan

Format index	Exception operation
1	If there are less than 4 occurrences of the "d" symbol, append "'[0x200B]'dddd".
5	Set format to "yy-M-d".
7	Set format to "yyyy.MM.dd".
10	Set format to "yyyy.M".

2.4.4.4.22 Uzbek Cyrillic

Format index	Exception operation
7	If "MMM." occurs at the end of the format, remove the last period symbol (0x002E).

2.4.4.4.23 Vietnamese

Format index	Exception operation
6	Set format to "d-MMMM-yy".

2.4.4.4.24 Bhutanese

Format index	Exception operation
5	Set format to "yy-M-d".
7	Set format to "yyyy.MM.dd".
10	Set format to "yyyy.M".

3 Structure Examples

3.1 Toolbar Customization Examples

The following subsections contain a series of examples of structures persisted in [MS-DOC] and [MS-XLS] when saving custom toolbars to a file.

3.1.1 Toolbar Control Example

This section contains an example for a **TBC** structure, described in [MS-XLS] section 2.6.4, which specifies a modified built-in **toolbar control** of type Button. The toolbar control was originally the Bold toolbar control, but has been modified as follows:

- The toolbar control has the string "MyTooltip" set as a **ToolTip**.
- The toolbar control executes the macro called "MySub" when it is clicked.
- The toolbar control displays a custom icon.

Note that a modified toolbar control is not the same as a **custom toolbar control**. If the toolbar control in this example did not execute a macro when clicked, it would still execute the Bold command. Custom toolbar controls never save the **tbcCmd** field to file and have a **TCID** equal to 0x0001.

If this example were for a **TBC** structure, as described in [MS-DOC] section 2.9.309, then the **TBC** structure would have a field called **cid** of type **Cid**, described in [MS-DOC] section 2.9.34, rather than a field called **tbcCmd**.

The following table shows the top level representation of a **TBC** structure, specified in [MS-XLS] section 2.6.4.

Offset	Size	Structure	Value
0000006F	04C5	TBC **- tbc**	
0000006F	000B	**A**: TBCHeader **- tbch**	
0000007A	0004	TBCCmd **- tbcCmd**	0x0004003A
0000007E	04B6	**B**: TBCData **- tbcd**	

Figure 8: Overview of a TBC structure

tbc: A **TBC** structure, described in [MS-XLS] section 2.6.4.

tbcCmd: A **TBCCmd** structure, described in [MS-XLS] section 2.6.5. Because the toolbar control in this example executes a macro when it is clicked, the value of this structure is not used.

The following table shows the **TBCHeader** structure (section 2.3.1.10) used in this example.

Offset	Size	Structure	Value
0000006F	000B	**A**: TBCHeader **- tbch**	
0000006F	0001	BYTE **- bSignature**	0x03

Offset	Size	Structure	Value
00000070	0001	BYTE **- bVersion**	0x01
00000071	0001	TBCFlags **- bFlagsTCR**	0x00
00000072	0001	BYTE **- tct**	0x01
00000073	0002	WORD **- tcid**	0x0071
00000075	0004	TBCSFlags **- tbct**	0x00AA0020
00000079	0001	BYTE **- bPriority**	0x03

Figure 9: Overview of a TBCHeader structure

bSignature: 0x03 specifies the toolbar control signature number.

bVersion: 0x01 specifies the toolbar control version number.

bFlagsTCR: Structure of type **TBCFlags** (section 2.3.1.11) that specifies toolbar control settings.

tct: 0x01 specifies that this toolbar control is of type Button.

tcid: 0x0071 specifies TCID for this toolbar control. In the Toolbar Control Data table described in [MS-CTXLS] section 2.2, the value 0x0071 corresponds to the Bold toolbar control.

tbct: Structure of type **TBCSFlags** (section 2.3.1.12) that specifies toolbar control settings.

bPriority: 0x03 specifies the toolbar control can be dropped from the **toolbar** when needed, relative to other toolbar controls in the toolbar.

The following table shows the **TBCData** structure (section 2.3.1.13) used in this example.

Offset	Size	Structure
0000007E	04B6	**B**: TBCData **- tbcd**
0000007E	0029	**C**: TBCGeneralInfo **- controlGeneralInfo**
000000A7	048D	**D**: TBCBSpecific **- controlSpecificInfo**

Figure 10: Overview of a TBCData structure

tbcd: A structure of type **TBCData** (section 2.3.1.13).

controlSpecificInfo: Because this is a toolbar control of type **Button**, this field contains a structure of type **TBCBSpecific** (section 2.3.1.17).

The following table shows the **TBCGeneralInfo** structure (section 2.3.1.14) used in this example.

Offset	Size	Structure	Value
0000007E	0029	**C**: TBCGeneralInfo **- controlGeneralInfo**	
0000007E	0001	TBCGIFlags **- bFlags**	0x06
0000007F	0001	WString **- descriptionText**	
00000080	0013	WString **- tooltip**	
00000093	0014	TBCExtraInfo **- extraInfo**	

Offset	Size	Structure	Value
00000093	0001	WString **- wstrHelpFile**	
00000094	0004	LONG **- idHelpContext**	0x00000000
00000098	0001	WString **- wstrTag**	
00000099	000B	WString **- wstrOnAction**	
000000A4	0001	WString **- wstrParam**	
000000A5	0001	BYTE **- tbcu**	0x01
000000A6	0001	BYTE **- tbmg**	0x03

Figure 11: Overview of a TBCGeneralInfo structure

Example notes:

- In this example the **customText** field of the **TBCGeneralInfo** structure (section 2.3.1.14) does not exist because the **bFlags** field has a **bFlags.fSaveText** value equal to 0.

controlGeneralInfo: A structure of type **TBCGeneralInfo** (section 2.3.1.14).

descriptionText: This field exists and contains a **WString** structure (section 2.3.1.4) because **bFlags.fSaveMiscUIStrings** equals 1 in this example. Because this toolbar control does not have descriptive text, the **cLen** field for the **WString** structure (section 2.3.1.4) equals 0x00 and its **data** field does not exist.

tooltip: This field exists and contains a **WString** structure (section 2.3.1.4) because **bFlags.fSaveMiscUIStrings** equals 1 in this example. The **WString** structure (section 2.3.1.4) specifies the string "MyTooltip".

extraInfo: This field exists and contains a **TBCExtraInfo** structure (section 2.3.1.16) because **bFlags.fSaveMiscCustom** equals 1 in this example. The **TBCExtraInfo** structure (section 2.3.1.16) specifies extra information saved for this toolbar control.

extraInfo.wstrHelpFile: This field contains a structure of type **WString** (section 2.3.1.4). Because this toolbar control does not have a help file, the **cLen** field for the **WString** structure (section 2.3.1.4) equals 0x00 and its **data** field does not exist.

extraInfo.idHelpContext: This field is ignored because this toolbar control does not have a help file.

extraInfo.wstrTag: This field contains a structure of type **WString** (section 2.3.1.4). Because this toolbar control does not have a **Tag** string, the **cLen** field for the **WString** structure (section 2.3.1.4) equals 0x00 and its **data** field does not exist.

extraInfo.wstrOnAction: This field contains a structure of type **WString** (section 2.3.1.4). In this example, the toolbar control executes a macro called "MySub". Therefore, this field specifies the "MySub" string.

extraInfo.wstrParam: This field contains a structure of type **WString** (section 2.3.1.4). The **cLen** field for the **WString** structure (section 2.3.1.4) equals 0x00 and its **data** field does not exist because this toolbar control does not have a **Param** string.

extraInfo.tbcu: 0x01 specifies that this toolbar control is applicable when the application is in **OLE** server mode during OLE merging.

extraInfo.tbmg: Because this toolbar control is not of type Popup, this field is ignored.

The following table shows the **TBCBSpecific** structure (section 2.3.1.17) used in this example.

Offset	Size	Structure	Value
000000A7	0423	**D**: TBCBSpecific **- controlSpecificInfo**	
000000A7	0001	TBCBSFlags **- bFlags**	0x88
000000A8	0422	TBCBitmap **- icon**	
000000A8	0004	LONG **- cbDIB**	0x00000428
000000AC	001E	BITMAPINFOHEADER **- biHeader**	
000000AC	0004	LONG **- bData**	0x00000028
000000B0	0002	WORD **- biWidth**	0x0010
000000B2	0002	WORD **- biHeight**	0x0010
000000B4	0001	BYTE **- biPlanes**	0x01
000000B5	0001	BYTE **- biBitCount**	0x20
000000B6	0002	WORD **- biCompression**	0x0000
000000B8	0002	WORD **- biSizeImage**	0x0000
000000BA	0004	LONG **- biXPelsPerMeter**	0x00000000
000000BE	0004	LONG **- biYPelsPerMeter**	0x00000000
000000C2	0004	LONG **- biClrUsed**	0x00000000
000000C6	0004	LONG **- biClrImportant**	0x00000000
000000CA	0400	BYTE **- bitmapData**	
000004CA	006A	TBCBitmap **- iconMask**	

Figure 12: Overview of a TBCBSpecific structure

Example notes:

- For simplicity, a detailed explanation of the **bFlags** and **iconMask** fields have been omitted.
- The **iBtnFace** of the **TBCBSpecific** structure (section 2.3.1.17) does not exist because the **bFlags** field has the value **bFlags.fCustomBtnFace** equal to 0.
- The **wstrAcc** field of the **TBCBSpecific** structure (section 2.3.1.17) does not exist because the **bFlags** field has the **bFlags.fAccelerator** equal to 0.
- The **biHeader.biBitCount** field for the **TBCBitmap** structure (section 2.3.1.1) contained by the **icon** property of the **TBCBSpecific** structure (section 2.3.1.17) is greater than 0x08. Therefore, the **colors** array for the **TBCBitmap** structure (section 2.3.1.1) contained by the **icon** property of the **TBCBSpecific** structure (section 2.3.1.17) does not exist.

controlSpecificInfo: Structure of type **TBCBSpecific** (section 2.3.1.17).

icon: Structure of type **TBCBitmap** (section 2.3.1.1) that specifies the bitmap used as the custom icon for the toolbar control. A **TBCBitmap** structure (section 2.3.1.1) is similar to a **DeviceIndependentBitmap** object, described in [MS-WMF] section 2.2.2.9, with the following differences:

- A **TBCBitmap** (section 2.3.1.1) contains a **cbDib** field used to calculate the total size of the structure. A **DeviceIndependentBitmap** object does not have this field.
- The **biHeader** field of a **TBCBitmap** (section 2.3.1.1) is of type **BITMAPINFOHEADER** (section 2.3.1.2) and has a size of 30 bytes. The **BitmapInfoHeader** object, described in [MS-WMF] section 2.2.2.3, for a **DeviceIndependentBitmap** object has a size of 40 bytes. The difference is caused by the following fields:
  - The **biWidth** field of a **BITMAPINFOHEADER** structure (section 2.3.1.2) is 2 bytes in size. The **Width** field of the **BitmapInfoHeader** object is 4 bytes in size.
  - The **biHeight** field of a **BITMAPINFOHEADER** structure (section 2.3.1.2) is 2 bytes in size. The **Height** field of the **BitmapInfoHeader** object is 4 bytes in size.
  - The **biPlanes** field of a **BITMAPINFOHEADER** structure (section 2.3.1.2) is 1 byte in size. The **Planes** field of the **BitmapInfoHeader** object is 2 bytes in size.
  - The **biBitCount** field of a **BITMAPINFOHEADER** structure (section 2.3.1.2) is 1 byte in size. The **BitCount** field of the **BitmapInfoHeader** object is 2 bytes in size.
  - The **biCompression** field of a **BITMAPINFOHEADER** structure (section 2.3.1.2) is 2 bytes in size. The **Compression** field of the **BitmapInfoHeader** object is 4 bytes in size.
  - The **biSizeImage** field of a **BITMAPINFOHEADER** structure (section 2.3.1.2) is 2 bytes in size. The **ImageSize** field of the **BitmapInfoHeader** object is 4 bytes in size.

icon.cbDIB: 0x00000428 specifies the total size in bytes of the **TBCBitmap** structure (section 2.3.1.1), excluding this field, plus 10. The total size of the **TBCBitmap** structure (section 2.3.1.1) is calculated using the following formula:

Total Size of **TBCBitmap** =

cbDIB – 10 + sizeOf(**cbDIB**) = 0x000000428 - 0xA + 0x4 = 0x000000422

icon.biHeader: Structure of type **BITMAPINFOHEADER** (section 2.3.1.2).

icon.biHeader.bData: This field is always equal to 0x00000028.

icon.biHeader.biWidth: 0x0010 specifies the width of the bitmap in pixels.

icon.biHeader.biHeight: 0x0010 specifies the height of the bitmap in pixels.

icon.biHeader.biPlanes: 0x01 specifies the number of planes for the target device.

icon.biHeader.biBitCount: 0x20 specifies the number of bits per pixel.

icon.biHeader.biCompression: 0x0000 specifies that this is an uncompressed bitmap. Bitmaps specified by **TBCBitmap** (section 2.3.1.1) are always uncompressed.

icon.biHeader.biSizeImage: 0x0000 because the **biCompression** field of the **BITMAPINFOHEADER** structure (section 2.3.1.2) equals 0x0000. This field has a value of 0x0000 and does not specify the size in bytes of the image.

icon.biHeader.biXPelsPerMeter: The value of this field is 0x00000000 and it is ignored.

icon.biHeader.biYPelsPerMeter: The value of this field is 0x00000000 and it is ignored.

icon.biHeader.biClrUsed: 0x00000000 specifies that this bitmap uses the maximum number of colors corresponding to the value of the **biBitCount** field.

icon.biHeader.biClrImportant: 0x00000000 specifies that all colors are required for this bitmap.

icon.bitmapData: Array of bytes that contain the actual bitmap data. This array has **cbDIB** – sizeOf(**colors**) – sizeOf(**biHeader**) -10 bytes.

3.1.2 Toolbar Delta Example

This section contains an example of a **TBDelta** structure, described in [MS-DOC] section 2.9.311, which specifies a **toolbar delta**.

A toolbar delta specifies a change made to a built-in **toolbar**. In this example, the toolbar delta specifies the insertion of a **toolbar control** at the end of the main menu toolbar.

For simplicity, the **TBC** structure, described in [MS-DOC] section 2.9.309, that specifies the toolbar control associated with this **TBDelta** structure has been omitted. However, the toolbar control in this example has the following characteristics:

- The toolbar control is a built-in toolbar control of type Button.
- The **TCID** of the toolbar control equals 0x0071.

The **TBC** structure associated with this toolbar delta is stored in the **rtbdc** array of the **CTBWRAPPER** structure, described in [MS-DOC] section 2.9.49. However, the **TBDelta** structure associated with this toolbar delta is stored in the **rCustomizations** array of the **Customization** structure, described in [MS-DOC] section 2.9.50. And, the **Customization** structure is stored in the **rCustomizations** array of the same **CTBWRAPPER** structure where the **rtbdc** array is located. The following diagram illustrates where the **TBC** structure associated with the **TBDelta** structure in this example is stored.

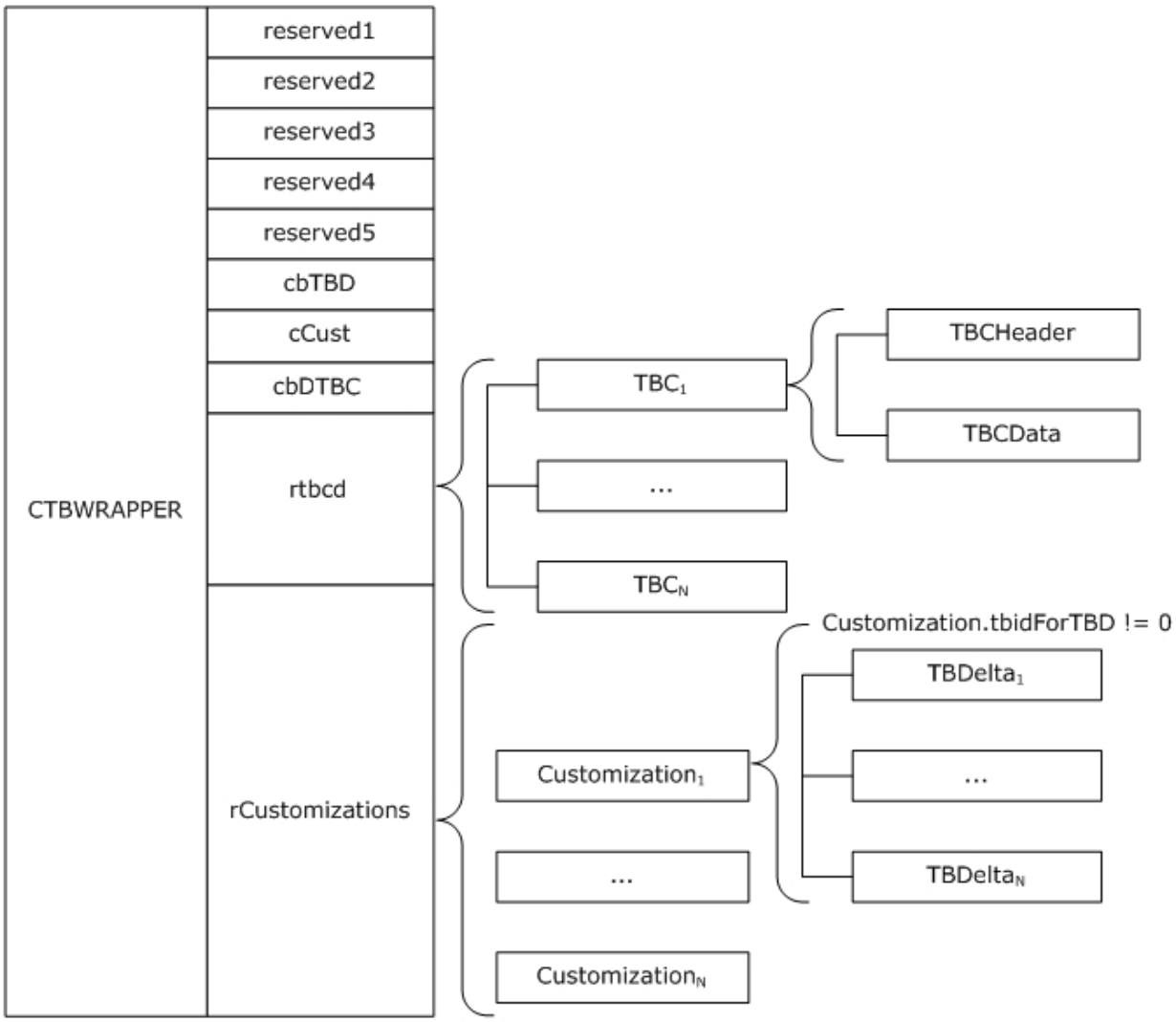


Figure 13: Position of a TBC associated with a TBDelta

The following table shows the **TBDelta** structure described in [MS-DOC] section 2.9.311 used in this example.

Offset	Size	Structure	Value
00000557	0012	TBDelta **- data**	
00000557	2 bits	USHORT **- dopr**	0x1
00000557	1 bit	USHORT **- fAtEnd**	0x1
00000557	5 bits	USHORT **- reserved1**	0x00
00000557	8 bits	USHORT **- ibts**	0x0A
00000559	0004	Cid **- cidNext**	0xFFFFFFFF
0000055D	0004	Cid **- cid**	0x00000169
00000561	0004	ULONG **- fc**	0x0000053E

Offset	Size	Structure	Value
00000565	1 bit	USHORT **- fOnDisk**	0x1
00000565	13 bits	USHORT **- iTB**	0x0000
00000565	1 bit	USHORT **- reserved2**	0x0
00000565	1 bit	USHORT **- fDead**	0x0
00000567	0002	USHORT **- cbTBC**	0x0001

Figure 14: Overview of a TBDelta structure

Example notes:

- Because of the simplicity of their values, a detailed explanation for the following fields has been omitted: **reserved1**, **fc**, **fOnDisk**, **iTB**, **reserved2**, **fDead**, and **cbTBC**.

data: Structure of type **TBDelta** described in [MS-DOC] section 2.9.311.

dopr: 0x0001 specifies that this is an insert operation, which means that a toolbar control has been inserted onto the toolbar with an ID equal to the value of the **tbidForTBD** field of the **Customization** structure, described in [MS-DOC] section 2.9.50, which contains this structure.

fAtEnd: 0x0001 specifies that the toolbar control associated with this toolbar delta was inserted at the end of the built-in toolbar when the toolbar delta was created.

ibts: 0x000A specifies the zero-based index of the toolbar control associated with this toolbar delta in the built-in toolbar when the toolbar delta was created. In this example, the toolbar control is the eleventh toolbar control.

cidNext: In this example, **cidNext** equals 0xFFFFFFFF because **fAtEnd** equals 1.

cid: Structure of type **Cid** as described in [MS-DOC] section 2.9.34, which specifies the command identifier of the toolbar control associated with the toolbar delta.

3.2 Document Summary Information Examples

This is an example of a Document Summary Information **stream** as specified in Property Set Storage (section 2.3.3). In this example, the Document Summary Information stream is from a document as described in [MS-DOC]. The name of this stream is "\005DocumentSummaryInformation". The contents of this stream from offset 0x0000 to 0x06B7 inclusive are shown in hexadecimal bytes. The far-left column is the byte count; the far-right characters are the interpretation of the bytes in the **ANSI character set**. The following sections describe in detail the meaning of portions of this stream.

```
00000000: FE FF 00 00 05 01 02 00  00 00 00 00 00 00 00 00 ................
00000010: 00 00 00 00 00 00 00 00  02 00 00 00 02 D5 CD D5 ................
00000020: 9C 2E 1B 10 93 97 08 00  2B 2C F9 AE 44 00 00 00 ........+,..D...
00000030: 05 D5 CD D5 9C 2E 1B 10  93 97 08 00 2B 2C F9 AE ............+,..
00000040: 98 02 00 00 54 02 00 00  0F 00 00 00 01 00 00 00 ....T...........
00000050: 80 00 00 00 02 00 00 00  88 00 00 00 0E 00 00 00 ................
00000060: 9C 00 00 00 0F 00 00 00  B0 00 00 00 1B 00 00 00 ................
00000070: C0 00 00 00 05 00 00 00  D0 00 00 00 06 00 00 00 ................
00000080: D8 00 00 00 11 00 00 00  E0 00 00 00 17 00 00 00 ................
00000090: E8 00 00 00 0B 00 00 00  F0 00 00 00 10 00 00 00 ................
000000A0: F8 00 00 00 13 00 00 00  00 01 00 00 16 00 00 00 ................
000000B0: 08 01 00 00 0D 00 00 00  10 01 00 00 0C 00 00 00 ................
000000C0: 1D 02 00 00 02 00 00 00  E4 04 00 00 1E 00 00 00 ................
000000D0: 0C 00 00 00 54 75 74 6F  72 69 61 6C 73 00 00 00 ....Tutorials...
000000E0: 1E 00 00 00 0C 00 00 00  50 61 75 6C 61 20 44 6F ........Paula Do
000000F0: 65 00 00 00 1E 00 00 00  08 00 00 00 42 69 67 43 e...........BigC
```

```
00000100: 6F 72 70 00 1E 00 00 00  08 00 00 00 44 72 61 66 orp.........Draf
00000110: 74 00 00 00 03 00 00 00  1F 00 00 00 03 00 00 00 t...............
00000120: 1D 00 00 00 03 00 00 00  86 03 00 00 03 00 00 00 ................
00000130: 00 00 0C 00 0B 00 00 00  00 00 00 00 0B 00 00 00 ................
00000140: 00 00 00 00 0B 00 00 00  00 00 00 00 0B 00 00 00 ................
00000150: 00 00 00 00 1E 10 00 00  09 00 00 00 1C 00 00 00 ................
00000160: 49 6E 76 6F 69 63 65 20  41 70 70 72 6F 76 61 6C Invoice Approval
00000170: 20 50 72 6F 63 65 64 75  72 65 73 00 13 00 00 00  Procedures.....
00000180: 41 70 70 72 6F 76 61 6C  20 50 72 6F 63 65 64 75 Approval Procedu
00000190: 72 65 00 15 00 00 00 20  20 20 20 4D 61 6E 61 67 re.....    Manag
000001A0: 65 72 20 41 70 70 72 6F  76 61 6C 00 18 00 00 00 er Approval.....
000001B0: 20 20 20 20 53 6B 69 70  2D 6C 65 76 65 6C 20 41     Skip-level A
000001C0: 70 70 72 6F 76 61 6C 00  1D 00 00 00 20 20 20 20 pproval.....
000001D0: 20 20 20 20 4F 72 67 61  6E 69 7A 61 74 69 6F 6E     Organization
000001E0: 61 6C 20 43 68 61 72 74  00 20 00 00 00 20 20 20 al Chart. ...
000001F0: 20 20 20 20 53 6B 69 70  20 2D 6C 65 76 65 6C 20      Skip-level
00000200: 76 65 72 69 66 69 63 61  74 69 6F 6E 00 13 00 00 verification....
00000210: 00 56 65 72 69 66 69 63  61 74 69 6F 6E 20 53 74 .Verification St
00000220: 65 70 73 00 19 00 00 00  20 20 20 20 4D 61 6E 61 eps.....    Mana
00000230: 67 65 72 20 56 65 72 69  66 69 63 61 74 69 6F 6E ger Verification
00000240: 00 1C 00 00 00 20 20 20  20 53 6B 69 70 2D 6C 65 .....    Skip-le
00000250: 76 65 6C 20 56 65 72 69  66 69 63 61 74 69 6F 6E vel Verification
00000260: 00 0C 10 00 00 04 00 00  00 1E 00 00 00 06 00 00 ................
00000270: 00 54 69 74 6C 65 00 03  00 00 00 01 00 00 00 1E .Title..........
00000280: 00 00 00 09 00 00 00 48  65 61 64 69 6E 67 73 00 .......Headings.
00000290: 03 00 00 00 08 00 00 00  20 04 00 00 09 00 00 00 ........ .......
000002A0: 00 00 00 00 50 00 00 00  01 00 00 00 D2 00 00 00 ....P...........
000002B0: 02 00 00 00 DA 00 00 00  03 00 00 00 E6 00 00 00 ................
000002C0: 04 00 00 00 B6 03 00 00  05 00 00 00 CA 03 00 00 ................
000002D0: 06 00 00 00 D2 03 00 00  07 00 00 00 DA 03 00 00 ................
000002E0: 07 00 00 01 FA 03 00 00  06 00 00 00 02 00 00 00 ................
000002F0: 0E 00 00 00 5F 50 49 44  5F 4C 49 4E 4B 42 41 53 ...._PID_LINKBAS
00000300: 45 00 03 00 00 00 0C 00  00 00 5F 50 49 44 5F 48 E........._PID_H
00000310: 4C 49 4E 4B 53 00 04 00  00 00 0B 00 00 00 44 65 LINKS.........De
00000320: 70 61 72 74 6D 65 6E 74  00 05 00 00 00 10 00 00 partment........
00000330: 00 44 6F 63 75 6D 65 6E  74 20 6E 75 6D 62 65 72 .Document number
00000340: 00 06 00 00 00 09 00 00  00 41 70 70 72 6F 76 65 .........Approve
00000350: 64 00 07 00 00 00 10 00  00 00 43 72 69 74 69 63 d.........Critic
00000360: 61 6C 53 65 63 74 69 6F  6E 00 02 00 00 00 E4 04 alSection.......
00000370: 00 00 41 00 00 00 02 00  00 00 00 00 00 00 41 00 ..A...........A.
00000380: 00 00 C8 02 00 00 24 00  00 00 03 00 00 00 69 00 ......$.......i.
00000390: 67 00 03 00 00 00 0C 00  00 00 03 00 00 00 00 00 g...............
000003A0: 00 00 03 00 00 00 05 00  00 00 1F 00 00 00 01 00 ................
000003B0: 00 00 00 00 79 32 1F 00  00 00 1B 00 00 00 53 00 ....y2........S.
000003C0: 6B 00 69 00 70 00 4C 00  65 00 76 00 65 00 6C 00 k.i.p.L.e.v.e.l.
000003D0: 56 00 65 00 72 00 69 00  66 00 69 00 63 00 61 00 V.e.r.i.f.i.c.a.
000003E0: 74 00 69 00 6F 00 6E 00  53 00 74 00 61 00 72 00 t.i.o.n.S.t.a.r.
000003F0: 74 00 00 00 00 00 00 03  00 00 00 14 00 3C 00 03 00 t...........<...
00000400: 00 00 09 00 00 00 03 00  00 00 00 00 00 00 03 00 ................
00000410: 00 00 05 00 00 00 1F 00  00 00 1F 00 00 00 43 00 ..............C.
00000420: 3A 00 5C 00 4D 00 69 00  63 00 72 00 6F 00 73 00 :.\.M.i.c.r.o.s.
00000430: 6F 00 66 00 74 00 4F 00  66 00 66 00 69 00 63 00 o.f.t.O.f.f.i.c.
00000440: 65 00 4F 00 72 00 67 00  43 00 68 00 61 00 72 00 e.O.r.g.C.h.a.r.
00000450: 74 00 2E 00 76 00 73 00  64 00 00 00 00 00 1F 00 t...v.s.d.......
00000460: 00 00 01 00 00 00 00 00  79 32 03 00 00 00 64 00 ........y2....d.
00000470: 32 00 03 00 00 00 06 00  00 00 03 00 00 00 00 00 2...............
00000480: 00 00 03 00 00 00 05 00  00 00 1F 00 00 00 1C 00 ................
00000490: 00 00 68 00 74 00 74 00  70 00 3A 00 2F 00 2F 00 ..h.t.t.p.:././.
000004A0: 77 00 77 00 77 00 2E 00  77 00 69 00 6E 00 67 00 w.w.w...w.i.n.g.
000004B0: 74 00 69 00 70 00 74 00  6F 00 79 00 73 00 2E 00 t.i.p.t.o.y.s...
000004C0: 63 00 6F 00 6D 00 2F 00  00 00 1F 00 00 00 01 00 c.o.m./.........
000004D0: 00 00 00 00 79 32 03 00  00 00 4A 00 41 00 03 00 ....y2....J.A...
000004E0: 00 00 03 00 00 00 03 00  00 00 00 00 00 00 03 00 ................
000004F0: 00 00 05 00 00 00 1F 00  00 00 01 00 00 00 00 00 ................
00000500: 79 32 1F 00 00 00 14 00  00 00 5F 00 56 00 65 00 y2........ .V.e.
00000510: 72 00 69 00 66 00 69 00  63 00 61 00 74 00 69 00 r.i.f.i.c.a.t.i.
00000520: 6F 00 6E 00 5F 00 53 00  74 00 65 00 70 00 73 00 o.n._.S.t.e.p.s.
00000530: 00 00 03 00 00 00 3E 00  25 00 03 00 00 00 00 00 ......>.%.......
00000540: 00 00 03 00 00 00 00 00  00 00 03 00 00 00 05 00 ................
```

```
00000550: 00 00 1F 00 00 00 31 00  00 00 68 00 74 00 74 00 ......1...h.t.t.
00000560: 70 00 3A 00 2F 00 2F 00  65 00 6E 00 2E 00 61 00 p.:././.e.n...a.
00000570: 64 00 61 00 74 00 75 00  6D 00 2E 00 63 00 6F 00 d.a.t.u.m...c.o.
00000580: 6D 00 2F 00 61 00 63 00  63 00 6F 00 75 00 6E 00 m./.a.c.c.o.u.n.
00000590: 74 00 69 00 6E 00 67 00  2F 00 6D 00 61 00 6E 00 t.i.n.g./.m.a.n.
000005A0: 61 00 67 00 65 00 72 00  6C 00 69 00 73 00 74 00 a.g.e.r.l.i.s.t.
000005B0: 2E 00 68 00 74 00 6D 00  6C 00 00 00 00 00 1F 00 ..h.t.m.l.......
000005C0: 00 00 0F 00 00 00 65 00  6D 00 61 00 69 00 6C 00 ......e.m.a.i.l.
000005D0: 41 00 64 00 64 00 72 00  65 00 73 00 73 00 65 00 A.d.d.r.e.s.s.e.
000005E0: 73 00 00 00 00 00 03 00  00 00 28 00 6B 00 03 00 s.........(.k...
000005F0: 00 00 FF FF FF FF 03 00  00 00 03 04 00 00 03 00 ................
00000600: 00 00 04 00 00 00 1F 00  00 00 01 00 00 00 00 00 ................
00000610: 79 32 1F 00 00 00 19 00  00 00 5F 00 53 00 6B 00 y2........_.S.k.
00000620: 69 00 70 00 2D 00 6C 00  65 00 76 00 65 00 6C 00 i.p.-.l.e.v.e.l.
00000630: 5F 00 56 00 65 00 72 00  69 00 66 00 69 00 63 00 _.V.e.r.i.f.i.c.
00000640: 61 00 74 00 69 00 6F 00  6E 00 00 00 00 00 1E 00 a.t.i.o.n.......
00000650: 00 00 0C 00 00 00 41 63  63 6F 75 6E 74 69 6E 67 ......Accounting
00000660: 00 00 03 00 00 00 05 00  00 00 0B 00 00 00 00 00 ................
00000670: 00 00 1E 00 00 00 18 00  00 00 53 69 67 6E 20 74 ..........Sign t
00000680: 68 65 20 61 70 70 72 6F  76 61 6C 20 66 6F 72 6D he approval form
00000690: 00 00 1E 00 00 00 1C 00  00 00 53 6B 69 70 4C 65 ..........SkipLe
000006A0: 76 65 6C 56 65 72 69 66  69 63 61 74 69 6F 6E 53 velVerificationS
000006B0: 74 61 72 74 00 00 00 00                          tart....
```

3.2.1 Document Summary Information Stream Overview

The following table contains the beginning part of a Document Summary Information **stream**. There are two **FilePointers** (section 2.2.1.5) in this section that point to a **Document Summary Information property set** (section 2.3.3.2.2) and a **User Defined property set** (section 2.3.3.2.3), respectively.

Offset	Size	Structure	Value
00000000	06B8	PropertySetStream **- DocumentSummaryInfoStream**	
00000000	0044	Stream header **- PropertySetStream**	
00000000	0002	WORD **- byteOrder**	0xFFFE
00000002	0002	WORD **- version**	0x0000
00000004	0004	PropertySetSystemIdentifier **- sysId**	
00000004	0001	BYTE **- OSMajorVersion**	0x05
00000005	0001	BYTE **- OSMinorVersion**	0x01
00000006	0002	WORD **- OSType**	0x0002
00000008	0010	GUID **- applicationClsid**	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000018	0004	DWORD **- cSections**	0x00000002
0000001C	0028	Array of identifier and offset **- rgIdOffset**	
0000001C	0014	Identifier and offset **- IdOffsetElement-1**	
0000001C	0010	GUID **- formatId**	02 D5 CD D5 9C 2E 1B 10 93 97 08 00 2B 2C F9 AE
0000002C	0004	FilePointer **- sectionOffset**	0x00000044

Offset	Size	Structure	Value
00000030	0014	Identifier and offset **- IdOffsetElement-2**	
00000030	0010	GUID **- formatId**	05 D5 CD D5 9C 2E 1B 10 93 97 08 00 2B 2C F9 AE
00000040	0004	FilePointer **- sectionOffset**	0x00000298
00000044	0254	**A**: PropertySet **- DocumentSummaryInformation**	
00000298	0420	**B**: PropertySet **- UserDefinedProperties**	

Figure 15: Overview of a DocumentSummaryInfoStream

DocumentSummaryInfoStream: This is an example of the **PropertySetStream** as described in [MS-OLEPS] section 2.21.

PropertySetStream.byteOrder: 0xFFFE is a reserved value.

PropertySetStream.version: 0x0000 is the version number of the property set.

PropertySetStream.sysId.OSMajorVersion: 0x05 indicates that the **major version** of the operating system that created the file is 5.

PropertySetStream.sysId.OSMinorVersion: 0x01 indicates that the **minor version** of the operating system that created the file is 1.

PropertySetStream.sysId.OSType: 0x0002 is a reserved value.

PropertySetStream.applicationClsid: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 is the CLID_NULL. It is a reserved value.

PropertySetStream.cSections: 0x00000002 indicates that this instance of a Document Summary Information stream contains two property sets, the **Document Summary Information property set** (section 2.3.3.2.2) and the **User Defined property set** (section 2.3.3.2.3).

PropertySetStream.rgIdOffset.IdOffsetElement-1: The first element in the **rgIdOffset** array.

PropertySetStream.rgIdOffset.IdOffsetElement-1.formatId: 02 D5 CD D5 9C 2E 1B 10 93 97 08 00 2B 2C F9 AE is the FMTID_DocSummaryInformation. It indicates that the associated **sectionOffset** points to a **Document Summary Information property set** (section 2.3.3.2.2).

PropertySetStream.rgIdOffset.IdOffsetElement-1.sectionOffset: 0x00000044 indicates that the beginning of the **Document Summary Information property set** (section 2.3.3.2.2) is 0x00000044 bytes after the beginning of the Document Summary Information stream. In this example, **DocumentSummaryInfoStream** begins at offset 0x00000000. Therefore, the **Document Summary Information property set** (section 2.3.3.2.2) begins at offset 0x00000044. See Document Summary Information Property Set Overview (section 3.2.2) for an example of this.

PropertySetStream.rgIdOffset.IdOffsetElement-2: The second element in the **rgIdOffset** array.

PropertySetStream.rgIdOffset.IdOffsetElement-2.formatId: 05 D5 CD D5 9C 2E 1B 10 93 97 08 00 2B 2C F9 AE is the FMTID_UserDefinedProperties. It indicates that the associated **sectionOffset** points to a **User Defined property set** (section 2.3.3.2.3).

PropertySetStream.rgIdOffset.IdOffsetElement-2.sectionOffset: 0x00000298 indicates that the beginning of the **User Defined property set** (section 2.3.3.2.3) is 0x00000298 bytes after the beginning of the Document Summary Information stream. In this example, **DocumentSummaryInfoStream** begins at offset 0x00000000. Therefore, the **User Defined**

property set (section 2.3.3.2.3) begins at offset 0x00000298. See User Defined Property Set Overview (section 3.2.3) for an example of this.

3.2.2 Document Summary Information Property Set Overview

The following table contains the beginning of a **Document Summary Information property set** (section 2.3.3.2.2) **stream**.

Offset	Size	Structure	Value
00000044	0254	**A**: PropertySet **- DocumentSummaryInformation**	
00000044	0004	DWORD **- cbSection**	0x00000254
00000048	0004	DWORD **- cProps**	0x0000000F
0000004C	0078	Array of PropertyIdentifierAndOffset **- rgProps**	
0000004C	0008	PropertyIdentifierAndOffset **- PidOffsetElement-1**	
0000004C	0004	DWORD **- propId**	0x00000001
00000050	0004	FilePointer **- propOffset**	0x00000080
00000054	0008	PropertyIdentifierAndOffset **- PidOffsetElement-2**	
00000054	0004	DWORD **- propId**	0x00000002
00000058	0004	FilePointer **- propOffset**	0x00000088
0000005C	0008	PropertyIdentifierAndOffset **- PidOffsetElement-3**	
0000005C	0004	DWORD **- propId**	0x0000000E
00000060	0004	FilePointer **- propOffset**	0x0000009C
00000064	0008	PropertyIdentifierAndOffset **- PidOffsetElement-4**	
00000064	0004	DWORD **- propId**	0x0000000F
00000068	0004	FilePointer **- propOffset**	0x000000B0
0000006C	0008	PropertyIdentifierAndOffset **- PidOffsetElement-5**	
0000006C	0004	DWORD **- propId**	0x0000001B
00000070	0004	FilePointer **- propOffset**	0x000000C0
00000074	0008	PropertyIdentifierAndOffset **- PidOffsetElement-6**	
00000074	0004	DWORD **- propId**	0x00000005
00000078	0004	FilePointer **- propOffset**	0x000000D0
0000007C	0008	PropertyIdentifierAndOffset **- PidOffsetElement-7**	
0000007C	0004	DWORD **- propId**	0x00000006
00000080	0004	FilePointer **- propOffset**	0x000000D8
00000084	0008	PropertyIdentifierAndOffset **- PidOffsetElement-8**	

Offset	Size	Structure	Value
00000084	0004	DWORD **- propId**	0x00000011
00000088	0004	FilePointer **- propOffset**	0x000000E0
0000008C	0008	PropertyIdentifierAndOffset **- PidOffsetElement-9**	
0000008C	0004	DWORD **- propId**	0x00000017
00000090	0004	FilePointer **- propOffset**	0x000000E8
00000094	0008	PropertyIdentifierAndOffset **- PidOffsetElement-10**	
00000094	0004	DWORD **- propId**	0x0000000B
00000098	0004	FilePointer **- propOffset**	0x000000F0
0000009C	0008	PropertyIdentifierAndOffset **- PidOffsetElement-11**	
0000009C	0004	DWORD **- propId**	0x00000010
000000A0	0004	FilePointer **- propOffset**	0x000000F8
000000A4	0008	PropertyIdentifierAndOffset **- PidOffsetElement-12**	
000000A4	0004	DWORD **- propId**	0x00000013
000000A8	0004	FilePointer **- propOffset**	0x00000100
000000AC	0008	PropertyIdentifierAndOffset **- PidOffsetElement-13**	
000000AC	0004	DWORD **- propId**	0x00000016
000000B0	0004	FilePointer **- propOffset**	0x00000108
000000B4	0008	PropertyIdentifierAndOffset **- PidOffsetElement-14**	
000000B4	0004	DWORD **- propId**	0x0000000D
000000B8	0004	FilePointer **- propOffset**	0x00000110
000000BC	0008	PropertyIdentifierAndOffset **- PidOffsetElement-15**	
000000BC	0004	DWORD **- propId**	0x0000000C
000000C0	0004	FilePointer **- propOffset**	0x0000021D
000000C4	01D4	Array of TypedPropertyValue structures **- DocSumProperties**	

Figure 16: Overview of a DocumentSummaryInformation property set

In this table, there is an array of 15 elements, **rgProps**. Each element in **rgProps** is a **PropertyIdentifierAndOffset** structure ([MS-OLEPS] section 2.19). Each of these structures represents a property that is stored in this **Document Summary Information property set** (section 2.3.3.2.2). Each **PropertyIdentifierAndOffset** structure (**PidOffsetElement-1** through **PidOffsetElement-15**) contains two fields, **propId** and **propOffset**. The **propId** field indicates the property of that element according to **PIDDSI** (section 2.3.3.2.2.1). The **propOffset** field indicates the offset to the **TypedPropertyValue** ([MS-OLEPS] section 2.15) structure containing the value of the property.

In this example, some **PidOffsetElements** share the same structure and therefore are omitted for brevity. **PidOffsetElement-2** to **PidOffsetElement-5** are **Lpstr** properties (section 2.3.3.1.4). For the structure of an **Lpstr** property (section 2.3.3.1.4), see the Category Property Example (section

3.2.2.2). **PidOffsetElement-6** to **PidOffsetElement-9** are VT_I4 **TypedPropertyValue** properties. For the structure of a VT_I4 **TypedPropertyValue** property, see the **LineCount Property Example** section 3.2.2.3). **PidOffsetElement-10** to **PidOffsetElement-13** are VT_BOOL **TypedPropertyValue** properties. For the structure of a VT_BOOL **TypedPropertyValue** property, see the **LinksDirty Property Example** (section 3.2.2.4).

DocumentSummaryInformation: A **DocumentSummaryInformation** is a property set as described in [MS-OLEPS] section 2.20.

cbSection: 0x00000254 indicates the count of bytes from the beginning of the DocumentSummaryInformation property set (at offset 0x00000044) to the end of the last property in the property set, **HeadingPairs** (section 3.2.2.6), at offset 0x00000297 inclusive.

cProps: 0x0000000F indicates that this **Document Summary Information property set** (section 2.3.3.2.2) contains 0x0000000F (15) properties.

rgProps: This is the container of an array of **PropertyIdentifierAndOffset** structures. In this example, there are 15 elements, one for each property.

rgProps.PidOffsetElement-1: CodePage (VT_I2 **TypedPropertyValue** property).

rgProps.PidOffsetElement-1.propId: 0x00000001 (**GKPIDDSI_CODEPAGE** (section 2.3.3.2.2.1) indicates that there is a **CodePage** property value at the stream offset indicated by **propOffset**.

rgProps.PidOffsetElement-1.propOffset: 0x00000080 indicates that the beginning of the **CodePage** property data is 0x00000080 bytes after the beginning of the **Document Summary Information property set** (section 2.3.3.2.2). In this example, **DocumentSummaryInformation** begins at offset 0x00000044. Therefore, the **CodePage** property begins at offset 0x000000C4. See the **CodePage Property Example** (section 3.2.2.1).

rgProps.PidOffsetElement-2: Category (**Lpstr** property (section 2.3.3.1.4)).

rgProps.PidOffsetElement-2.propId: 0x00000002 (**GKPIDDSI_CATEGORY** (section 2.3.3.2.2.1)) indicates that there is a **Category** property value at the stream offset indicated by **propOffset**.

rgProps.PidOffsetElement-2.propOffset: 0x00000088 indicates that the beginning of the **Category** property data is 0x00000088 bytes after the beginning of the **Document Summary Information property set** (section 2.3.3.2.2). In this example, **DocumentSummaryInformation** begins at offset 0x00000044. Therefore, the **Category** property begins at offset 0x000000CC. See the Category Property Example (section 3.2.2.2).

rgProps.PidOffsetElement-3: Manager (**Lpstr** property (section 2.3.3.1.4)).

rgProps.PidOffsetElement-4: Company (**Lpstr** property (section 2.3.3.1.4)).

rgProps.PidOffsetElement-5: ContentStatus (**Lpstr** property (section 2.3.3.1.4)).

rgProps.PidOffsetElement-6: LineCount (VT_I4 **TypedPropertyValue** property).

rgProps.PidOffsetElement-6.propId: 0x00000005 (**GKPIDDSI_LINECOUNT** (section 2.3.3.2.2.1) indicates that there is a **LineCount** property value at the stream offset indicated by **propOffset**.

rgProps.PidOffsetElement-6.propOffset: 0x000000D0 indicates that the **LineCount** property data is 0x000000D0 bytes after the beginning of the **Document Summary Information property set** (section 2.3.3.2.2). In this example, **DocumentSummaryInformation** begins at offset 0x00000044. Therefore, the **LineCount** property begins at offset 0x00000114. See the **LineCount Property Example** (section 3.2.2.3).

rgProps.PidOffsetElement-7: ParagraphCount (VT_I4 **TypedPropertyValue** property).

rgProps.PidOffsetElement-8: CountCharsWithSpaces (VT_I4 **TypedPropertyValue** property).

rgProps.PidOffsetElement-9: Version (VT_I4 **TypedPropertyValue** property).

rgProps.PidOffsetElement-10: Scale (VT_BOOL **TypedPropertyValue** property).

rgProps.PidOffsetElement-11: LinksDirty (VT_BOOL **TypedPropertyValue** property).

rgProps.PidOffsetElement-11.propId: 0x00000010 (**GKPIDDSI_LINKSDIRTY** (section 2.3.3.2.2.1)) indicates that there is a **LinksDirty** property value at the stream offset indicated by **propOffset**.

rgProps.PidOffsetElement-11.propOffset: 0x000000F8 indicates that the **LinksDirty** property data is 0x000000F8 bytes after the beginning of the **Document Summary Information property set** (section 2.3.3.2.2). In this example, **DocumentSummaryInformation** begins at offset 0x00000044. Therefore, the **LinksDirty** property begins at offset 0x00000013C. See the **LinksDirty Property Example** (section 3.2.2.4).

rgProps.PidOffsetElement-12: SharedDoc (VT_BOOL **TypedPropertyValue** property).

rgProps.PidOffsetElement-13: HyperlinksChanged (VT_BOOL **TypedPropertyValue** property).

rgProps.PidOffsetElement-14: DocumentParts (**VtVecUnalignedLpstr** property (section 2.3.3.1.10)).

rgProps.PidOffsetElement-14.propId: 0x0000000D (**GKPIDDSI_DOCPARTS** (section 2.3.3.2.2.1)) indicates that there is a **DocumentParts** property value at the stream offset indicated by **propOffset**.

rgProps.PidOffsetElement-14.propOffset: 0x00000110 indicates that the **DocumentParts** property data is 0x00000110 bytes after the beginning of **Document Summary Information property set** (section 2.3.3.2.2). In this example, **DocumentSummaryInformation** begins at offset 0x00000044. Therefore, the **DocumentParts** property begins at offset 0x00000154. See the **DocumentParts Property Example** (section 3.2.2.5).

rgProps.PidOffsetElement-15: HeadingPairs (**VtVecHeadingPair** property (section 2.3.3.1.10)).

rgProps.PidOffsetElement-15.propId: 0x0000000C (**GKPIDDSI_HEADINGPAIR** (section 2.3.3.2.2.1)) indicates that there is a **HeadingPairs** property value at the stream offset indicated by **propOffset**.

rgProps.PidOffsetElement-15.propOffset: 0x0000021D indicates that the **HeadingPairs** property data is 0x0000021D bytes after the beginning of the **Document Summary Information property set** (section 2.3.3.2.2). In this example, **DocumentSummaryInformation** begins at offset 0x00000044. Therefore, the **HeadingPairs** property begins at offset 0x00000261. See the **HeadingPairs Property Example** (section 3.2.2.6).

3.2.2.1 CodePage Property Example

This is an example of a **CodePage** property (VT_I2 **TypedPropertyValue** property).

Offset	Size	Structure	Value
000000C4	0008	TypedPropertyValue **- GKPIDDSI_CODEPAGE**	
000000C4	0002	WORD **- wType**	0x0002
000000C6	0002	WORD **- padding**	0x0000
000000C8	0002	WORD **- value**	0x04E4
000000CA	0002	WORD **- unused**	0x0000

Figure 17: Structure of a CodePage property

GKPIDDSI_CODEPAGE: CodePage.

wType: 0x0002 indicates this property is a VT_I2 **TypedPropertyValue** property (2-byte integer).

value: 0x04E4 (1252) indicates the **code page** used for the property values of the **Lpstr** (section 2.3.3.1.4) and **UnalignedLpstr** (section 2.3.3.1.5) properties in the Document Summary Information property set (section 2.3.3.2.2).

3.2.2.2 Category Property Example

This is an example of a **Category** property (**Lpstr** property (section 2.3.3.1.4)).

Offset	Size	Structure	Value
000000CC	0014	TypedPropertyValue **- GKPIDDSI_CATEGORY**	
000000CC	0002	WORD **- wType**	0x001E
000000CE	0002	WORD **- padding**	0x0000
000000D0	0010	Lpstr **- stringValue**	
000000D0	0004	DWORD **- cch**	0x0000000C
000000D4	000C	Array of CHAR **- value**	Tutorials

Figure 18: Structure of a Category property

GKPIDDSI_CATEGORY: Category.

wType: 0x001E indicates that this property is an **Lpstr** property (section 2.3.3.1.4).

stringValue.cch: 0x0000000C indicates that the count of characters of the **Lpstr** (section 2.3.3.1.4) value is 12. The **value** field contains 10 characters (9 for "Tutorials" plus a terminating NULL character). There are 2 additional bytes after the terminating NULL character which pad the **value** field to a multiple of 4 bytes.

stringValue.value: The value of this field is "Tutorials" followed by a null-terminator and 2 bytes of padding, so the length of this field is 0x0000000C (12). The 2 bytes of padding is added to make the length a multiple of 4 bytes.

3.2.2.3 LineCount Property Example

This is an example of a **LineCount** property (VT_I4 **TypedPropertyValue** property).

Offset	Size	Structure	Value
00000114	0008	TypedPropertyValue **- GKPIDDSI_LINECOUNT**	
00000114	0002	WORD **- wType**	0x0003
00000116	0002	WORD **- padding**	0x0000
00000118	0004	DWORD **- value**	0x0000001F

Figure 19: Structure of a LineCount property

GKPIDDSI_LINECOUNT: LineCount.

wType: 0x0003 indicates that this property is a VT_I4 **TypedPropertyValue** property (4-byte integer).

value: 0x0000001F indicates that the count of lines in the example document is 0x0000001F (31).

3.2.2.4 LinksDirty Property Example

This is an example of a **LinksDirty** property (VT_BOOL **TypedPropertyValue** property).

Offset	Size	Structure	Value
0000013C	0008	TypedPropertyValue **- GKPIDDSI_LINKSDIRTY**	
0000013C	0002	WORD **- wType**	0x000B
0000013E	0002	WORD **- padding**	0x0000
00000140	0004	DWORD **- value**	0x00000000

Figure 20: Structure of a LinksDirty property

GKPIDDSI_LINKSDIRTY: LinksDirty.

wType: 0x000B indicates that this property is a VT_BOOL **TypedPropertyValue** property (false if the **value** field is 0x00000000, or true if it is any other value).

value: 0x00000000 is the value of the **value** field of this VT_BOOL **TypedPropertyValue** property, meaning false. This indicates that the linked properties (**propId** 0x00000006 in the **UserDefined property set** (section 3.2.3) in this example) do not require updating the next time the file is opened.

3.2.2.5 DocumentParts Property Example

This is an example of a **DocumentParts** property (**VtVecUnalignedLpstr** property (section 2.3.3.1.10)).

The **DocumentParts** property ought to be viewed together with the **HeadingPairs Property** (section 3.2.2.6) because the data in the two are closely related. This relationship is illustrated by the following diagram.

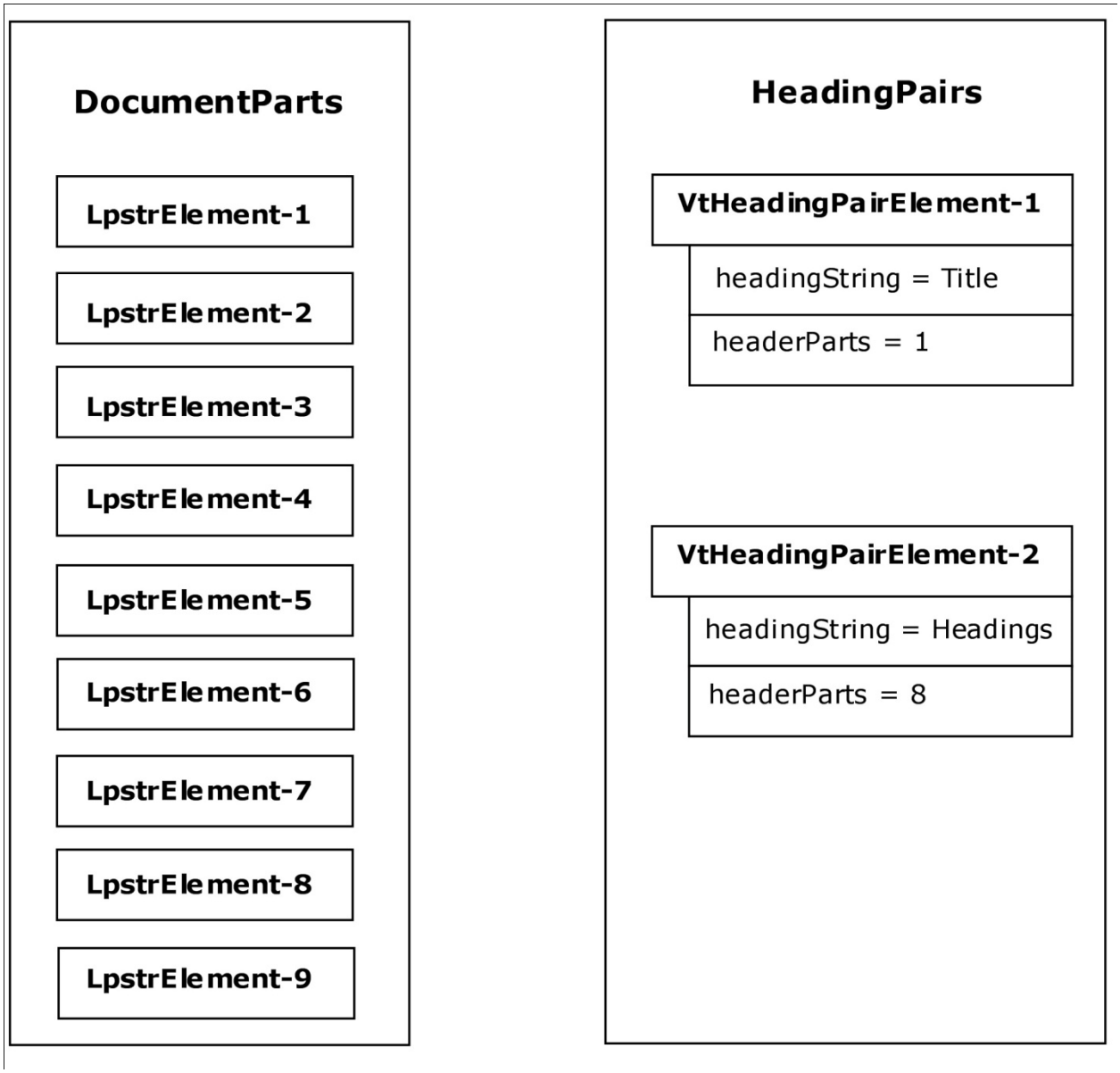


Figure 21: Relationship between the DocumentParts property and the HeadingPairs property

In the figure, the first element of the **HeadingPairs** property has a **headingString** value of "Title", meaning it is associated with the title group in the **DocumentParts** property. The value of the **headerParts** field is 1, meaning there is only 1 element in the **DocumentParts** property that is in the title group, **LpstrElement-1**. The **headingString** of the second element in the **HeadingPairs** property is "Headings", meaning this element is associated with the headings group in the **DocumentParts** property. **HeaderParts** equals 8, meaning there are 8 elements in the **DocumentParts** property that are in the headings group. They are **LpstrElement-2** through **LpstrElement-9**.

Offset	Size	Structure	Value
00000154	010D	TypedPropertyValue **- GKPIDDSI_DOCPARTS**	

Offset	Size	Structure	Value
00000154	0002	WORD **- wType**	0x101E
00000156	0002	WORD **- padding**	0x0000
00000158	0109	VtVecUnalignedLpstrValue **- UnalignedLpstr Vector**	
00000158	0004	DWORD **- cElements**	0x00000009
0000015C	0105	Array of UnalignedLpstr **- rgString**	
0000015C	0020	UnalignedLpstr **- LpstrElement-1**	
0000015C	0004	DWORD **- cch**	0x0000001C
00000160	001C	Array of CHAR **- value**	Invoice Approval Procedures
0000017C	0017	UnalignedLpstr **- LpstrElement-2**	
0000017C	0004	DWORD **- cch**	0x00000013
00000180	0013	Array of CHAR **- value**	Approval Procedure
00000193	0019	UnalignedLpstr **- LpstrElement-3**	
00000193	0004	DWORD **- cch**	0x00000015
00000197	0015	Array of CHAR **- value**	Manager Approval
000001AC	001C	UnalignedLpstr **- LpstrElement-4**	
000001AC	0004	DWORD **- cch**	0x00000018
000001B0	0018	Array of CHAR **- value**	Skip-level Approval
000001C8	0021	UnalignedLpstr **- LpstrElement-5**	
000001C8	0004	DWORD **- cch**	0x0000001D
000001CC	001D	Array of CHAR **- value**	Organizational Chart
000001E9	0024	UnalignedLpstr **- LpstrElement-6**	
000001E9	0004	DWORD **- cch**	0x00000020
000001ED	0020	Array of CHAR **- value**	Skip-level verification
0000020D	0017	UnalignedLpstr **- LpstrElement-7**	
0000020D	0004	DWORD **- cch**	0x00000013
00000211	0013	Array of CHAR **- value**	Verification Steps
00000224	001D	UnalignedLpstr **- LpstrElement-8**	
00000224	0004	DWORD **- cch**	0x00000019
00000228	0019	Array of CHAR **- value**	Manager Verification
00000241	0020	UnalignedLpstr **- LpstrElement-9**	
00000241	0004	DWORD **- cch**	0x0000001C

Offset	Size	Structure	Value
00000245	001C	Array of CHAR **- value**	Skip-level Verification

Figure 22: Structure of a DocumentParts property

In this example, the **LpstrElements** share the same structure. For brevity, only one of them is explained in the following section. **LpstrElement-2** is explained because it is more representative in the calculation of count of bytes.

GKPIDDSI_DOCPARTS: DocumentParts.

wType: 0x101E indicates this property value is an array of single-byte character strings. In the case of **DocumentParts** this indicates the property is a **VtVecUnalignedLpstr** property (section 2.3.3.1.10).

UnalignedLpstr Vector.cElements: 0x00000009 indicates there are 9 elements in the **VtVecUnalignedLpstr** (section 2.3.3.1.10).

UnalignedLpstr Vector.rgString.LpstrElement-2: The second **UnalignedLpstr** (section 2.3.3.1.5) in this vector.

UnalignedLpstr Vector.rgString.LpstrElement-2.cch: 0x00000013 (19) is the count of bytes of the **value** field of the **UnalignedLpstr** (section 2.3.3.1.5) ("Approval Procedure" followed by a terminating NULL character). The **UnalignedLpstr** (section 2.3.3.1.5) structure does not require padding. Therefore, in this example, even though 19 is not a multiple of 4 bytes, no padding is added to the value field.

UnalignedLpstr Vector.rgString.LpstrElement-2.value: "Approval Procedure" is the value of the **value** field of this **UnalignedLpstr** (section 2.3.3.1.5).

The rest of the **UnalignedLpstr** (section 2.3.3.1.5) structures in this example have the same format as the Vector.rgString.LpstrElement-2 structure defined previous.

3.2.2.6 HeadingPairs Property Example

This is an example of a **HeadingPairs** property (**VtVecHeadingPair** property (section 2.3.3.1.15)).

Offset	Size	Structure	Value
00000261	0037	TypedPropertyValue **- GKPIDDSI_HEADINGPAIR**	
00000261	0002	WORD **- wType**	0x100C
00000263	0002	WORD **- padding**	0x0000
00000265	0033	VtVecHeadingPairValue **- GKPIDDSI_HEADINGPAIR**	
00000265	0004	DWORD **- cElements**	0x00000004
00000269	002F	Array of VtHeadingPair **- rgHeadingPairs**	
00000269	0016	VtHeadingPair **- VtHeadingPairElement-1**	
00000269	000E	VtUnalignedString **- headingString**	
00000269	0002	WORD **- stringType**	0x001E
0000026B	0002	WORD **- padding**	0x0000
0000026D	000A	UnalignedLpstr **- stringValue**	

Offset	Size	Structure	Value
0000026D	0004	DWORD **- cch**	0x00000006
00000271	0006	Array of CHAR **- value**	Title
00000277	0008	TypedPropertyValue **- headerParts**	
00000277	0002	WORD **- wType**	0x0003
00000279	0002	WORD **- padding**	0x0000
0000027B	0004	DWORD **- value**	0x00000001
0000027F	0019	VtHeadingPair **- VtHeadingPairElement-2**	
0000027F	0011	VtUnalignedString **- headingString**	
0000027F	0002	WORD **- stringType**	0x001E
00000281	0002	WORD **- padding**	0x0000
00000283	000D	UnalignedLpstr **- stringValue**	
00000283	0004	DWORD **- cch**	0x00000009
00000287	0009	Array of CHAR **- value**	Headings
00000290	0008	TypedPropertyValue **- headerParts**	
00000290	0002	WORD **- wType**	0x0003
00000292	0002	WORD **- padding**	0x0000
00000294	0004	DWORD **- value**	0x00000008

Figure 23: Structure of a HeadingPairs property

The **HeadingPairs** property needs to be viewed together with the **DocumentParts property** (section 3.2.2.5), because data in the two are closely related. See the **DocumentParts property** (section 3.2.2.5) example for more explanation.

wType: 0x100C indicates this property is a vector of variant types.

GKPIDDSI_HEADINGPAIR.cElements: 0x00000004 indicates there are two elements in **rgHeadingPairs**. **cElements** is always twice the number of elements in **rgHeadingPairs** because each **VtHeadingPair** (section 2.3.3.1.13) element contains two values, **headingString** and **headerParts**.

GKPIDDSI_HEADINGPAIR.rgHeadingPairs: This is a container of the array of **VtHeadingPair** (section 2.3.3.1.13).

GKPIDDSI_HEADINGPAIR.rgHeadingPairs.VtHeadingPairElement-1: The first **VtHeadingPair** (section 2.3.3.1.13) in this array. This maps to **LpstrElement-1** of the table labeled "Structure of a DocumentParts property" in section 3.2.2.5, in the **DocumentParts** property.

GKPIDDSI_HEADINGPAIR.rgHeadingPairs.VtHeadingPairElement-1.headingString: The **headingString** in this **VtHeadingPair** (section 2.3.3.1.13) element is an **UnalignedLpstr** (section 2.3.3.1.5) structure. The **value** field of this structure is "Title" and indicates that the associated group in the **DocumentParts** property contains the title of this document.

GKPIDDSI_HEADINGPAIR.rgHeadingPairs.VtHeadingPairElement-1.headerParts: The **headingParts** of this **headingPair** is a VT_I4 **TypedPropertyValue** property. The value of it is

0x00000001, meaning this **headingPair** is associated with 1 element in the **DocumentParts** property.

GKPIDDSI_HEADINGPAIR.rgHeadingPairs.VtHeadingPairElement-2: This is the second **VtHeadingPair** (section 2.3.3.1.13) element in this vector. It conforms to the same structure as the first element but contains different data. The **value** field of **headingString** is "Headings", indicating that the associated elements in the **DocumentParts** property specify the heading of the document. The **value** field of the **headerParts** has a value of 8, meaning there are eight elements of **UnalignedLpstr** (section 2.3.3.1.5) structure in the **DocumentParts** property associated with **VtHeadingPairElement-2**. They are **LpstrElement-2** through to **LpstrElement-9** of the table labeled "Structure of a DocumentParts property" in section 3.2.2.5, in the **DocumentParts** property.

3.2.3 User Defined Property Set Overview

The following table contains the beginning of a **User Defined property set** (section 2.3.3.2.3). In this table, there is an array, **rgUserProps**. Each element in **rgUserProps** is a **PropertyIdentifierAndOffset** structure ([MS-OLEPS] section 2.19). The structure of this section of the **User Defined property set** (section 2.3.3.2.3) is similar to that described in Document Summary Information Property Set Overview (section 3.2.2). The major difference is that in the **User Defined property set** (section 2.3.3.2.3) most **propId** values do not have a meaning defined by this specification, but instead have a meaning defined by their names as found in the **Dictionary property** (section 3.2.3.1).

Offset	Size	Structure	Value
00000298	0420	**B**: PropertySet **- UserDefinedProperties**	
00000298	0004	DWORD **- cbSection**	0x00000420
0000029C	0004	DWORD **- cProps**	0x00000009
000002A0	0048	Array of PropertyIdentifierAndOffset **- rgUserProps**	
000002A0	0008	PropertyIdentifierAndOffset **- PidOffsetElement-1**	
000002A0	0004	DWORD **- propId**	0x00000000
000002A4	0004	FilePointer **- dictionaryOffset**	0x00000050
000002A8	0008	PropertyIdentifierAndOffset **- PidOffsetElement-2**	
000002A8	0004	DWORD **- propId**	0x00000001
000002AC	0004	FilePointer **- codepageOffset**	0x000000D2
000002B0	0008	PropertyIdentifierAndOffset **- PidOffsetElement-3**	
000002B0	0004	DWORD **- propId**	0x00000002
000002B4	0004	FilePointer **- linkbaseOffset**	0x000000DA
000002B8	0008	PropertyIdentifierAndOffset **- PidOffsetElement-4**	
000002B8	0004	DWORD **- propId**	0x00000003
000002BC	0004	FilePointer **- hyperlinksOffset**	0x000000E6
000002C0	0008	PropertyIdentifierAndOffset **- PidOffsetElement-5**	
000002C0	0004	DWORD **- propId**	0x00000004

Offset	Size	Structure	Value
000002C4	0004	FilePointer **- propOffset**	0x000003B6
000002C8	0008	PropertyIdentifierAndOffset **- PidOffsetElement-6**	
000002C8	0004	DWORD **- propId**	0x00000005
000002CC	0004	FilePointer **- propOffset**	0x000003CA
000002D0	0008	PropertyIdentifierAndOffset **- PidOffsetElement-7**	
000002D0	0004	DWORD **- propId**	0x00000006
000002D4	0004	FilePointer **- propOffset**	0x000003D2
000002D8	0008	PropertyIdentifierAndOffset **- PidOffsetElement-8**	
000002D8	0004	DWORD **- propId**	0x00000007
000002DC	0004	FilePointer **- propOffset**	0x000003DA
000002E0	0008	PropertyIdentifierAndOffset **- PidOffsetElement-9**	
000002E0	0004	DWORD **- propId**	0x01000007
000002E4	0004	FilePointer **- propLinkOffset**	0x000003FA
000002E8	03D0	Array of properties **- UserDefinedProperties**	

Figure 24: Overview of a User Defined property set

UserDefinedProperties: **UserDefinedProperties** is a **PropertySet** as detailed in [MS-OLEPS] section 2.20.

cbSection: 0x00000420 indicates the count of bytes from the beginning of the **UserDefinedProperties** property set (0x00000298) to the end of the last property in the property set, the **CriticalSection_Link** property (section 3.2.3.4). If the size is calculated as 0x0000069A + 0x001C - 0x00000298, it actually results in a size of 0x0000041E instead of 0x00000420. This is because the property set is padded to a multiple of 4 bytes. After the end of the **CriticalSection_Link** property (section 3.2.3.4) property, there are 2 bytes of padding to make the count of bytes of the property set a multiple of 4 bytes, resulting in a length of 0x00000420.

cProps: 0x00000009 indicates that this **User Defined property set** (section 2.3.3.2.3) contains nine properties.

rgUserProps: This is the container of an array of **PropertyIdentifierAndOffset** structures. In this example, there are nine elements, one for each property.

rgUserProps.PidOffsetElement-1: Dictionary (**Dictionary** property).

rgUserProps.PidOffsetElement-1.propId: 0x00000000 indicates that there is a **Dictionary** property value at the **stream** offset indicated by the **dictionaryOffset** value.

rgUserProps.PidOffsetElement-1.dictionaryOffset: 0x00000050 indicates that the beginning of the **Dictionary** property data is 0x00000050 bytes after the beginning of the **UserDefinedProperties** property set (0x00000298). In this example, the **Dictionary** property begins at offset 0x000002E8. See the **Dictionary property** example (section 3.2.3.1).

rgUserProps.PidOffsetElement-2: CodePage (VT_I2 **TypedPropertyValue** property).

rgUserProps.PidOffsetElement-2.propId: 0x00000001 indicates that there is a **CodePage** property value at the stream offset indicated by the **codepageOffset** value.

rgUserProps.PidOffsetElement-2.codepageOffset: 0x000000D2 indicates that the beginning of the **CodePage** property data is 0x000000D2 bytes after the beginning of the **UserDefinedProperties** property set.

rgUserProps.PidOffsetElement-3: User-defined property.

rgUserProps.PidOffsetElement-3.propId: 0x00000002 indicates there is a user-defined property value at the stream offset indicated by the **linkbaseOffset** value. The name of this user-defined property is specified in the **Dictionary** property as "_PID_LINKBASE". See the Dictionary Property Example (section 3.2.3.1).

rgUserProps.PidOffsetElement-3.linkbaseOffset: 0x000000DA indicates the user-defined property with **propId** 0x00000002 begins 0x000000DA bytes after the beginning of the **UserDefinedProperties** property set. In this example, this property begins at offset 0x00000372. See the LinkBase Property Example (section 3.2.3.2).

rgUserProps.PidOffsetElement-4: User-defined property.

rgUserProps.PidOffsetElement-4.propId: 0x00000003 indicates there is a user-defined property value at the stream offset indicated by the **hyperlinksOffset** value. The name of this user-defined property is specified in the **Dictionary** property as "_PID_HLINKS". See the Dictionary Property Example (section 3.2.3.1).

rgUserProps.PidOffsetElement-4.hyperlinksOffset: 0x000000E6 indicates the user-defined property with **propId** 0x00000003 begins 0x000000E6 bytes after the beginning of the **UserDefinedProperties** property set. In this example, this property begins at offset 0x0000037E. See the Hyperlinks Property Example (section 3.2.3.3).

rgUserProps.PidOffsetElement-5: User-defined property.

rgUserProps.PidOffsetElement-5.propId: 0x00000004 indicates there is a user-defined property value at the stream offset indicated by the **propOffset** value. The name of this user-defined property is specified in the **Dictionary** property as "Department". See the Dictionary Property Example (section 3.2.3.1).

rgUserProps.PidOffsetElement-5.propOffset: 0x000003B6 indicates the user-defined property with **propId** 0x00000004 begins 0x000003B6 bytes after the beginning of the **UserDefinedProperties** property set. This property is a **Lpstr** (section 2.3.3.1.4) property, see Category Property Example (section 3.2.2.2) for an example of a **Lpstr** (section 2.3.3.1.4) property.

rgUserProps.PidOffsetElement-6: User-defined property.

rgUserProps.PidOffsetElement-6.propId: 0x00000005 indicates there is a user-defined property value at the stream offset indicated by the **propOffset** value. The name of this user-defined property is specified in the **Dictionary** property as "Document number". See the Dictionary Property Example (section 3.2.3.1).

rgUserProps.PidOffsetElement-6.propOffset: 0x000003CA indicates the user-defined property with **propId** 0x00000005 begins 0x000003CA bytes after the beginning of the **UserDefinedProperties** property set. This property is a VT_I4 **TypedPropertyValue** property. See the LineCount Property Example (section 3.2.2.3) for an example of a VT_I4 **TypedPropertyValue** property.

rgUserProps.PidOffsetElement-7: User-defined property.

rgUserProps.PidOffsetElement-7.propId: 0x00000006 indicates there is a user-defined property value at the stream offset indicated by the **propOffset** value. The name of this user-defined property is specified in the **Dictionary** property as "Approved". See the Dictionary Property Example (section 3.2.3.1).

rgUserProps.PidOffsetElement-7.propOffset: 0x000003D2 indicates the user-defined property with **propId** 0x00000006 begins 0x000003D2 bytes after the beginning of the **UserDefinedProperties** property set. This property is a VT_BOOL **TypedPropertyValue** property. See the LinksDirty Property Example (section 3.2.2.4) for an example of a VT_BOOL **TypedPropertyValue** property.

rgUserProps.PidOffsetElement-8: User-defined linked property.

rgUserProps.PidOffsetElement-8.propId: 0x00000007 indicates there is a user-defined property value at the stream offset indicated by the **propOffset** value. The name of this user-defined property is specified in the **Dictionary** property as "CriticalSection". See the **Dictionary property** example (section 3.2.3.1).

rgUserProps.PidOffsetElement-8.propOffset: 0x000003DA indicates the user-defined property with **propId** 0x00000007 begins 0x000003DA bytes after the beginning of the **UserDefinedProperties** property set. In this example, this property begins at offset 0x00000672. See the Linked Property Example (section 3.2.3.4).

rgUserProps.PidOffsetElement-9: User-defined Link.

rgUserProps.PidOffsetElement-9.propId: 0x01000007 indicates that the user-defined property in this property set with **propId** 0x01000007 is linked to document content. This user-defined Link value is at the stream offset indicated by the **propLinkOffset** value. See the Linked Property Example (section 3.2.3.4).

rgUserProps.PidOffsetElement-9.propLinkOffset: 0x000003FA indicates that the user-defined property link with **propId** 0x10000007 begins 0x000003FA bytes after the beginning of the **UserDefinedProperties** property set. In this example, this property begins at offset 0x00000692. See the Linked Property Example (section 3.2.3.4).

3.2.3.1 Dictionary Property Example

This is an example of a **Dictionary** property as detailed in [MS-OLEPS] section 2.17. This property gives a mapping between **propertyId** and **propertyName** for the **User Defined property set** (section 2.3.3.2.3), except for the **Dictionary** property, the **CodePage** property, and the **Links** (section 3.2.3.4). In this example, there are 9 elements in the array of **PropertyIdentifierAndOffset** (**rgUserProps** of the previous table). Each element represents a property. However there are only 6 DictionaryEntry elements ([MS-OLEPS] section 2.16) in the **rgEntries** in the **Dictionary** property of following table. The 3 elements that are excluded are the element associated with the **Dictionary** property (**PidOffsetElement-1** in **rgUserProps**), the element associated with the **CodePage** property (**PidOffsetElement-2** in **rgUserProps**), and the element associated with the property link (**PidOffsetElement-9** in **rgUserProps**).

Offset	Size	Structure	Value
000002E8	0082	Dictionary **- Dictionary**	
000002E8	0004	DWORD **- cEntries**	0x00000006
000002EC	007E	Array of DictionaryEntry **- rgEntries**	
000002EC	0016	DictionaryEntry **- LpstrNameIdElement-1**	
000002EC	0004	DWORD **- propertyId**	0x00000002

Offset	Size	Structure	Value
000002F0	0012	UnalignedLpstr **- propertyName**	
000002F0	0004	DWORD **- cch**	0x0000000E
000002F4	000E	Array of CHAR **- value**	_PID_LINKBASE
00000302	0014	DictionaryEntry **- LpstrNameIdElement-2**	
00000302	0004	DWORD **- propertyId**	0x00000003
00000306	0010	UnalignedLpstr **- propertyName**	
00000306	0004	DWORD **- cch**	0x0000000C
0000030A	000C	Array of CHAR **- value**	_PID_HLINKS
00000316	0013	DictionaryEntry **- LpstrNameIdElement-3**	
00000316	0004	DWORD **- propertyId**	0x00000004
0000031A	000F	UnalignedLpstr **- propertyName**	
0000031A	0004	DWORD **- cch**	0x0000000B
0000031E	000B	Array of CHAR **- value**	Department
00000329	0018	DictionaryEntry **- LpstrNameIdElement-4**	
00000329	0004	DWORD **- propertyId**	0x00000005
0000032D	0014	UnalignedLpstr **- propertyName**	
0000032D	0004	DWORD **- cch**	0x00000010
00000331	0010	Array of CHAR **- value**	Document number
00000341	0011	DictionaryEntry **- LpstrNameIdElement-5**	
00000341	0004	DWORD **- propertyId**	0x00000006
00000345	000D	UnalignedLpstr **- propertyName**	
00000345	0004	DWORD **- cch**	0x00000009
00000349	0009	Array of CHAR **- value**	Approved
00000352	0018	DictionaryEntry **- LpstrNameIdElement-6**	
00000352	0004	DWORD **- propertyId**	0x00000007
00000356	0014	UnalignedLpstr **- propertyName**	
00000356	0004	DWORD **- cch**	0x00000010
0000035A	0010	Array of CHAR **- value**	CriticalSection

Figure 25: Structure of a Dictionary property

cEntries: 0x00000006 indicates there are six entries in **rgEntrie**s.

rgEntries: This is an array of **DictionaryEntry**. The **CodePage** property value in **UserDefinedProperties** is not CP_WINUNICODE (0x04B0) so each **DictionaryEntry** contains an **UnalignedLpstr** (section 2.3.3.1.5), see [MS-OLEPS] section 2.16.

rgEntries.LpstrNameIdElement-1: This is the first element in this array.

rgEntries.LpstrNameIdElement-1.propertyId: 0x00000002 is the same as one of the **propId** elements in **rgUserProps**, which indicates that this **DictionaryEntry** corresponds to the user-defined property with **propId** 0x00000002.

rgEntries.LpstrNameIdElement-1.propertyName: This is an **UnalignedLpstr** (section 2.3.3.1.5) structure.

rgEntries.LpstrNameIdElement-1.propertyName.cch: 0x0000000E is the count of character value for this **UnalignedLpstr** (section 2.3.3.1.5) structure. The **cch** field in **LpstrNameIdElement-1** to **LpstrNameIdElement-6** is the count of character value for each of the **UnalignedLpstr** structures. See the DocumentParts Property Example (section 3.2.2.5) for the structure of an **UnalignedLpstr**.

rgEntries.LpstrNameIdElement-1.propertyName.value: "_PID_LINKBASE" indicates the property name of the user-defined property with **propId** 0x00000002. See the LinkBase Property Example (section 3.2.3.2).

rgEntries.LpstrNameIdElement-2: This is the second element in this array.

rgEntries.LpstrNameIdElement-2.propertyId: 0x00000003 is the same as one of the **propId** elements in **rgUserProps**, which indicates that this **DictionaryEntry** corresponds to the user-defined property with **propId** 0x00000003.

rgEntries.LpstrNameIdElement-2.propertyName: This is an **UnalignedLpstr** (section 2.3.3.1.5) structure.

rgEntries.LpstrNameIdElement-2.propertyName.value: "_PID_HLINKS" indicates the property name of the user-defined property with **propId** 0x00000003. See the Hyperlinks Property Example (section 3.2.3.3).

rgEntries.LpstrNameIdElement-3: This is the third element in this array.

rgEntries.LpstrNameIdElement-3.propertyId: 0x00000004 is the same as one of the **propId** elements in **rgUserProps**, which indicates that this **DictionaryEntry** corresponds to the user-defined property with **propId** 0x00000004.

rgEntries.LpstrNameIdElement-3.propertyName: This is an **UnalignedLpstr** (section 2.3.3.1.5) structure.

rgEntries.LpstrNameIdElement-3.propertyName.value: "Department" indicates the property name of the user-defined property with **propId** 0x00000004.

rgEntries.LpstrNameIdElement-4: This is the fourth element in this array.

rgEntries.LpstrNameIdElement-4.propertyId: 0x00000005 is the same as one of the **propId** elements in **rgUserProps**, which indicates that this **DictionaryEntry** corresponds to the user-defined property with **propId** 0x00000005.

rgEntries.LpstrNameIdElement-4.propertyName: This is an **UnalignedLpstr** (section 2.3.3.1.5) structure.

rgEntries.LpstrNameIdElement-4.propertyName.value: "Document number" indicates the property name of the user-defined property with **propId** 0x00000005.

rgEntries.LpstrNameIdElement-5: This is the fifth element in this array.

rgEntries.LpstrNameIdElement-5.propertyId: 0x00000006 is the same as one of the **propId** elements in **rgUserProps**, which indicates that this **DictionaryEntry** corresponds to the user-defined property with **propId** 0x00000006.

rgEntries.LpstrNameIdElement-5.propertyName: This is the container for an **UnalignedLpstr** (section 2.3.3.1.5) structure.

rgEntries.LpstrNameIdElement-5.propertyName.value: "Approved" indicates the property name of the user-defined property with **propId** 0x00000006.

rgEntries.LpstrNameIdElement-6: This is the sixth element in this array.

rgEntries.LpstrNameIdElement-6.propertyId: 0x00000007 is the same as one of the **propId** elements in **rgUserProps**, which indicates that this **DictionaryEntry** corresponds to the user-defined property with **propId** 0x00000007.

rgEntries.LpstrNameIdElement-6.propertyName: This is an **UnalignedLpstr** (section 2.3.3.1.5) structure.

rgEntries.LpstrNameIdElement-6.propertyName.value: "CriticalSection" indicates the property name of the user-defined property with **propId** 0x00000007. See the Linked Property Example (section 3.2.3.4).

3.2.3.2 LinkBase Property Example

This is an example of the **LinkBase** property (VT_BLOB **TypedPropertyValue** property).

Offset	Size	Structure	Value
00000372	000A	TypedPropertyValue **- _PID_LINKBASE**	
00000372	0002	WORD **- wType**	0x0041
00000374	0002	WORD **- padding**	0x0000
00000376	0004	DWORD **- cb**	0x00000002
0000037A	0002	Array of WCHAR **- value**	0x0000
0000037C	0002	Array of bytes **- padding**	0x0000

Figure 26: Structure of a LinkBase property

wType: 0x0041 indicates the property is a VT_BLOB **TypedPropertyValue** property.

cb: 0x00000002 indicates the count of bytes of the value of this property. In this example, the **value** field contains an empty string, so 0x00000002 refers to the count of bytes of the **Unicode** terminating NULL character.

value: The **value** field of the property contains only a Unicode terminating NULL character.

padding: 0x0000 is 2 bytes of padding. This padding is added so that the length of this VT_BLOB **TypedPropertyValue** property structure is a multiple of 4 bytes.

3.2.3.3 Hyperlinks Property Example

This is an example of **a VtHyperlinks** property (section 2.3.3.1.21). There is a **rgHyperlink** vector in the property, which contains all the **hyperlinks** within this sample document.

Offset	Size	Structure	Value
0000037E	02D0	VtHyperlinks **- _PID_HLINKS**	
0000037E	0002	WORD **- wType**	0x0041

Offset	Size	Structure	Value
00000380	0002	WORD **- padding**	0x0000
00000382	02CC	VtHyperlinkValue **- vtValue**	
00000382	0004	DWORD **- cbData**	0x000002C8
00000386	02C8	VecVtHyperlink **- vecHyperlink**	
00000386	0004	DWORD **- cElements**	0x00000024
0000038A	02C4	Array of VtHyperlink **- rgHyperlink**	
0000038A	006C	VtHyperlink **- linkElement-1**	
000003F6	0074	VtHyperlink **- linkElement-2**	
0000046A	006C	VtHyperlink **- linkElement-3**	
000004D6	005C	VtHyperlink **- linkElement-4**	
00000532	00B4	VtHyperlink **- linkElement-5**	
000005E6	0068	VtHyperlink **- linkElement-6**	

Figure 27: Structure of a Hyperlinks property

wType: 0x0041 indicates the property is of VT_BLOB **TypedPropertyValue** property.

vtValue.cbData: 0x000002C8 indicates the count of bytes of the **vecHyperlink** field. In this example, it is 0x000005E6 + 0x0068 – 0x00000386 = 0x000002C8.

vtValue.vecHyperlink.cElements: 0x00000024 indicates there are six **VtHyperlink** structures (section 2.3.3.1.18) in this array. The number of elements is 1/6 of the value of **cElements** property as each **VtHyperlink** element (section 2.3.3.1.18) contains six values.

vtValue.vecHyperlink.rgHyperlink: This is a vector of **VtHyperlink** structures (section 2.3.3.1.21).

vtValue.vecHyperlink.rgHyperlink.linkElement-1: See the LinkElement-3.2.3.3.1 Example (section 3.2.3.3.1) for the structure of this **VtHyperlink** (section 2.3.3.1.18). The 6 **linkElements** here share the same structure and are described in the following sections.

3.2.3.3.1 LinkElement-1 Example

This is an example of a **linkElement** which consists of 6 structures: **dwHash**, **dwApp**, **dwOfficeArt**, **dwInfo**, **hlink1**, and **hlink2**. The values of **dwApp**, **dwOfficeArt**, and **dwInfo** define where the **hyperlink** is applied. The hyperlink can be applied to a shape, a picture in the document, an external link to a picture, or other content in the documents. The values of **hlink1** and **hlink2** define where the hyperlink links. The hyperlink can point to a specific item (for example, a bookmark or a heading) within the same document, or an external file, or an external site, or a specific item in the external site or file. **dwHash** is the calculated value of the **Hyperlink Hash** (section 2.4.2) algorithm with the values of **hlink1** and **hlink2** as inputs.

Offset	Size	Structure	Value
0000038A	006C	VtHyperlink **- linkElement-1**	
0000038A	0008	TypedPropertyValue **- dwHash**	
0000038A	0002	WORD **- wType**	0x0003

Offset	Size	Structure	Value
0000038C	0002	WORD **- padding**	0x0000
0000038E	0004	DWORD **- value**	0x00670069
00000392	0008	TypedPropertyValue **- dwApp**	
00000392	0002	WORD **- wType**	0x0003
00000394	0002	WORD **- padding**	0x0000
00000396	0004	DWORD **- value**	0x0000000C
0000039A	0008	TypedPropertyValue **- dwOfficeArt**	
0000039A	0002	WORD **- wType**	0x0003
0000039C	0002	WORD **- padding**	0x0000
0000039E	0004	DWORD **- value**	0x00000000
000003A2	0008	TypedPropertyValue **- dwInfo**	
000003A2	0002	WORD **- wType**	0x0003
000003A4	0002	WORD **- padding**	0x0000
000003A6	0004	DWORD **- value**	0x00000005
000003AA	000C	TypedPropertyValue **- hlink1**	
000003AA	0002	WORD **- wType**	0x001F
000003AC	0002	WORD **- padding**	0x0000
000003AE	0008	Lpwstr **- vtValue**	
000003AE	0004	DWORD **- cch**	0x00000001
000003B2	0002	Array of WCHAR **- value**	
000003B4	0002	Array of bytes **- padding**	
000003B4	0001	BYTE **- paddingByte-1**	0x79
000003B5	0001	BYTE **- paddingByte-2**	0x32
000003B6	0040	TypedPropertyValue **- hlink2**	
000003B6	0002	WORD **- wType**	0x001F
000003B8	0002	WORD **- padding**	0x0000
000003BA	003C	Lpwstr **- vtValue**	
000003BA	0004	DWORD **- cch**	0x0000001B
000003BE	0036	Array of WCHAR **- value**	SkipLevelVerificationStart
000003F4	0002	Array of bytes **- padding**	
000003F4	0001	BYTE **- paddingByte-1**	0x00

Offset	Size	Structure	Value
000003F5	0001	BYTE **- paddingByte-2**	0x00

Figure 28: LinkElement example 1

linkElement-1: This is a hyperlink that is applied to text content in the document and that links to a bookmark in the same document.

dwHash: The **value** field of this structure is described in Hyperlink Hash (section 2.4.2).

dwApp: In this example, the hyperlink is not applied on a shape or a picture or an external link to a picture in this document. Therefore, the value of the **dwApp** structure is an index into a **PlcFld** ([MS-DOC] section 2.8.35) that corresponds to the beginning character of the hyperlink field in the document content. The algorithm to determine the value of **dwApp** for this example is described in [MS-DOC] section 2.4.7.

dwOfficeArt: This hyperlink is not applied to a shape, so the **value** field of this structure is 0x00000000.

dwInfo: The **value** field of this structure is 0x00000005 as the hyperlink is not applied to a shape and is in a document file as specified by [MS-DOC].

hlink1: This is the **hyperlink target**. The **value** field of this structure is an empty string as this hyperlink target is the document itself.

hlink2: This is the **hyperlink location**. In this example, the **value** field of the structure, "SkipLevelVerificationStart", is the bookmark in the document that this hyperlink links to.

hlink2.wType: 0x001F indicates that **vtValue** is an **Lpwstr** (section 2.3.3.1.6).

hlink2.vtValue.cch: 0x0000001B indicates the count of characters of the **value** field of the property. 0x0000001B (27) is the count of characters of "SkipLevelVerificationStart" (26) plus the terminating NULL character (1).

hlink2.vtValue.value: The value of this field is the **Unicode** string "SkipLevelVerificationStart" followed by a terminating NULL character.

hlink2.vtValue.padding: This field adds 2 bytes of padding so that the total length of the **Lpwstr** structure (section 2.3.3.1.6) is a multiple of 4 bytes.

3.2.3.3.2 LinkElement-2 Example

This is an example of a **linkElement** that illustrates a **hyperlink** to a document other than the document that contains the link.

Offset	Size	Structure	Value
000003F6	0074	VtHyperlink **- linkElement-2**	
000003F6	0008	TypedPropertyValue **- dwHash**	
000003F6	0002	WORD **- wType**	0x0003
000003F8	0002	WORD **- padding**	0x0000
000003FA	0004	DWORD **- value**	0x003C0014
000003FE	0008	TypedPropertyValue **- dwApp**	
000003FE	0002	WORD **- wType**	0x0003

Offset	Size	Structure	Value
00000400	0002	WORD **- padding**	0x0000
00000402	0004	DWORD **- value**	0x00000009
00000406	0008	TypedPropertyValue **- dwOfficeArt**	
00000406	0002	WORD **- wType**	0x0003
00000408	0002	WORD **- padding**	0x0000
0000040A	0004	DWORD **- value**	0x00000000
0000040E	0008	TypedPropertyValue **- dwInfo**	
0000040E	0002	WORD **- wType**	0x0003
00000410	0002	WORD **- padding**	0x0000
00000412	0004	DWORD **- value**	0x00000005
00000416	0048	TypedPropertyValue **- hlink1**	
00000416	0002	WORD **- wType**	0x001F
00000418	0002	WORD **- padding**	0x0000
0000041A	0044	Lpwstr **- vtValue**	
0000041A	0004	DWORD **- cch**	0x0000001F
0000041E	003E	Array of WCHAR **- value**	C:\MicrosoftOfficeOrgChart.vsd
0000045C	0002	Array of bytes **- padding**	
0000045C	0001	BYTE **- paddingByte**	0x00
0000045D	0001	BYTE **- paddingByte**	0x00
0000045E	000C	TypedPropertyValue **- hlink2**	
0000045E	0002	WORD **- wType**	0x001F
00000460	0002	WORD **- padding**	0x0000
00000462	0008	Lpwstr **- vtValue**	
00000462	0004	DWORD **- cch**	0x00000001
00000466	0002	Array of WCHAR **- value**	
00000468	0002	Array of bytes **- padding**	
00000468	0001	BYTE **- paddingByte**	0x79
00000469	0001	BYTE **- paddingByte**	0x32

Figure 29: LinkElement example 2

linkElement-2: This is a hyperlink applied to text content in the document and links to an external file.

dwHash: The **value** field of this structure is calculated as specified in the **Hyperlink Hash** (section 2.4.2).

dwApp: In this example, the hyperlink is not applied on a shape or a picture or an external link to a picture in this document. Therefore, the value of the **dwApp** structure is an index into a **PlcFld** ([MS-DOC] section 2.8.35) that corresponds to the beginning character of the hyperlink field in the document content. The algorithm to determine the value of **dwApp** for this example is described in [MS-DOC] section 2.4.7.

dwOfficeArt: This hyperlink is not applied to a shape, so the **value** field of this structure is 0x00000000.

dwInfo: The **value** field of this structure is 0x00000005 as the hyperlink is not applied to a shape and it is in a document specified by [MS-DOC].

hlink1: This is the **hyperlink target**. The **value** field of this structure is "C:\MicrosoftOfficeOrgChart.vsd" indicating the name of the file that the hyperlink links to.

hlink2: This is the **hyperlink location**. The **value** field of the structure is an empty string because the hyperlink is not referring to a specific item within that file.

3.2.3.3.3 LinkElement-3 Example

This is an example of a **linkElement** that illustrates a **hyperlink** to a web resource outside of the document that contains the link.

Offset	Size	Structure	Value
0000046A	006C	VtHyperlink **- linkElement-3**	
0000046A	0008	TypedPropertyValue **- dwHash**	
0000046A	0002	WORD **- wType**	0x0003
0000046C	0002	WORD **- padding**	0x0000
0000046E	0004	DWORD **- value**	0x00320064
00000472	0008	TypedPropertyValue **- dwApp**	
00000472	0002	WORD **- wType**	0x0003
00000474	0002	WORD **- padding**	0x0000
00000476	0004	DWORD **- value**	0x00000006
0000047A	0008	TypedPropertyValue **- dwOfficeArt**	
0000047A	0002	WORD **- wType**	0x0003
0000047C	0002	WORD **- padding**	0x0000
0000047E	0004	DWORD **- value**	0x00000000
00000482	0008	TypedPropertyValue **- dwInfo**	
00000482	0002	WORD **- wType**	0x0003
00000484	0002	WORD **- padding**	0x0000
00000486	0004	DWORD **- value**	0x00000005
0000048A	0040	TypedPropertyValue **- hlink1**	
0000048A	0002	WORD **- wType**	0x001F

Offset	Size	Structure	Value
0000048C	0002	WORD **- padding**	0x0000
0000048E	003C	Lpwstr **- vtValue**	
0000048E	0004	DWORD **- cch**	0x0000001C
00000492	0038	Array of WCHAR **- value**	http://www.wingtiptoys.com/
000004CA	000C	TypedPropertyValue **- hlink2**	
000004CA	0002	WORD **- wType**	0x001F
000004CC	0002	WORD **- padding**	0x0000
000004CE	0008	Lpwstr **- vtValue**	
000004CE	0004	DWORD **- cch**	0x00000001
000004D2	0002	Array of WCHAR **- value**	
000004D4	0002	Array of bytes **- padding**	
000004D4	0001	BYTE **- paddingByte**	0x79
000004D5	0001	BYTE **- paddingByte**	0x32

Figure 30: LinkElement example 3

linkElement-3: This is a hyperlink associated with a hyperlink field applied to a picture in the document and linked to an external site.

dwHash: The **value** field of this structure is calculated as specified in the **Hyperlink Hash** (section 2.4.2).

dwApp: In this example, the hyperlink is not applied on a shape or a picture or an external link to a picture in this document. Therefore, the value of the **dwApp** structure is an index into a **PlcFld** ([MS-DOC] section 2.8.35) that corresponds to the beginning character of the hyperlink field in the document content. The algorithm to determine the value of **dwApp** for this example is described in [MS-DOC] section 2.4.7.

dwOfficeArt: This hyperlink is not applied to a shape, so the **value** field of this structure is 0x00000000.

dwInfo: The **value** field of this structure is 0x00000005 as the hyperlink is not applied to a shape and it is in a document specified by [MS-DOC].

hlink1: This is the **hyperlink target**. The **value** field of this structure is "http://www.wingtiptoys.com/" specifying the site that the hyperlink links to.

hlink2: This is the **hyperlink location**. The **value** field of the structure is an empty string because the hyperlink is not referring to a specific item within that hyperlink target.

3.2.3.3.4 LinkElement-4 Example

This is an example of a **linkElement** that illustrates a **hyperlink** to a header inside of the document that contains the link.

Offset	Size	Structure	Value
000004D6	005C	VtHyperlink **- linkElement-4**	

Offset	Size	Structure	Value
000004D6	0008	TypedPropertyValue **- dwHash**	
000004D6	0002	WORD **- wType**	0x0003
000004D8	0002	WORD **- padding**	0x0000
000004DA	0004	DWORD **- value**	0x0041004A
000004DE	0008	TypedPropertyValue **- dwApp**	
000004DE	0002	WORD **- wType**	0x0003
000004E0	0002	WORD **- padding**	0x0000
000004E2	0004	DWORD **- value**	0x00000003
000004E6	0008	TypedPropertyValue **- dwOfficeArt**	
000004E6	0002	WORD **- wType**	0x0003
000004E8	0002	WORD **- padding**	0x0000
000004EA	0004	DWORD **- value**	0x00000000
000004EE	0008	TypedPropertyValue **- dwInfo**	
000004EE	0002	WORD **- wType**	0x0003
000004F0	0002	WORD **- padding**	0x0000
000004F2	0004	DWORD **- value**	0x00000005
000004F6	000C	TypedPropertyValue **- hlink1**	
000004F6	0002	WORD **- wType**	0x001F
000004F8	0002	WORD **- padding**	0x0000
000004FA	0008	Lpwstr **- vtValue**	
000004FA	0004	DWORD **- cch**	0x00000001
000004FE	0002	Array of WCHAR **- value**	
00000500	0002	Array of bytes **- padding**	
00000500	0001	BYTE **- paddingByte**	0x79
00000501	0001	BYTE **- paddingByte**	0x32
00000502	0030	TypedPropertyValue **- hlink2**	
00000502	0002	WORD **- wType**	0x001F
00000504	0002	WORD **- padding**	0x0000
00000506	002C	Lpwstr **- vtValue**	
00000506	0004	DWORD **- cch**	0x00000014
0000050A	0028	Array of WCHAR **- value**	_Verification_Steps

Figure 31: LinkElement example 4

linkElement-4: This is a hyperlink applied to text content in the document and linked to a heading section in the same document.

dwHash: The **value** field of this structure is calculated as specified in the **Hyperlink Hash** (section 2.4.2).

dwApp: In this example, the hyperlink is not applied on a shape or a picture or an external link to a picture in this document. Therefore, the value of the **dwApp** structure is an index into a **PlcFld** ([MS-DOC] section 2.8.35) that corresponds to the beginning character of the hyperlink field in the document content. The algorithm to determine the value of **dwApp** for this example is described in [MS-DOC] section 2.4.7.

dwOfficeArt: This hyperlink is not applied to a shape, so the **value** field of this structure is 0x00000000.

dwInfo: The **value** field of this structure is 0x00000005 as the hyperlink is not applied to a shape and it is in a document specified by [MS-DOC].

hlink1: This is the **hyperlink target**. The **value** field of this structure is an empty string as the hyperlink target is the document itself.

hlink2: This is the **hyperlink location**. The **value** field of the structure, "_Verification_Steps", is the heading in this document that this hyperlink links to.

3.2.3.3.5 LinkElement-5 Example

This is an example of a **linkElement** that illustrates a **hyperlink** to a web resource outside of the document that contains the link, along with an anchor value within that web resource.

Offset	Size	Structure	Value
00000532	00B4	VtHyperlink **- linkElement-5**	
00000532	0008	TypedPropertyValue **- dwHash**	
00000532	0002	WORD **- wType**	0x0003
00000534	0002	WORD **- padding**	0x0000
00000536	0004	DWORD **- value**	0x0025003E
0000053A	0008	TypedPropertyValue **- dwApp**	
0000053A	0002	WORD **- wType**	0x0003
0000053C	0002	WORD **- padding**	0x0000
0000053E	0004	DWORD **- value**	0x00000000
00000542	0008	TypedPropertyValue **- dwOfficeArt**	
00000542	0002	WORD **- wType**	0x0003
00000544	0002	WORD **- padding**	0x0000
00000546	0004	DWORD **- value**	0x00000000
0000054A	0008	TypedPropertyValue **- dwInfo**	
0000054A	0002	WORD **- wType**	0x0003

Offset	Size	Structure	Value
0000054C	0002	WORD **- padding**	0x0000
0000054E	0004	DWORD **- value**	0x00000005
00000552	006C	TypedPropertyValue **- hlink1**	
00000552	0002	WORD **- wType**	0x001F
00000554	0002	WORD **- padding**	0x0000
00000556	0068	Lpwstr **- vtValue**	
00000556	0004	DWORD **- cch**	0x00000031
0000055A	0062	Array of WCHAR **- value**	http://en.adatum.com/accounting/managerlist.html
000005BC	0002	Array of bytes **- padding**	
000005BC	0001	BYTE **- paddingByte**	0x00
000005BD	0001	BYTE **- paddingByte**	0x00
000005BE	0028	TypedPropertyValue **- hlink2**	
000005BE	0002	WORD **- wType**	0x001F
000005C0	0002	WORD **- padding**	0x0000
000005C2	0024	Lpwstr **- vtValue**	
000005C2	0004	DWORD **- cch**	0x0000000F
000005C6	001E	Array of WCHAR **- value**	emailAddresses
000005E4	0002	Array of bytes **- padding**	
000005E4	0001	BYTE **- paddingByte**	0x00
000005E5	0001	BYTE **- paddingByte**	0x00

Figure 32: LinkElement example 5

linkElement-5: This is a hyperlink applied to text content in the document and linked to a bookmark on an external site.

dwHash: The **value** field of this structure is calculated as specified in the **Hyperlink Hash** (section 2.4.2).

dwApp: In this example, the hyperlink is not applied on a shape or a picture or an external link to a picture in this document. Therefore, the value of the **dwApp** structure is an index into a **PlcFld** ([MS-DOC] section 2.8.35) that corresponds to the beginning character of the hyperlink field in the document content. The algorithm to determine the value of **dwApp** for this example is described in [MS-DOC] section 2.4.7.

dwOfficeArt: This hyperlink is not applied to a shape, so the **value** field of this structure is 0x00000000.

dwInfo: The **value** field of this structure is 0x00000005 as the hyperlink is not applied to a shape and it is in a document specified by [MS-DOC].

hlink1: This is the **hyperlink target**. The **value** field of this structure is "http://en.adatum.com/accounting/managerlist.html" indicating the site that the hyperlink links to.

hlink2: This is the **hyperlink location**. The **value** field of the structure, "emailAddresses", is the anchor in the target site that this hyperlink links to.

3.2.3.3.6 LinkElement-6 Example

This is an example of a **linkElement** that illustrates a **hyperlink** located on a shape embedded in the document to a header within the document that contains the link.

Offset	Size	Structure	Value
000005E6	0068	VtHyperlink **- linkElement-6**	
000005E6	0008	TypedPropertyValue **- dwHash**	
000005E6	0002	WORD **- wType**	0x0003
000005E8	0002	WORD **- padding**	0x0000
000005EA	0004	DWORD **- value**	0x006B0028
000005EE	0008	TypedPropertyValue **- dwApp**	
000005EE	0002	WORD **- wType**	0x0003
000005F0	0002	WORD **- padding**	0x0000
000005F2	0004	DWORD **- value**	0xFFFFFFFF
000005F6	0008	TypedPropertyValue **- dwOfficeArt**	
000005F6	0002	WORD **- wType**	0x0003
000005F8	0002	WORD **- padding**	0x0000
000005FA	0004	DWORD **- value**	0x00000402
000005FE	0008	TypedPropertyValue **- dwInfo**	
000005FE	0002	WORD **- wType**	0x0003
00000600	0002	WORD **- padding**	0x0000
00000602	0004	DWORD **- value**	0x00000004
00000606	000C	TypedPropertyValue **- hlink1**	
00000606	0002	WORD **- wType**	0x001F
00000608	0002	WORD **- padding**	0x0000
0000060A	0008	Lpwstr **- vtValue**	
0000060A	0004	DWORD **- cch**	0x00000001
0000060E	0002	Array of WCHAR **- value**	
00000610	0002	Array of bytes **- padding**	
00000610	0001	BYTE **- paddingByte**	0x79

Offset	Size	Structure	Value
00000611	0001	BYTE **- paddingByte**	0x32
00000612	003C	TypedPropertyValue **- hlink2**	
00000612	0002	WORD **- wType**	0x001F
00000614	0002	WORD **- padding**	0x0000
00000616	0038	Lpwstr **- vtValue**	
00000616	0004	DWORD **- cch**	0x00000019
0000061A	0032	Array of WCHAR **- value**	_Skip-level_Verification
0000064C	0002	Array of bytes **- padding**	
0000064C	0001	BYTE **- paddingByte**	0x00
0000064D	0001	BYTE **- paddingByte**	0x00

Figure 33: LinkElement example 6

linkElement-6: This is a hyperlink applied to a shape in the document and linked to a heading section within the same document.

dwHash: The **value** field of this structure is calculated as specified in the **Hyperlink Hash** (section 2.4.2).

dwApp: The **value** field of this structure is 0xFFFFFFFF as the hyperlink is applied to a shape.

dwOfficeArt: The **value** field of this structure is 0x00000402. It indicates the identifier of the shape to which the hyperlink is applied.

dwInfo: The **value** field of this structure is 0x00000004 because the hyperlink is applied to a shape as defined in [MS-ODRAW] section 2.5.1.

hlink1: This is the **hyperlink target**. The **value** field of this structure is an empty string when the hyperlink target is the document itself.

hlink2: This is the **hyperlink location**. The **value** field of the structure, "_Skip-level_Verification", is the heading in this document that this hyperlink links to.

3.2.3.4 Linked Property Example

The following table illustrates the value property of a link/value property pair that defines a **Linked Property** (section 2.3.3.2.3.3).

Offset	Size	Structure	Value
00000672	0020	TypedPropertyValue **- CriticalSection**	
00000672	0002	WORD **- wType**	0x001E
00000674	0002	WORD **- padding**	0x0000
00000676	001C	Lpstr **- stringValue**	
00000676	0004	DWORD **- cch**	0x00000018
0000067A	0018	Array of CHAR **- value**	Sign the approval form

Figure 34: Structure of the CriticalSection property

The **CriticalSection** property is an **Lpstr** property (section 2.3.3.1.4), see Category Property Example (section 3.2.2.2). The **value** field of **CriticalSection** has been populated with the string contained by the bookmark specified in **CriticalSection_Link**.

The following table illustrates the link property of a link/value property pair that defines a **Linked Property** (section 2.3.3.2.3.3).

Offset	Size	Structure	Value
00000692	0024	VtString **- CriticalSection_Link**	
00000692	0002	WORD **- stringType**	0x001E
00000694	0002	WORD **- padding**	0x0000
00000696	0020	Lpstr **- stringValue**	
00000696	0004	DWORD **- cch**	0x0000001C
0000069A	001C	Array of CHAR **- value**	SkipLevelVerificationStart

Figure 35: Structure of the CriticalSection_Link property

The **CriticalSection_Link** property is an **Lpstr** property (section 2.3.3.1.4), see Category Property Example (section 3.2.2.2). The **value** field of **CriticalSection_Link** is a bookmark in the document content of the containing document file. The value contained in this bookmark is stored in the **CriticalSection** property.

The previous two tables have **propId** values of 0x00000007 and 0x01000007 in **rgUserProps**, see UserDefined Property Set Overview (section 3.2.3). Taken together they are an example of a **Linked Property** (section 2.3.3.2.3.3). The table with **propID** 0x01000007 (**CriticalSection_Link**) stores the name of the bookmark. The table with **propID** 0x00000007 (**CriticalSection**) stores the value of the bookmark. When the example file was saved, the value of the bookmark, the name of which was saved in the **CriticalSection_Link** property, was saved in the **CriticalSection** property.

3.3 SmartTag Examples

This section contains an example about how **smart tag** information is embedded in the document.

In the following example, two smart tags of the following types are embedded in the document:

- Stockticker
- Date

The following is the example for the shared data stored in the **PropertyBagStore** structure (section 2.3.4.1).

Offset	Size	Structure	Value
0000142A	00BA	PropertyBagStore **- propBagStore**	
0000142A	0004	ULONG **- cFactoidType**	0x00000002
0000142E	007F	FactoidTypes **- factoidTypes**	
0000142E	0043	**A**: FactoidType **- factoidType-0**	
00001471	003C	**B**: FactoidType **- factoidType-1**	

Offset	Size	Structure	Value
000014AD	0002	USHORT **- cbHdr**	0x000C
000014AF	0002	USHORT **- sVer**	0x0100
000014B1	0004	ULONG **- cfactoid**	0x089A7948
000014B5	0004	ULONG **- cste**	0x00000008
000014B9	002B	**C**: IndexedStringTable **- stringTable**	

Figure 36: PropertyBagStore structure

For simplicity this structure is broken down into substructures that are explained after the following descriptions.

propBagStore: Structure of type **PropertyBagStore** (section 2.3.4.1).

cFactoidType: 0x00000002 specifies the number of distinct smart tag types embedded in the document.

factoidTypes: Array of two **FactoidType** structures (section 2.3.4.2) as specified by the **cFactoidType** field.

cbHdr: 0x000C specifies the count of total bytes, including itself and the **sVer**, **cfactoid**, and **cste** fields.

sVer: 0x0100 specifies the version of this structure.

cfactoid: 0x089A7948 is an arbitrary and ignored value.

cste: 0x00000008 specifies the total number of entries in the **stringTable**.

The substructures identified by label **A**, **B**, and **C** in the preceding example are shown in the table labeled "FactoidType structure for Stockticker smart tag type", the table labeled "FactoidType structure for Date smart tag type", and the table labeled "IndexedStringTable structure", respectively.

Offset	Size	Structure	Value
0000142E	0043	**A**: FactoidType **- factoidType-0**	
0000142E	0004	ULONG **- cbFactoid**	0x0000003F
00001432	0004	ULONG **- id**	0x00000001
00001436	002C	PBString **- rgbUri**	
00001436	15 bits	USHORT **- cch**	0x002A
00001436	1 bit	USHORT **- fAnsiString**	0x1
00001438	002A	array of bytes **- rgxch**	urn:schemas-microsoft-com:office:smarttags
00001462	000D	PBString **- rgbTag**	
00001462	15 bits	USHORT **- cch**	0x000B
00001462	1 bit	USHORT **- fAnsiString**	0x1
00001464	000B	array of bytes **- rgxch**	stockticker
0000146F	0002	PBString **- rgbDownLoadURL**	

Offset	Size	Structure	Value
0000146F	15 bits	USHORT **- cch**	0x0000
0000146F	1 bit	USHORT **- fAnsiString**	0x1
00001471	0000	array of bytes **- rgxch**	

Figure 37: FactoidType structure for Stockticker smart tag type

This is the first **FactoidType** structure (section 2.3.4.2) corresponding to the Stockticker smart tag type. For simplicity, detailed descriptions for the subfields **cch** and **fAnsiString** of **rgbTag** and **rgbDownloadURL** have been omitted.

cbFactoid: 0x0000003F specifies the size of this structure excluding the **cbFactoid** field.

id: 0x00000001 specifies the unique id (across this document) for this smart tag type.

rgbUri.cch: 0x002A specifies the number of characters in the string pointed to by **rbgUri.rgxch**.

rgbUri.fAnsiString: 0x1 specifies that the string pointed to by **rgbUri.rgxch** is an **ANSI character set** string.

rgbUri.rgxch: "urn:schemas-microsoft-com:office:smarttags" specifies the **XML namespace URI** for this smart tag type.

rgbTag.rgxch: "stockticker" specifies the tag name for this smart tag type.

rgbDownLoadURL.rgxch: This field is empty because there is no download **URL** specified for this particular smart tag type.

Offset	Size	Structure	Value
00001471	003C	**B**: FactoidType **- factoidType-1**	
00001471	0004	ULONG **- cbFactoid**	0x00000038
00001475	0004	ULONG **- id**	0x00000002
00001479	002C	PBString **- rgbUri**	
00001479	15 bits	USHORT **- cch**	0x002A
00001479	1 bit	USHORT **- fAnsiString**	0x1
0000147B	002A	array of bytes **- rgxch**	urn:schemas-microsoft-com:office:smarttags
000014A5	0006	PBString **- rgbTag**	
000014A5	15 bits	USHORT **- cch**	0x0004
000014A5	1 bit	USHORT **- fAnsiString**	0x1
000014A7	0004	array of bytes **- rgxch**	date
000014AB	0002	PBString **- rgbDownLoadURL**	
000014AB	15 bits	USHORT **- cch**	0x0000
000014AB	1 bit	USHORT **- fAnsiString**	0x1
000014AD	0000	array of bytes **- rgxch**	

Figure 38: FactoidType structure for Date smart tag type

This is the second **FactoidType** structure (section 2.3.4.2) corresponding to the Date smart tag type. For simplicity, detailed descriptions for all the fields except **id** have been omitted.

id: 0x00000002 specifies the unique id (across this document) for the Date smart tag.

Offset	Size	Structure	Value
000014B9	002B	**C**: IndexedStringTable **- stringTable**	
000014B9	0004	PBString **- stringTableEntry-0**	
000014B9	15 bits	USHORT **- cch**	0x0002
000014B9	1 bit	USHORT **- fAnsiString**	0x1
000014BB	0002	array of bytes **- rgxch**	10
000014BD	0006	PBString **- stringTableEntry-1**	
000014BD	15 bits	USHORT **- cch**	0x0004
000014BD	1 bit	USHORT **- fAnsiString**	0x1
000014BF	0004	array of bytes **- rgxch**	2003
000014C3	0004	PBString **- stringTableEntry-2**	
000014C3	15 bits	USHORT **- cch**	0x0002
000014C3	1 bit	USHORT **- fAnsiString**	0x1
000014C5	0002	array of bytes **- rgxch**	21
000014C7	0005	PBString **- stringTableEntry-3**	
000014C7	15 bits	USHORT **- cch**	0x0003
000014C7	1 bit	USHORT **- fAnsiString**	0x1
000014C9	0003	array of bytes **- rgxch**	Day
000014CC	0004	PBString **- stringTableEntry-4**	
000014CC	15 bits	USHORT **- cch**	0x0002
000014CC	1 bit	USHORT **- fAnsiString**	0x1
000014CE	0002	array of bytes **- rgxch**	ls
000014D0	0007	PBString **- stringTableEntry-5**	
000014D0	15 bits	USHORT **- cch**	0x0005
000014D0	1 bit	USHORT **- fAnsiString**	0x1
000014D2	0005	array of bytes **- rgxch**	Month
000014D7	0007	PBString **- stringTableEntry-6**	
000014D7	15 bits	USHORT **- cch**	0x0005
000014D7	1 bit	USHORT **- fAnsiString**	0x1
000014D9	0005	array of bytes **- rgxch**	trans

Offset	Size	Structure	Value
000014DE	0006	PBString **- stringTableEntry-7**	
000014DE	15 bits	USHORT **- cch**	0x0004
000014DE	1 bit	USHORT **- fAnsiString**	0x1
000014E0	0004	array of bytes **- rgxch**	Year

Figure 39: IndexedStringTable structure

This is an array of string table entries. For simplicity, only the detailed description for entries at indexes 1 and 7 are shown. These two entries are referred to by the Date smart tag's **PropertyBag** structure (section 2.3.4.3) to form a key/value pair. The descriptions for the **cch** and **fAnsiString** subfields are also omitted.

stringTableEntry-1: String table entry at index 1.

stringTableEntry-1.rgxch: "2003" specifies this particular string entry.

stringTableEntry-7: String table entry at index 7.

stringTableEntry-7.rgxch: "Year" specifies this particular string entry.

The following example shows how data private to the individual smart tag is persisted. In this example, the smart tags for the following data items were embedded in the document:

- 10/21/2003
- MSFT

The first data item was recognized as a date by the Date smart tag, and the second data item was recognized as a stock symbol by the Stockticker smart tag.

Offset	Size	Structure	Value
000014E4	002C	PropertyBags **- propBags**	
000014E4	0026	PropertyBag **- propertyBag-0**	
000014E4	0002	USHORT **- id**	0x0002
000014E6	0002	USHORT **- cProp**	0x0004
000014E8	0002	USHORT **- cbUnknown**	0x0000
000014EA	0020	Properties **- properties**	
000014EA	0008	Property **- property-0**	
000014EA	0004	ULONG **- keyIndex**	0x00000004
000014EE	0004	ULONG **- valueIndex**	0x00000006
000014F2	0008	Property **- property-1**	
000014F2	0004	ULONG **- keyIndex**	0x00000005
000014F6	0004	ULONG **- valueIndex**	0x00000000
000014FA	0008	Property **- property-2**	

Offset	Size	Structure	Value
000014FA	0004	ULONG **- keyIndex**	0x00000003
000014FE	0004	ULONG **- valueIndex**	0x00000002
00001502	0008	Property **- property-3**	
00001502	0004	ULONG **- keyIndex**	0x00000007
00001506	0004	ULONG **- valueIndex**	0x00000001
0000150A	0006	PropertyBag **- propertyBag-1**	
0000150A	0002	USHORT **- id**	0x0001
0000150C	0002	USHORT **- cProp**	0x0000
0000150E	0002	USHORT **- cbUnknown**	0x0000
00001510	0000	Properties **- properties**	

Figure 40: PropertyBag structures

For simplicity detailed descriptions for **propertyBag-1.cProp** and **propertyBag-1.cbUnknown** have been omitted.

propBags: An array of **PropertyBag** (section 2.3.4.3).

propertyBag-0: The first **PropertyBag** (section 2.3.4.3).

propertyBag-0.id: 0x0002 specifies the **FactoidType** (section 2.3.4.2) **id**. This value maps it to the Date smart tag type as shown in the **factoidTypes.factoidType-1** field of **PropertyBag** (section 2.3.4.3).

propertyBag-0.cProp: 0x0004 specifies the number of properties stored as part of this smart tag.

propertyBag-0.cbUnknown: 0x0000 specifies that this field is ignored.

propertyBag-0.properties: An array of key/value pair indexes in the **StringTable** field as shown in the preceding example. For simplicity, only the entry at index 3 is shown.

propertyBag-0.properties.property-3: Key/value pair at index 3. The pair as shown evaluates to "Year"="2003".

propertyBag-0.properties.property-3.keyIndex: 0x00000007 specifies index in the **stringTable** field of **PropertyBag** (section 2.3.4.3). This maps to the string "Year".

propertyBag-0.properties.property-3.valueIndex: 0x00000001 specifies the index in the **stringTable** field of **PropertyBag** (section 2.3.4.3). This maps to the string "2003".

propertyBag-1: The second **PropertyBag** (section 2.3.4.3).

propertyBag-1.id: 0x0001 specifies the **FactoidType** (section 2.3.4.2) **id**. This value maps it to the Stockticker smart tag type as shown in the **factoidTypes.factoidType-0** field of the **PropertyBag** (section 2.3.4.3).

propertyBag-1.properties: This smart tag does not have any properties.

3.4 Visual Basic for Applications Digital Signature Example Structures

This is an example of a **Visual Basic for Applications (VBA) digital signature** storage structure. In this example, the structure of the digital signature is for a document as specified by [MS-XLS]. The digital signature of a document specified by [MS-PPT] has the same structure as described in this example. The digital signature of a document specified by [MS-DOC] is slightly different than this example. The difference is described in Visual Basic for Applications Digital Signature (section 1.3.4).

Offset	Size	Structure	Value
00000120	0754	DigSigBlob **- vbaDigSig**	
00000120	0004	ULONG **- cb**	0x0000074C
00000124	0004	ULONG **- serializedPointer**	0x00000008
00000128	074C	DigSigInfoSerialized **- signatureInfo**	
00000128	0004	DWORD **- cbSignature**	0x000003E4
0000012C	0004	DWORD **- signatureOffset**	0x0000002C
00000130	0004	DWORD **- cbSigningCertStore**	0x00000340
00000134	0004	DWORD **- certStoreOffset**	0x00000410
00000138	0004	DWORD **- cbProjectName**	0x00000000
0000013C	0004	DWORD **- projectNameOffset**	0x00000750
00000140	0004	BOOL **- fTimestamp**	0x00000000
00000144	0004	DWORD **- cbTimestampUrl**	0x00000000
00000148	0004	DWORD **- timestampUrlOffset**	0x00000752
0000014C	03E4	SignedData **- pbSignatureBuffer**	30 82 03 E0 06 09 ...
00000530	0340	VBASigSerializedCertStore **- pbSigningCertStoreBuffer**	
00000530	0004	DWORD **- version**	0x00000000
00000534	0004	DWORD **- fileType**	0x54524543
00000538	032C	CertStoreCertificateGroup **- certGroup**	
00000538	0108	Array of SerializedPropertyEntry **- elementList**	
00000538	0020	SerializedPropertyEntry **- certStorePropElement-1**	
00000538	0004	DWORD **- id**	0x00000014
0000053C	0004	DWORD **- encodingType**	0x00000001
00000540	0004	DWORD **- length**	0x00000014
00000544	0014	Array of bytes **- value**	3C 6B 4E A5 18 10 4F 7F 8F 6F 03 AD 5D FB D1 B5 71 1E 93 21
00000558	00A8	SerializedPropertyEntry **- certStorePropElement-2**	
00000600	0020	SerializedPropertyEntry **- certStorePropElement-3**	

Offset	Size	Structure	Value
00000620	0020	SerializedPropertyEntry **- certStorePropElement-4**	
00000640	0224	SerializedCertificateEntry **- certificateElement**	
00000640	0004	DWORD **- id**	0x00000020
00000644	0004	DWORD **- encodingType**	0x00000001
00000648	0004	DWORD **- length**	0x00000218
0000064C	0218	Array of bytes **- certificate**	30 82 02 14 30 82 ...
00000864	000C	EndElementMarkerEntry **- endMarkerElement**	
00000864	0004	DWORD **- id**	0x00000000
00000868	0008	Array of bytes **- marker**	00 00 00 00 00 00 00 00
00000870	0002	Array of WCHAR **- rgchProjectNameBuffer**	
00000872	0002	Array of WCHAR **- rgchTimestampBuffer**	
00000874	0000	Array of bytes **- padding**	

Figure 41: VBA digital signature

In this example, **certStorePropElement-2** through **certStorePropElement-4** have the same structure as **certStorePropElement-1**. See the certStorePropElement-1 element for details.

cb: 0x0000074C is the count of bytes of the **signatureInfo** and **padding** fields combined.

serializedPointer: 0x00000008 is the offset for the **signatureInfo** structure.

signatureInfo: This is the structure that contains detailed data for the VBA digital signature.

signatureInfo.cbSignature: 0x000003E4 is the count of bytes for the **pbSignatureBuffer** field.

signatureInfo.signatureOffset: 0x0000002C indicates that the **pbSignatureBuffer** field begins 0x0000002C bytes after the beginning of its parent **DigSigBlob** (section 2.3.2.2). In this example, the parent **DigSigBlob** begins at offset 0x00000120. Therefore, the **pbSignatureBuffer** field begins at offset 0x0000014C.

signatureInfo.cbSigningCertStore: 0x00000340 is the count of bytes of the **pbSigningCertStoreBuffer**.

signatureInfo.certStoreOffset: 0x00000410 indicates that the **pbSignatureCertStoreBuffer** field begins 0x00000410 bytes after the beginning of its parent **DigSigBlob** (section 2.3.2.2). In this example, the parent **DigSigBlob** begins at offset 0x00000120. Therefore, the **pbSigningCertStoreBuffer** field begins at offset 0x00000530.

signatureInfo.cbProjectName: 0x00000000 is the count of bytes not including the terminating NULL character of the **rgchProjectNameBuffer** field.

signatureInfo.projectNameOffset: 0x00000750 indicates that the **rgchProjectNameBuffer** field begins 0x00000750 bytes after the beginning of its parent **DigSigBlob** (section 2.3.2.2). In this example, the parent **DigSigBlob** begins at offset 0x00000120. Therefore, the **rgchProjectNameBuffer** field begins at offset 0x00000870.

signatureInfo.fTimestamp: 0x00000000 is a reserved value.

signatureInfo.cbTimestampUrl: 0x00000000 is the count of bytes not including the terminating NULL character of the **rgchTimestampBuffer** field.

signatureInfo.timestampUrlOffset: 0x00000752 indicates that the **rgchTimestampBuffer** field begins 0x00000752 bytes after the beginning of its parent **DigSigBlob** (section 2.3.2.2). In this example, the parent **DigSigBlob** begins at offset 0x00000120. Therefore, the **rgchTimestampBuffer** field begins at offset 0x00000872.

signatureInfo.pbSignatureBuffer: The value of this field is a VBA digital signature as described in VBA Digital Signature (section 2.3.2.4).

signatureInfo.pbSigningCertStoreBuffer: The value of this field is a **VBASigSerializedCertStore** structure (section 2.3.2.5.5) containing the **digital certificate** used to create the digital signature.

signatureInfo.pbSigningCertStoreBuffer.version: 0x00000000 indicates the version of the structure.

signatureInfo.pbSigningCertStoreBuffer.fileType: 0x54524543 indicates that the data contained in **pbSigningCertStoreBuffer** is a digital certificate store type.

signatureInfo.pbSigningCertStoreBuffer.certGroup: This is a container of **digital certificate store** elements representing a single digital certificate and a collection of properties associated with that certificate.

signatureInfo.pbSigningCertStoreBuffer.certGroup.elementList: This is the container of an array of **SerializedPropertyEntry** (section 2.3.2.5.3).

signatureInfo.pbSigningCertStoreBuffer.certGroup.elementList.certStorePropElement-1: This is the first element in this **SerializedPropertyEntry** (section 2.3.2.5.3) array.

signatureInfo.pbSigningCertStoreBuffer.certGroup.elementList.certStorePropElement-1.id: 0x00000014 is the identifier of the property.

signatureInfo.pbSigningCertStoreBuffer.certGroup.elementList.certStorePropElement-1.encodingType: 0x00000001 is a reserved value.

signatureInfo.pbSigningCertStoreBuffer.certGroup.elementList.certStorePropElement-1.length: 0x00000014 is the count of bytes of the **value** field in this element.

signatureInfo.pbSigningCertStoreBuffer.certGroup.elementList.certStorePropElement-1.value: The value of this field is ignored on read.

signatureInfo.pbSigningCertStoreBuffer.certGroup.certificateElement: This is a **SerializedCertificateEntry** structure (section 2.3.2.5.1) containing the certificate stored in this digital certificate store.

signatureInfo.pbSigningCertStoreBuffer.certGroup.certificateElement.id: 0x00000020 indicates the beginning of a **certificateElement**.

signatureInfo.pbSigningCertStoreBuffer.certGroup.certificateElement.encodingType: 0x00000001 is a reserved value.

signatureInfo.pbSigningCertStoreBuffer.certGroup.certificateElement.length: 0x00000218 is the count of bytes of the certificate field in this element.

signatureInfo.pbSigningCertStoreBuffer.certGroup.certificateElement.certificate: The value of this field is the certificate data ([RFC3280]).

signatureInfo.pbSigningCertStoreBuffer.endMarkerElement: This is the container of a special entry in the **VBASigSerializedCertStore** structure (section 2.3.2.5.5) that marks the end of the **VBASigSerializedCertStore** structure.

signatureInfo.pbSigningCertStoreBuffer.endMarkerElement.id: 0x00000000 is a reserved value.

signatureInfo.pbSigningCertStoreBuffer.endMarkerElement.marker: 00 00 00 00 00 00 00 00 is the sentinel value for the end of the store.

signatureInfo.rgchProjectNameBuffer: The value of the field is an empty string (a single null **Unicode** character).

signatureInfo.rgchTimestampBuffer: The value of the field is an empty string (a single null Unicode character).

padding: The **padding** field is an array of bytes. The size of this array is the number of bytes necessary to pad the size of the **signatureInfo** field to a multiple of 4 bytes. In this example, the count of bytes of the **signatureInfo** field is 0x0000074C, a multiple of 4 bytes. Therefore, no padding is needed.

4 Security Considerations

4.1 Toolbar Customization

None.

4.2 Property Set Storage

None.

4.3 Visual Basic for Applications Digital Signature

The **VBA** project data **hash** as described in [MS-OVBA] section 2.4.2 continues to require MD5 hash, even if the digital signing of that data uses a different algorithm. This requirement exists to support cross-version compatibility. This means that even if the **digital signature** uses an alternate algorithm, the **digest** field of **DigestInfo** in **SpcIndirectData** (section 2.3.2.4.3.1) contains a 16-byte MD5 hash, even if the **digestAlgorithm** field of **DigestInfo** in **SpcIndirectData** (section 2.3.2.4.3.1) indicates a non-MD5 digest algorithm<74>.

5 Appendix A: 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 Office 97
- Microsoft Office 2000
- Microsoft Office XP
- Microsoft Office 2003
- the 2007 Microsoft Office system
- Microsoft Office 2010 suites
- Microsoft Office 2013
- Microsoft Office 2016
- Microsoft Office 2019

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.

<1> Section 1.5: These persistence formats provide interoperability with applications that create or read documents conforming to this structure, including Microsoft Word 97, Microsoft Word 2000, Microsoft Word 2002, Microsoft Office Word 2003, Microsoft Excel 97, Microsoft Excel 2000, Microsoft Excel 2002, Microsoft Office Excel 2003, Microsoft PowerPoint 97, Microsoft PowerPoint 2000, Microsoft PowerPoint 2002, and Microsoft Office PowerPoint 2003. These persistence formats can also be used for interoperability with Microsoft Office Word 2007, Microsoft Word 2010, Microsoft Word 2013, Microsoft Office Excel 2007, Microsoft Excel2010, Microsoft Excel 2013, Microsoft Office PowerPoint 2007, Microsoft PowerPoint 2010, or Microsoft PowerPoint 2013 when compatibility with Word 97, Word 2000, Word 2002, Office Word 2003, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, and Office PowerPoint 2003 is a primary concern.

<2> Section 2.3.1.6: In Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013, **cCL** specifies the number of **toolbar controls** in the toolbar. In Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 **cCL** does not specify the number of toolbar controls in the **toolbar** and the value of **cCL** is equal to 0xFF (-1).

<3> Section 2.3.1.8: In Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013, **fCtlModified** can be 0 and the toolbar can still save its toolbar controls to the file. In Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013, **fCtlModified** is equal to 1 and ignored.

<4> Section 2.3.1.12: In Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013, **fNoSetCursor** is equal to 0 if the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001. In Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013, **fNoSetCursor** is equal to the appropriate value listed in [MS-CTXLS] section 2.2 if the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001.

<5> Section 2.3.1.12: In Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013, **fNoAccel** is equal to 0 if the **tct** value of the **TBCHeader** structure (section 2.3.1.10) that contains this structure is equal to 0x0F. Otherwise, if **tct** is not equal to 0x0F, **fNoAccel** is equal to 1. In Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013, **fNoAccel** is equal to the appropriate value listed in [MS-CTDOC] section 2.2 if the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001.

<6> Section 2.3.1.12: In Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 **fAlwaysEnabled** is equal to 0 if the **tcid** value of the **TBCHeader** structure (section 2.3.1.10) that contains this structure does not equal 0x0001. In Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013, **fAlwaysEnabled** is equal to the appropriate value listed in [MS-CTXLS] section 2.2 if the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001.

<7> Section 2.3.1.12: In Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013, **fExclusivePopup** is equal to 0. In Excel 97, and Excel 2000, **fExclusivePopup** is equal to 1 when the **tcid** field of the **TBCHeader** structure (section 2.3.1.10) that contains this structure equals 0x0001. In Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013, **fExclusivePopup** is equal to 1 when the **tcid** field of the **TBCHeader** structure that contains this structure equals 0x0001 and when the **tct** field the **TBCHeader** structure (section 2.3.1.10) that contains this structure equals one of the following values: 0x0A, 0x0B, 0x0C, 0x0D, or 0x0E. In Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013, **fExclusivePopup** is equal to 0 when the **tcid** field of the **TBCHeader** structure that contains this structure equals 1 and when the **tct** field the **TBCHeader** structure that contains this structure does not equal any of the following values: 0x0A, 0x0B, 0x0C, 0x0D, and 0x0E. In Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013, **fExclusivePopup** is equal to the appropriate value listed in [MS-CTXLS] section 2.2 if the **tcid** value of the **TBCHeader** structure that contains this structure does not equal 0x0001.

<8> Section 2.3.1.12: In Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013, **fDefaultBehavior** is equal to 0. In Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013, **fDefaultBehavior** is equal to 0 when the **tcid** field of the **TBCHeader** structure (section 2.3.1.10) that contains this structure equals 0x0001 and is equal to 1 when the **tcid** field of the **TBCHeader** structure that contains this structure does not equal 0x0001.

<9> Section 2.3.1.18: In Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013, **reserved1** is equal to 1. In Word 97, and Excel 97, **reserved1** is equal to 0 and ignored.

<10> Section 2.3.2.4.1: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 use the MD5 algorithm to create the signature digest, specified by the identifier "1.2.840.113549.2.5".

<11> Section 2.3.2.4.4.2: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 do not provide a value for the **programName** string when signing the **VBA** project. However, if one is provided the applications interpret it.

<12> Section 2.3.2.4.4.2: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the **moreInfo** field when signing the VBA project. However, if the field is provided, the applications interpret it.

<13> Section 2.3.2.5.3: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 write properties in the **digital certificate store** as a byproduct of the way the digital certificate store is constructed, but none of the properties specify any behavior and are ignored when encountered.

<14> Section 2.3.3: A minimal document is not required to contain this property set storage, but all files written through a standard save operation by Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 include this property set storage.

<15> Section 2.3.3.1.4: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 do not constrain this value on write, but enforce this constraint on read.

<16> Section 2.3.3.1.4: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 write out the count of bytes including padding and the null-terminating character.

<17> Section 2.3.3.1.4: The [MS-OLEPS] specification requires that padding is done with 0x00 values, but Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 write out arbitrary values for their padding in this case.

<18> Section 2.3.3.1.5: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 do not constrain this value on write, but enforce this constraint on read.

<19> Section 2.3.3.1.6: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 do not constrain this value on write, but enforce this constraint on read.

<20> Section 2.3.3.1.6: The [MS-OLEPS] specification requires that padding is done with 0x00 values, but Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 write out arbitrary values for their padding in this case.

<21> Section 2.3.3.1.18: For PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013 document content, other than **hyperlinks** on shapes, the value is 0x00000007.

<22> Section 2.3.3.1.18: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 generate this value as specified in [MS-DOC] section 2.4.7. Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 generate this value as specified in [MS-XLS] section 2.7.1. For PowerPoint 97, PowerPoint 2000, PowerPoint

2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013, the value is 0x00000006.

<23> Section 2.3.3.1.18: For hyperlinks on shapes, this value is generated as specified in [MS-ODRAW] section 2.5.1. For hyperlinks in other Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 document content, this value is 0x00000005. For hyperlinks in other Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 document content, this value is 0x00000006. For hyperlinks in other PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013 document content, this value is 0x0000007.

<24> Section 2.3.3.2.1.1: The **GKPIDSI_PAGECOUNT** property is written and interpreted by Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 only. PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, or preserve an existing value but ignore it.

<25> Section 2.3.3.2.1.1: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 can write an estimate for this value. PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013 will calcuPIDSIlate an accurate value for this property on each save.

<26> Section 2.3.3.2.1.1: The **GKPIDSI_WORDCOUNT** property is written and interpreted by Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013. Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, or preserve an existing value but ignore it.

<27> Section 2.3.3.2.1.1: The **GKPIDSI_CHARCOUNT** property is written and interpreted by Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 only. PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, or preserve an existing value but ignore it.

<28> Section 2.3.3.2.2.1: The **GKPIDDSI_PRESFORMAT** property is written and interpreted by PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013 only. Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, write an empty string for the value, or preserve a read value from an existing file but ignore it.

<29> Section 2.3.3.2.2.1: Only written by PowerPoint 97, PowerPoint 2000, PowerPoint 2002, and Office PowerPoint 2003.

<30> Section 2.3.3.2.2.1: Only written by Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013.

<31> Section 2.3.3.2.2.1: Only written by Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013.

<32> Section 2.3.3.2.2.1: Only written by Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013.

<33> Section 2.3.3.2.2.1: The **GKPIDDSI_BYTECOUNT** property is written by Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013 only. Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, or preserve an existing value but ignore it. Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 usually omit this property or in some scenarios write a value that is inaccurate for this property, because Word 97,

Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 only set or update this property when the **BuiltInDocumentProperties** property is called from the application's document object model and do not recalculate it on save.

<34> Section 2.3.3.2.2.1: The **GKPIDDSI_LINECOUNT** property is written and interpreted by Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 only. PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, or preserve an existing value but ignore it.

<35> Section 2.3.3.2.2.1: In some cases Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 write an estimate for this value. PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013 will calculate an accurate value for this property on each save.

<36> Section 2.3.3.2.2.1: The **GKPIDDSI_PARACOUNT** property is written and interpreted by Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013. Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, or preserve an existing value but ignore it.

<37> Section 2.3.3.2.2.1: The **GKPIDDSI_SLIDECOUNT** property is written and interpreted by PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013 only. Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, write 0, or preserve an existing value but ignore it.

<38> Section 2.3.3.2.2.1: The **GKPIDDSI_NOTECOUNT** property is written and interpreted by PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013 only. Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, write 0, or preserve an existing value but ignore it.

<39> Section 2.3.3.2.2.1: The **GKPIDDSI_HIDDENCOUNT** property is written and interpreted by PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013 only. Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, write 0, or preserve an existing value but ignore it.

<40> Section 2.3.3.2.2.1: The **GKPIDDSI_MMCLIPCOUNT** property is written and interpreted by PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013 only. Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, write 0, or preserve an existing value but ignore it.

<41> Section 2.3.3.2.2.1: The **GKPIDDSI_LINKSDIRTY** property is interpreted by Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 only. PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 write 0x00000000 for the property value. Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 will preserve an existing value but ignore it (though in the value preservation case, a nonzero value will get converted to the value 0xFFFFFFF on save). PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 fix up linked fields regardless of the state

of this property. Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 write the **GKPIDDSI_LINKSDIRTY** property as FALSE (0x00000000) because they write out updated linked property values, if applicable, in the **User Defined Property Set** (section 2.3.3.2.3) when saving a document.

<42> Section 2.3.3.2.2.1: The **GKPIDDSI_CCHWITHSPACES** property is written and interpreted by Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 only. PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 omit the property, or will preserve an existing value but ignore it.

<43> Section 2.3.3.2.2.1: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 write the **GKPIDDSI_HYPERLINKSCHANGED** property as FALSE because the applications write out an updated **"_PID_HLINKS"** property, if applicable, in the User Defined Property Set (section 2.3.3.2.3) when saving a document.

<44> Section 2.3.3.2.2.1: Word 97, Word 2000, Word 2002, Office Word 2003, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Excel 97, Excel 2000, Excel 2002, and Office Excel 2003 write their respective application **minor version** numbers as the minor version number portion of the **GKPIDDSI_VERSION** property.

<45> Section 2.3.3.2.2.1: PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 use this property for storing their VBA **digital signature**. Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 store their VBA digital signature in an unrelated location and will not write this property. Word 97, PowerPoint 97, and Excel 97 do not write digital signatures for VBA.

<46> Section 2.3.3.2.2.1: Word 97, Word 2000, Word 2002, Office Word 2003, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Excel 97, Excel 2000, Excel 2002, and Office Excel 2003 do not write this property, but preserve an existing value if it already exists. Office Word 2007, Word 2010, Word 2013, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Office Excel 2007, Excel 2010, and Excel 2013 will write this property if opening a file from or saving a file to a Windows SharePoint Services document library, in which case they will write the value "document".

<47> Section 2.3.3.2.2.1: Word 97, Word 2000, Word 2002, Office Word 2003, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Excel 97, Excel 2000, Excel 2002, and Office Excel 2003 do not write this property, but preserve an existing value if it already exists.

<48> Section 2.3.3.2.2.1: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 do not write this property, but preserve an existing value if it already exists.

<49> Section 2.3.3.2.2.1: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 do not write this property, but preserve an existing value if it already exists.

<50> Section 2.3.3.2.3.2: Excel 97 and Word 97 write a random **GUID** for the **_PID_GUID** property if the property does not yet exist. PowerPoint 97 writes a random GUID for this property on document creation only.

<51> Section 2.3.3.2.3.2: The **_MarkAsFinal** property applies only to Office Excel 2007, Excel 2010, Excel 2013, Office Word 2007, Word 2010, Word 2013, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013. Other applications do not write this property and ignore it on read, but preserve an existing value.

<52> Section 2.3.3.2.3.2: The **Microsoft Theme** property applies only to Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013. Other applications do not write this property and ignore it on read, but preserve an existing value.

<53> Section 2.3.3.2.3.2: The **Presentation** property applies only to Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013. Other applications do not write this value and ignore it on read, but preserve an existing value.

<54> Section 2.3.3.2.3.2: The **SlideDescription** property applies only to Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013. Other applications do not write this value and ignore it on read, but preserve an existing value.

<55> Section 2.3.3.2.3.2: The **_AllowSignedDocumentWithoutReadOnly** property applies only to Office Excel 2007, Excel 2010, Excel 2013, Office Word 2007, Word 2010, Word 2013, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013. Previous versions of the applications ignore this property, but preserve an existing value.

<56> Section 2.3.3.2.3.2: The **_TemplateID** property applies only to Office Word 2003, Office PowerPoint 2003, Office Excel 2003, Office Word 2007, Office PowerPoint 2007, Office Excel 2007, Word 2010, Word 2013, PowerPoint 2010, PowerPoint 2013, Excel 2010, and Excel 2013. Previous versions of the applications ignore this property but preserve an existing value.

<57> Section 2.3.3.2.3.2: Unless otherwise specified, these properties apply only to Office Word 2003, Office PowerPoint 2003, Office Excel 2003, Office Word 2007, Office PowerPoint 2007, Office Excel 2007, Word 2010, Word 2013, PowerPoint 2010, PowerPoint 2013, Excel 2010, and Excel 2013. Previous versions of the applications do not write these values and ignore them on read, but will preserve an existing value.

<58> Section 2.3.3.2.3.2: The **_CheckOutSrcUrl** property applies only to Office Word 2007, Office PowerPoint 2007, Office Excel 2007, Word 2010, Word 2013, PowerPoint 2010, PowerPoint 2013, Excel 2010, and Excel 2013. Previous versions of the applications do not write this value and ignore it on read, but will preserve an existing value.

<59> Section 2.3.3.2.3.2: Regardless of case, "none" is considered an invalid identifier for a **smart document** solution by Office Word 2003, Office Word 2007, Word 2010, Word 2013, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013.

<60> Section 2.3.3.2.3.2: Office Word 2003, Office Word 2007, Word 2010, Word 2013, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 read and write this property. Other applications do not write this value and ignore it on read, but will preserve an existing value.

<61> Section 2.3.3.2.3.2: Office Word 2003, Office Word 2007, Word 2010, Word 2013, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 read and write this property. Other applications do not write this value and ignore it on read, but will preserve an existing value.

<62> Section 2.3.3.2.3.2: Office Word 2003, Office Word 2007, Word 2010, and Word 2013 read and write this property. Other applications do not write this value and ignore it on read, but will preserve an existing value.

<63> Section 2.3.3.2.3.2: Regardless of case, "none" is considered an invalid identifier for a solution by Office Word 2003, Office Word 2007, Word 2010, and Word 2013.

<64> Section 2.3.3.2.3.2: Office Word 2003, Office Word 2007, Word 2010, and Word 2013 read and write this property. Other applications do not write this value and ignore it on read, but will preserve an existing value.

<65> Section 2.3.3.2.3.2: For Office Word 2003, Office Excel 2003, Office Word 2007, Office Excel 2007, Word 2010, Word 2013, Excel 2010, and Excel 2013 documents, these solutions are generated through Visual Studio Tools for Office document **add-in** solutions. **Managed code** document add-in solutions are not supported by other applications or versions.

<66> Section 2.3.3.2.3.2: Office Word 2003, Office Excel 2003, Office Word 2007, Office Excel 2007, Word 2010, Word 2013, Excel 2010, and Excel 2013 interpret this property and support managed code document add-in solutions. Other applications ignore the property on read, but will preserve an existing value.

<67> Section 2.3.3.2.3.2: Office Word 2003, Office Excel 2003, Office Word 2007, Office Excel 2007, Word 2010, Word 2013, Excel 2010, and Excel 2013 interpret this property and support managed code document add-in solutions. Other applications ignore the property on read, but will preserve an existing value.

<68> Section 2.3.3.2.3.2: Unless otherwise specified, these properties apply only to Word 2002, PowerPoint 2002, Excel 2002, Office Word 2003, Office PowerPoint 2003, Office Excel 2003, Office Word 2007, Office PowerPoint 2007, Office Excel 2007, Word 2010, Word 2013, PowerPoint 2010, PowerPoint 2013, Excel 2010, and Excel 2013. Previous versions of the applications do not write these values and ignore them on read, but will preserve an existing value.

<69> Section 2.3.3.2.3.2: Office Word 2003, Office PowerPoint 2003, Office Excel 2003, Office Word 2007, Office PowerPoint 2007, Office Excel 2007, Word 2010, Word 2013, PowerPoint 2010, PowerPoint 2013, Excel 2010, and Excel 2013 read and write this property. Previous versions of the applications do not write this value and ignore it on read, but will preserve an existing value.

<70> Section 2.3.3.2.3.2: Office Word 2003, Office PowerPoint 2003, Office Excel 2003, Office Word 2007, Office PowerPoint 2007, Office Excel 2007, Word 2010, PowerPoint 2010, Excel 2010, Word 2013, PowerPoint 2013, and Excel 2013 ignore this property unless the registry key **HKEY_CURRENT_USER\Software\Microsoft\Office\<*version*>\Outlook\Options\Mail** has a **DWORD** value named "AdHocReviewBehavior" with a value of 0, where <*version*> is "11.0" for Office Word 2003, Office PowerPoint 2003, and Office Excel 2003; "12.0" for Office Word 2007, Office PowerPoint 2007, and Office Excel 2007; "14.0" for Word 2010, PowerPoint 2010, and Excel 2010; or "15.0" for Word 2013, PowerPoint 2013, and Excel 2013.

<71> Section 2.3.3.2.3.3: Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, and Word 2013 create links to bookmarks for their linked properties, where the link entry's value is the name of the bookmark. Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013 create links to named ranges for their linked properties, where the link entry's value is the name of the range. PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, and PowerPoint 2013 create links to bookmark portions of slide content for their linked properties, where the link entry's value is the identifier of the bookmarked slide content.

<72> Section 2.3.6: Word 97, Word 2000, Word 2002, Office Word 2003, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Excel 97, Excel 2000, Excel 2002, and Office Excel 2003 do not create this **storage**.

<73> Section 2.3.6.1.4: Custom XML data item with the target namespace http://schemas.microsoft.com/office/2006/coverPageProps is only written by Office Word 2007, Word 2010, and Word 2013.

<74> Section 4.3: If the digest data of the digital signature is not MD5 encoded, the signature will not be verified by existing versions of Word 97, Word 2000, Word 2002, Office Word 2003, Office Word 2007, Word 2010, Word 2013, PowerPoint 97, PowerPoint 2000, PowerPoint 2002, Office PowerPoint 2003, Office PowerPoint 2007, PowerPoint 2010, PowerPoint 2013, Excel 97, Excel 2000, Excel 2002, Office Excel 2003, Office Excel 2007, Excel 2010, and Excel 2013.

6 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
2.3.3.2.1.1 PIDSI	Clarified product behavior related to GKPIDSI_WORDCOUNT.	Minor
2.3.3.2.3.2 Reserved Properties	Added descriptions of new classification properties.	Major

7 Index

A

C

D

R

S

T

U

V