MICROSOFT OFFICE POWERPOINT 97-2007 BINARY FILE FORMAT SPECIFICATION

[*.ppt]

Includes Binary File Format Documentation

Relevant To:

Microsoft Office PowerPoint 2007

Microsoft Office PowerPoint 2003

Microsoft Office PowerPoint 2002

Microsoft Office PowerPoint 2000

Microsoft Office PowerPoint 1997

Microsoft Office PowerPoint 97-2007 Binary File Format (.ppt) Specification

Contents

Introduction

Microsoft PowerPoint for Windows 97 uses OLE 2 compound files; this is the OLE implementation of the Structured Storage Model standard. An OLE 2 compound file is "a file system within a file"; it contains a hierarchical system of storages and streams. A storage is analogous to a directory because it holds other storages and streams, and a stream is analogous to a file because it holds information but no other storage elements. For more information on this technology, please refer to http://download.microsoft.com/download/0/B/E/0BE8BDD7-E5E8-422A-ABFD-4342ED7AD886/WindowsCompoundBinaryFileFormatSpecification.pdf.

Purpose and Scope

This document describes the PowerPoint 97-2007 file format, and it is intended for use by developers of applications that interact with PowerPoint files. This document is a programming and technical reference. It assumes familiarity with both PowerPoint and a high level programming language like C, C++ or Visual Basic.

Vocabulary

- ***Collections:*** Sets of objects. Objects within the set are referenced by their index in the set.
- ***External objects:*** Objects that can be brought into PowerPoint using the Insert Object dialog. This includes pictures, sounds, movies, etc.
- ***Master Coordinates:*** The reference system used by PowerPoint to put all objects on the screen. The origin for the system is the center of the slide. There are two axes, X (horizontal) and Y (vertical). Values on the X axis increase when you move to the right and the origin is 0. Values on the Y axis increase when moving down. Master coordinates are always 576 dpi.
- ***View:*** Refers to the way a presentation is seen on the screen at a particular moment. This includes the current view, whether the guides or rulers are visible, and the view scale.

Abbreviations

The following abbreviations are used throughout the document:

BOOL1: *Boolean* one-byte value.

UBYTE: Unsigned one-byte value.

UINT2: Unsigned two-byte integer value.

UINT4: Unsigned four-byte integer value.

SINT2: Signed two-byte integer value.

SINT4: Signed four-byte integer value.

Additions for PowerPoint 2007

Several records were added to the binary file format with the release of PowerPoint 2007. PowerPoint 2007 introduced a new XML-based file format. While this is the default format for documents saved by PowerPoint 2007, PowerPoint 2007 also provides the capability to save files to the binary PowerPoint file format used in previous versions.

Several new records were added to the binary file format to store information about documents authored in PowerPoint 2007. This release of the PowerPoint binary file format

documentation includes each of the records added to the format in PowerPoint 2007. Each of these records is used to store information about features specific to PowerPoint 2007 and later versions. This data is preserved in the binary format so that when reopened in PowerPoint 2007 or later, documents will retain data and features that are only available in the newer versions.

The description of each new record begins with the note, “Added in PowerPoint 2007.” Many of these records are used to store XML data from the new XML-based format where the binary file format has no records in which store equivalent information. Most of these records are variable length containers that contain an XML package that is equivalent to a ZIP file. Within the ZIP file are XML parts that contain snippets of XML. Details about the container format and the meaning of XML data within these new records may be found in the publicly available Office Open XML specification (Ecma Internation Standard 376). Information about the XML elements relevant to PowerPoint exists in the PresentationML and DrawingML sections of that documentation.

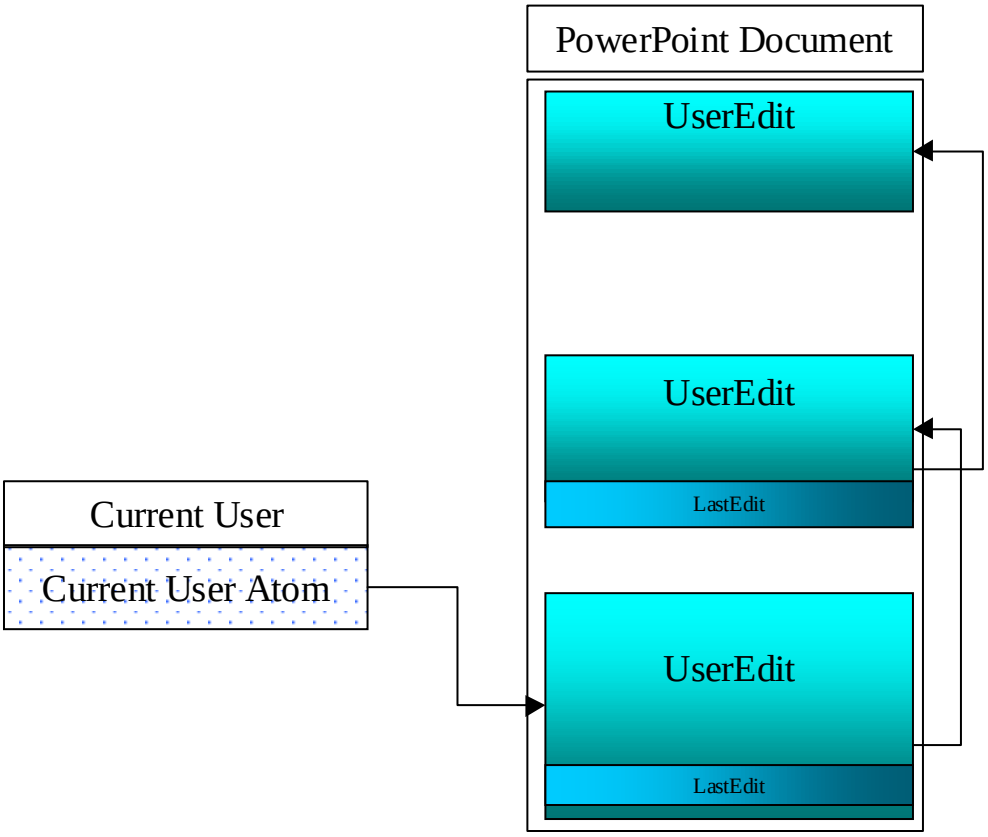
File Format Overview

PowerPoint 97 files are OLE DocObject files consisting of the following streams:

- **Current User** - Keeps the name of the user who last opened the presentation.
- **PowerPoint Document** - Keeps all of the information about a PowerPoint presentation. This document explains its layout and contents.
- **Pictures (Optional)**– Contains data about the pictures (metafiles, PNG, JPG, etc) contained in a PowerPoint presentation.
- **Summary Information and DocumentSummaryInformation (Optional)** - Keeps statistics about the document, following a Microsoft Office standard. .

Current User Stream

The Current User Stream contains a pointer to the latest saved edit in the document stream. The document stream contains one or more user edit structures. A graphical representation of this looks like:



UserEditAtom Structure

The UserEditAtom structure is as follows:

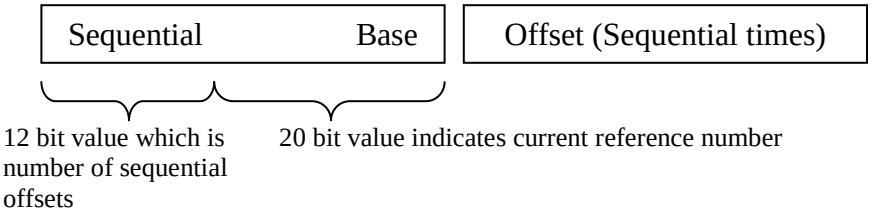
```
struct PSR_UserEditAtom
{
   sint4  lastSlideID;    // slideID of last viewed slide
   uint4  version;        // This is major/minor/build which
did the edit
   uint4  offsetLastEdit; // File offset of last edit
   uint4  offsetPersistDirectory; // Offset to PersistPtrs for
this edit.
   uint4  documentRef; // reference to document atom
   uint4  maxPersistWritten;      // Addr of last persist ref
written to the file (max seen so far).
   sint2  lastViewType;   // enum view type
};
```

UserEditAtom Element Descriptions

- **lastSlideID and lastViewType:** SlideID of last slide viewed and view type for saved view, respectively. Allow a document window to be opened in its saved configuration.

- **version:** Major/minor/build which did the edit.
- **offsetLastEdit:** Pointer to the last user edit. This is a 32 bit fixed offset from the beginning of the file. (This is 0 if no previous edits exist. It is illegal to place a LastEdit structure at offset 0 in the file.)
- **offsetPersistDirectory:** Contains the persistent references (32 bit offset from the beginning of the document stream) in the current user edit. References are number sequentially from 1 (0 is not a valid value) and each user edit will contain a persistent directory. This directory contains only the references made by the current user and the document data included in the edit. To find additional references, PowerPoint begins with the directory of the last edit and then searches recursively through the previous edits until the reference is found.

  The persistent directory is encoded as follows:



- **documentRef:** Reverence to the document atom.
- **MaxPersistWritten:** Address of the last persist ref written to the file. This is the maximum value contained in the file, maintained so that new user edits can be properly numbered.

Persistent Directory Example

Suppose the current save of a PowerPoint document contains the following:

Reference	File Offset
1	1024
2	2048
3	4096
6	8196
8	10000
9	20000

The following would be saved to the file:

Hex	Decimal	Meaning
1772	6002	PST_PersistPtrIncrementalBlock
24	36	Length of Atom
300001	3145729	3 consecutive offsets starting at 1
400	1024	Offset to ref(1)
800	2048	Offset to ref(2)
1000	4096	Offset to ref(3)

100006	1048582	1 consecutive refs starting at 6
2000	8192	Offset to ref(6)
200008	2087160	2 consecutive refs starting at 8
2710	10000	Offset to ref(8)
4E20	20000	Offset to ref(9)

For an example of an application that tracks user edits see appendix B.

PowerPoint Document Stream

The PowerPoint Document Stream keeps all the information about a PowerPoint presentation. A PowerPoint file stores its data in records (see Appendix B). There are two different kinds of records in a file: atoms and containers. We could, as with storages and streams, compare atoms and containers to files and directories, respectively. Atoms, like files, keep the actual information. Containers, just like directories, can contain files and other directories.

- **Atoms**: Records that contain information about a PowerPoint object and are kept inside containers.
- **Containers:** Records that keep atoms and other containers in a logical and organized way.

A Slide

A typical PowerPoint file will have Slide containers. A Slide container keeps all the atoms and containers necessary to describe a single PowerPoint slide.

Physical File Format

Each record, weather its an atom or a container, has a Record Header. The record header is a structure defined as follows:

```
struct RecordHeader
{
   psrVersion     recVer      : 4
   psrInstance    recInstance : 12;
   psrType        recType;
   psrSize        recLen;
};
```

Record Version: (recVer) Indicates the version if the record is an atom. All versions are prefixed by VER and are enumerated in Appendix B. If the record is a container, this field has a value of 0xFFFF.

Record Instance: (recInstance) Differentiates atoms. Depending on the instance a record's contents it can have different meanings. For example a list container can store a list of slides or a list of fonts, and it's instance would vary accordingly. Instances are prefixed by INS (see Appendix B). The instance of a record is useful for differentiating atoms when there is more than one atom of the same type in a particular container.

Record Type: (recType) Indicates the signature or type of the record. Each record has a symbolic and a numeric signature (see Appendix B). All the symbolic signatures are prefixed by PST. For example, the symbolic signature for a slide is PST_Slide which has a value of 1006. A description of each of the different types can be found in the Record Descriptions section.

Record Length: (recLen) Stores the length of the record in bytes. If the record is an atom, it refers to the length of the atom excluding the header. If the record is a container, it refers to the sum of the lengths of the atoms inside it, plus the length of the record headers.

Record Descriptions

This section describes each of the storage types listed in Appendix B. It contains the symbolic and numeric signature for each record. It is organized alphabetically by symbolic signatures, with the numeric signatures in parentheses next to it. For an index organized by number, please refer to Appendix A.

As stated before there are two kinds of storage elements in a PowerPoint file: atoms and containers. Atoms are described by indicating each of the fields' contents and their meaning. An atom's description is done in this section using types and offsets; but it is done using C++ language syntax in Appendix B. Containers are described in this section by indicating their use and the atoms and containers that they hold.

AnimationAtom12 (11019)

Added in PowerPoint 2007.

A variable length container which contains animation XML for a slide. The purpose of this record is that when we open the file back in PowerPoint 2007 we can correctly restore the PowerPoint 2007 animations for a slide.

The data is actually a package in Office Open XML format, which can be simply opened as a zip file. The package's main part contains the XML for a <timing> element that conforms to the schema defined by CT_SlideTiming. The package may also contain parts for embedded sounds referenced within the CT_SlideTiming XML.

For more information about the xml data representing animations, refer to the Office Open XML PresentationML documentation.

AnimationHashAtom12 (11021)

Added in PowerPoint 2007.

An unsigned integer that contains a CRC Hash value that is used to determine whether animations or shapes for a slide have been modified in PowerPoint 2003 or below. The values hashed are the bytes of the binary stream that represent the animation timing tree as converted from the PowerPoint 2007 representation to the PowerPoint 2003 representation, followed by the bytes that represent the PowerPoint 2003 shape IDs of the shapes on the slide.

AnimationHash12 Fields

Offset	Type	Name	Contents
0	uint4	animationChecksum	Checksum for the animation

AnimationInfo (4116)

A container for information about animation. It contains:

1. AnimationInfoAtom (4081)
2. Sound (2022), optional

AnimationInfoAtom (4081)

An atom containing information about animation. This record is written out for binary compatibility with older PPT versions (PPT 2000 and PPT 97).

AnimationInfoAtom Fields

Offset	Type	Name	Contents
0	GrColorAtom	dimColor;	Color to use for dimming
4	uint4	flags	Set of flags that determine type of build: Bit 1: Reverse Bit 3: Automatic Bit 5: Sound Bit 7: StopSound Bit 9: Play Bit 11: Synchronous Bit 13: Hide Bit 15: AnimateBg
8	uint4	soundRef	0 if storage is from clipboard. Otherwise index(ID) in SoundCollection list.
12	sint4	delayTime	Delay before playing object in ms
16	uint2	orderID	Order of build: -2: Follow Master Slide Other: Order ID
18	uint2	slideCount	Number of slides to play object
20	sbyte1	buildType	Type of buildL 0: No Build 1: All at once 2: Build by Text Level 1 3: Build by Text Level 2 4: Build by Text Level 3 5: Build by Text Level 4 6: Build by Text Level 5 7: Graph by Series 8: Graph by Category 9: Element in Series 10: Element in Category
21	sbyte1	flyMethod	Animation effect: 0: None 1: Random 2: Blinds 3: Checker 4: Cover 5: Dissolve 6: Fade 7: Pull 8: Random Bar 9: Strips

			10: Wipe 11: Zoom 12: Fly 13: Split 14: Flash 15: (unused) 16: (unused) 17: Diamond 18: Plus 19: Wedge 20: Push 21: Comb 22: Newsflash 23: Alphafade 24: Blur 25: Pushelem 26: Wheel 27: Circle
22	sbyte1	flyDirection	Animation direction: 0: Left 1: Up 2: Right 3: Down 4: LeftUp 5: RightUp 6: LeftDown 7: RightDown 8: FromLeftEdge 9: FromBottomEdge 10: FromRightEdge 11: FromTopEdge 12: LeftSlow 13: UpSLow 14: RightSlow 15: DownSLow 16: ZoomIn 17: ZoomInSlightly 18: ZoomOut 19: ZoomOutSlightly 20: ZoomCenter 21: ZoomBottom 22: StretchAcross 23: StretchLeft 24: StretchUp 25: StretchRight

			26: StretchDown 27: Rotate 28: Spiral
23	sbyte1	afterEffect	What to do after build: 0: None 1: Dim 2: Hide 3: HideImmediately
24	sbyte1	subEffect	Additional effect info 0: None 1: Build by Word 2: Build by Letter
25	sbyte1	oleVerb	Determines object's class (sound, video, other)

BinaryTagData (5003)

A container for the binary value data of a Programmable Tag. Interpretation of its content is dependent of the Programmable Tag client.

Clients using Programmable Tags to store version dependent binary file format extensions:

1. Document (1000)
2. Handout (4041)
3. MainMaster (1016)
4. Notes (1008)
5. Slide (1006)
6. msofbtClientData

BlipCollection (2040)

A container for information about the pictures of all picture bullets in the presentation, It contains:

1. BlipEntity (1001)

BlipEntity (2041)

A container for information about a single picture bullet: It contains:

BlipEntity Fields

Offset	Type	Name	Contents
0	ubyte	winBlipType	Preferred format for this picture on windows operating systems
1	ubyte	macBlipType	Preferred format for this picture on Macinstosh operating systems

Follwing these, starting at offset 2, is a variable-length record containing the binary picture data. The format of this record is describe under the heading msofbtBlip* in the “Office Drawing Binary File Format specification”.

BookmarkCollection (2019)

A container for bookmark related atoms. Bookmarks are text links used mainly for exporting PowerPoint property fields to Lotus Notes fields or columns. The contents of a Bookmark Collection depend on whether the presentation has bookmarks or not. When the presentation doesn't have bookmarks, a BookmarkCollection contains only a BookmarkSeedAtom (2025). When the presentation has bookmarks, in addition it contains a set of a BookmarkEntityAtom (4048) and a CString (4026) for each bookmark:

1. BookmarkSeedAtom (2025), Instance BookmarkSeedAtom (2)
2. BookmarkEntityAtom (4048)
3. CString (4026), containing the value of the bookmark

BookmarkEntityAtom (4048)

Atom that tracks bookmarks.

BookmarkEntityAtom Fields

Offset	Type	Name	Contents
0	uint4	bookmarkID	Unique ID used to keep track of bookmarks.
4	uint2[32]	bookmarkName	User-friendly bookmark name

Note: There has to be a one-to-one correspondence between bookmarks in the PowerPoint data and in the properties saved by the properties dialog (which is done by Office). If PowerPoint detects any discrepancy between the two sets of data, PowerPoint will delete the bookmark. This situation can arise naturally if the user employs a third party tool to change the properties of a presentation.

BookmarkSeedAtom (2025)

This atom the seed bookmark ID. This ID is a number used internally by PowerPoint to compute a unique ID for the bookmark. If you are trying to create a new bookmark outside of PowerPoint, you should give the bookmark ID a number higher than this one.

BookmarkSeedAtom Fields

Offset	Type	Name	Contents
0	uint4	bookmarkID	Unique ID used to generate bookmark IDs.

BroadCastDocInfo9 (6014)

A container for per-document broadcast information. It contains:

1. CString (4026), Instance Title (1), optional
2. CString (4026), Instance Description (2), optional
3. CString (4026), Instance Speaker (3), optional
4. CString (4026), Instance Contact (4, optional
5. CString (4026), Instance RexServerName (5), optional
6. CString (4026), Instance EmailAddress(6), optional
7. CString (4026), Instance EmailName (7), optional
8. CString (4026), Instance ChatURL (8), optional
9. CString (4026), Instance ArchiveDir (9), optional

10. CString (4026), Instance NetShowFilesBaseDir (10), optional
11. CString (4026), Instance NetShowFilesDir (11), optional
12. CString (4026), Instance NetShowServerName (12), optional
13. CString (4026), Instance PPtFilesBaseDir (13), optional
14. CString (4026), Instance PptFilesDir (14), optional
15. CString (4026), Instance PptFilesBaseURL (15), optional
16. CString (4026), Instance UserName (16), optional
17. CString (4026), Instance BroadcastDateTime (17), optional
18. CString (4026), Instance PresentationName (18), optional
19. CString (4026), Instance AsdFileName (19), optional
20. CString (4026), Instance EntryID (20), optional
21. BroadcastDocInfoAtom (6015)

BroadCastDocInfoAtom (6015)

An atom for for per-document broadcast information. It contains:

BroadCastDocInfoAtom Fields:

Offset	Type	Name	Contents
0	uint2	flags	
2	uint2[8]	startTime	Time and date of the start of the broadcast Index 0: Year Index 1: Month Index 2: Day of week Index 3: Day Index 4: Hour Index 5: Minute Index 6: Second Index 7: Milliseconds
18	uint2[8]	endTime	Time and date of the end of the broadcast Same format as above

BuildAtom (11011)

An atom for general information about Builds. It contains:

BuildAtom Fields:

Offset	Type	Name	Contents
0	uint4	type	Type of Build 0: Undefined 1: Paragraph Build 2: Chart Build 3: Diagram Build
4	uint4	buildID	Unique Build ID. Build IDs are generated incrementally.
8	uint4	shapeID	ID identifying the Shape this Build belongs to

12	bool1	fExpanded	True, if the Build has been expanded
13	bool1	fUIExpanded	True, if the Build should be shown expanded in the UI

BuildList (11010)

A container for animation data related to Builds. It contains:

1. ChartBuild (11012), optional
2. DiagramBuild (11014), optional
3. ParaBuild (11016), optional

ChartBuild (11012)

A container for animation information about Chart Builds. It contains:

1. BuildAtom (11011)
2. ChartBuildAtom (11013)

ChartBuildAtom (11013)

An atom for animation information about Chart Builds. It contains:

ChartBuildAtom Fields:

Offset	Type	Name	Contents
0	uint4	buildType	Type of Chart Build: 0: Nonce 1: Series 2: Category 3: ElementInSeries 4: ElemenInCategory 5: Custom
4	bool1	fAnimBackground	

ColorMapping (1039)

Added in PowerPoint 2007.

A string containing the XML for a CT_ColorMapping element with the tag name “clrMap” or “clrMapOverride” if it is an override on a non-top-level slide. This represents the color mapping for a slide.

For more info about the xml color mapping data, refer to the Office Open XML DrawingML specification (Ecma Internation Standard 376).

ColorSchemeAtom (2032)

The color scheme atom is an array of 8 color references (COLORREF), which contain the RGB value for each color in the color scheme. The order of scheme colors is as in the custom tab of the Color Scheme dialog:

[0] Background
[1] Text and lines

[2] Shadows
[3] Title text
[4] Fills
[5] Accent
[6] Accent and hyperlink
[7] Accent and followed hyperlink

Comment10 (12000)

A container for information about specific comments. It contains:

1. CString (4026), Instance Author (0): Author of the comment
2. CString (4026), Instance Text (1): Text of the comment
3. CString (4026), Instance AuthorIndex (2): Initials of the author
4. CommentAtom10 (12201)

CommentAtom10 (12001)

An atom for information about specific comments. It contains:

CommentAtom10 Fields:

Offset	Type	Name	Contents
0	sint4	index	Index of the comment (the number after the initials)
4	uint2[8]	dateTime	Time and date of the comment Index 0: Year Index 1: Month Index 2: Day of week Index 3: Day Index 4: Hour Index 5: Minute Index 6: Second Index 7: Milliseconds
20	GPointAtom	anchor	Position of the comment

CommentIndex10 (12004)

A container for general information about comments. It contains:

1. CString (4026), Instance Author (0): Last author adding comments
2. CommentAtom10 (12201)

CommentIndexAtom10 (12005)

An atom for general information about comments. It contains:

CommentAtom10 Fields:

Offset	Type	Name	Contents
0	sint4	colorIndex	Last used color index for comments
4	sint4	seed	Last used index for comments

CompositeMasterId (1053)

Added in PowerPoint 2007.

A slide-level fixed-length record with single uint4. The presence of this record means that the slide is a PowerPoint 2007 content master merged with its PowerPoint 2007 main master. The PowerPoint 2007 main master is the main master with OriginalMainMasterId12 record with the same id.

CompositeMasterId12Atom Fields

Offset	Type	Name	Contents
0	uint4	compositeMasterID	Composite master id

CString (4026)

CString is a special container, its size is variable depending on the length of the string. CString characters are stored in UNICODE. The unit of the size is in bytes so it is twice the number of characters in the string.

CurrentUserAtom (4086)

This is written to the current user stream. The interpretation of the OffsetToCurrentEdit is crucial to locate the top level UserEditAtom.

CurrentUserAtom Fields:

Offset	Type	Name	Contents
0	uint4	size	sizeof(PSR_CurrentUserAtom)
4	uint4	magic	Magic number to ensure this is a PowerPoint file 0xE391C05F: PPT File 0xF3D1C4DF: Encrypted PPT File
8	uint4	offsetToCurrentEdit	Offset in main stream to current edit field
12	uint2	lenUserName	Length of user name
14	uint2	docFileVersion	1012 for PP97+
16	ubyte1	majorVersion	3 for PP97+
17	ubyte1	minorVersion	0 for PP97+
18	char1[lenUserName]	userName	ANSI version of the username
18+lenUserName	uint4	relVersion	Release version 8: Regular PPT File 9: PPT File contains multiple masters
22+lenUserName	char2[lenUserName	userName2	Unicode version of the username

DateTimeMCAtom (4087)

DateTimeMCAtom is an atom that stores the position of a date in a text and it also stores which of thirteen standard PowerPoint formats the date takes the form of. See the Date and Time dialog for all these different formats.

DateTimeMCAtom fields

Offset	Type	Name	Content
0	sint4	position	The position of the character in a text.
4	ubyte1	index	A number from 0-12 that specifies a date format.

DefaultRulerAtom (4011)

Used only within the PST_Environment container to store the default ruler for new texts. This atom is of variable length. It is equivalent to a PST_TextRulerAtom with all defined bits set in the mask.

DiagramBuild (11014)

A container for animation information about Diagram Builds. It contains:

1. BuildAtom (11011)
2. DiagramBuildAtom (11015)

DiagramBuildAtom (11015)

An atom for animation information about Diagram Builds. It contains:

DiagramBuildAtom10 Fields

Offset	Type	Name	Contents
0	uint4	buildType	Diagram Build Type: 0: None 1: DepthByNode 2: DepthByBranch 3: BreadthByNode 4: BreadthByLevel 5: ClockWise 6: ClockWiseIn 7: ClockWiseOut 8: CounterClockWise 9: CounterClockWiseIn 10: CounterClockWiseOut 11: InByRing 12: OutByRing 13: Up 14: Down 15: AllAtOnce 16: Custom

Diff10 (12013)

A container for collaboration info. It contains:

1. DiffAtom10 (12014)

DiffAtom10 (12014)

An atom for collaboration info. It contains information about the committed status of revisions to the document. It is a generic atom for various parts of the document. It contains:

DiffAtom10 Fields

Offset	Type	Name	Contents
0	bool1	fIndex	Has different meaning, depending on gmiTag field: Header/Footer (12): 0: Header/Footer is for Entire Document 1: Header/Footer is for Slide only

			InteractiveInfo (24): 0: OnMouseMove 1: OnMouseClick
1	uint4	gmiTag	Type of revision this atom relates to: 0: Document 1: Slide base 2: Slide 3: Main Master 4: Slide list 5: Master list 6: Shape list 7: Shape 8: (unused) 9: Text 10: Notes 11: SlideShow 12: Header/Footer 13: (unused) 14: Named show 15: Named show list 16: (unused) 17: (unused) 18: Recolor info 19: External object 20: (unused) 21: Table list 22: Table 23: InteractiveInfo
5	uint4	commit	Commit status 0: Not commited 1: Commited

DiffTree10 (12012)

A container for collaboration info. It contains:

1. CString (4026): Name of the reviewer this collaboration information was created by
2. Diff10 (12013)

DocFlags12 (1061)

Added in PowerPoint 2007.

Atom that tracks the Document level flags added in PowerPoint 2007.

BookmarkEntityAtom Fields

Offset	Type	Name	Contents
0	ubyte1	flags12	Bit1: Whether we compress pictures

			on save Bit2 – Bit8: not used so far

DocToolbarStatesAtom (14001)

An atom containing information about the state of Toolbars. It contains:

DocToolbarStatesAtom fields

Offset	Type	Name	Content
0	ubyte1	toolbarStates	Bit 1: Reviewing Toolbar Bit 2: Reviewing Gallery Toolbar

Document : Powerpoint Document (1000)

Document is a container that marks the beginning of the PowerPoint document. The documentRef field of the UserEditAtom (4085) entry points to this Document container. A Document container can also be part of a ProgTags (5000) container. It contains:

1. DocumentAtom (1001)
2. ExObjList (1033), optional
3. Environment (1010), Instance: DocEnvironment (0)
4. SoundCollection (2020), Instance: Sounds (5), optional
5. PPDrawingGroup (1035)
6. SlideListWithText (4080) , Instance: DocMasterList (1)
7. List (2000), Instance: DocInfoList (0)
8. SmartTagStore11 (14003), optional
9. OutlineTextProps11 (4021), optional
10. FontCollection10 (2005), optional
11. TxMasterStyle10Atom (4018), optional
12. TextDefaults10Atom (4020), optional
13. GridSpacingAtom10 (1037)
14. CommentIndex10 (12004), optional
15. FontEmbedFlags10 (13000), optional
16. CString (4026), Instance: Copyright (1), optional
17. CString (4026), Instance: Keywords (2), optional
18. FilterPrivacyFlags10 (14000), optional
19. OutlineTextProps10 (4019), optional
20. DocToolbarStatesAtom (14001), optional
21. SlideListTable10 (12017), optional
22. DiffTree10 (12012), optional
23. CString (4026), Instance: ModifyPswd (3), optional
24. PhotoAlbumInfoAtom (14002), optional
25. TxMasterStyle9Atom (4013), optional
26. BlipCollection (2040), optional
27. TextDefaults9Atom (4016), optional
28. SrKinsoku (4040), optional
29. ExHyperlink9 (4068), optional
30. PresAdvisoryFlags9 (6010), optional
31. HTMLDocInfoAtom (6011), optional

32. HTMLPublishInfo9 (6013), optional
33. BroadcastDocInfo9 (6014), optional
34. HeadersFooters (4057), Instance: SlideHeadersFooters (3), optional
35. HeadersFooters (4057), Instance: NotesHeaderFooters (4), optional
36. SlideListWithText (4080), Instance: DocSlideList (0), optional
37. SlideListWithText (4080), Instance: DocNotesList (2), optional
38. SSDocInfoAtom (1025), optional
39. NamedShows(1040), optional
40. Summary (1026), Instance: BookmarkCollecton (0), optional
41. PrintOptions (6000), optional
42. EndDocument (1002)
43. DocFlags12 (1061), optional
44. RoundTripCustomTableStyles12 (1064), optional

DocumentAtom (1001)

A document atom is a record that stores miscellaneous information about the PowerPoint presentation.

DocumentAtom Fields

Offset	Type	Name	Contents
0	GPointAtom	slideSize	Slide size in Master coordinates
8	GPointAtom	notesSize	Notes page size
16	GRatioAtom	serverZoom	The scale used when the Powerpoint document is embedded. The default is 1: 2
24	uint4	notesMasterPersist	Reference to NotesMaster (0 if none)
28	uint4	handoutMasterPersist	Reference to HandoutMaster(0 if none)
32	uint2	firstSlideNum	Number of the first slide
34	sint2	slideSizeType	Size of the document's slides. Valid values are from 0-6. *See SlideSize field values table below for valid values.*
36	bool1	saveWithFonts	indicates if document was saved with embedded true type fonts
37	bool1	omitTitlePlace	Set if the placeholders on the title slide are omitted
38	bool1	rightToLeft	Flag for Bidi version
39	bool1	showComments	Visibility of comment shapes

SlideSize Field Values

Value	Meaning
0	On screen
1	Letter sized paper
2	A4 paper
3	35mm
4	Overhead
5	Banner
6	Custom

EndDocument (1002)

Marks the end of the Document container. It has no content.

Environment (1010)

The container for shared text entities, such as fonts, styles, rulers, etc. This container has:

1. SrKinsoku (4040), Instance DocKinsoku (2), optional
2. FontCollection (2005), optional
3. TxCFExceptionAtom (4004), optional
4. TxPFExceptionAtom (4005), optional
5. DefaultRulerAtom (4011), optional
6. TxSpecialInfoAtom (4009), optional
7. TxMasterStyleAtom (4003)

ExAviMovie (4102)

A container to store data relating to an AVI movie. It contains:

1. ExVideo (4101)

ExCDAudio (4110)

A container to store data relating to CD audio. It contains:

1. ExMediaAtom (4100)
2. ExCDAudioAtom (4114)

ExCDAudioAtom (4114)

An atom containing information about CD audio. It contains:.

ExCDAudioAtom Fields

Offset	Type	Name	Contents
0	uint4	start	Start of audio, in TMSF format (frame:minute:second:track)
4	uint4	end	End of audio, in TMSF format (frame:minute:second:track)

ExControl (4078)

Container for OLE Control object. It contains:

1. ExControlAtom (4091)
2. ExOleObjAtom (4035)
3. CString (4026), Instance MenuName (1) used for menus and the Links dialog box.
4. CString (4026), Instance ProgID (2) that stores the OLE Programmatic Identifier. A ProgID is a string that uniquely identifies a given object.
5. CString (4026), Instance ClipboardName (3) that appears in the paste special dialog.
6. MetaFile(4033), optional

ExControlAtom (4091)

Contains a long integer, slideID, which stores the unique slide identifier of the slide where this control resides.

ExControlAtom Fields

Offset	Type	Name	Contents
0	uint4	slideID	Slide of this control

ExEmbed (4044)

A container for embedded objects. It contains:

1. ExEmbedAtom.(4045)
2. ExOleObjAtom (4035)
3. CString (4026), Instance MenuName (1) used for menus and the Links dialog box.
4. CString (4026), Instance ProgID (2) that stores the OLE Programmatic Identifier. A ProgID is a string that uniquely identifies a given object.
5. CString (4026), Instance ClipboardName (3) that appears in the paste special dialog.
6. MetaFile(4033), optional

ExEmbedAtom (4045)

This atom contains information about an embedded object.

ExEmbededAtom Fields

Offset	Type	Name	Contents
0	sint4	followColorScheme	This field indicates how the object follows the color scheme. Valid values are: 0 - doesn't follow the color scheme 1 - follows the entire color scheme 2 - follows the text and background scheme
4	bool1	cantLockServerB	Set if the embedded server can not be locked
5	bool1	noSizeToServerB	Set if don't need to send the dimension to the embedded object
6	Bool1	isTable	Set if the object is a Word table

ExHyperlink (4055)

A container for OLE Hyperlink objects. It contains:

1. ExHyperlinkAtom (4051)
2. CString (4026), Instance FriendlyName (0): The hyperlink's user-readable name
3. CString (4026), Instance INS_Target (1): The full path of the hyperlink destination file
4. CString (4026), Instance INS_Location (3): The hyperlink's location within the destination file

ExHyperlink9 (4068)

A container with addition information about OLE Hyperlink objects. It contains:

1. ExHyperlinkAtom (4051)
2. CString (4026), Instance 0, optional: Screen Tip of the Hyperlink
3. ExHyperlinkFlags (4120), Instance HlinkFlags (0)

ExHyperlinkAtom (4051)

This atom contains information about an OLE hyperlink object.

ExHyperLinkAtom Fields

Offset	Type	Name	Contents
0	sint4	objID	Unique external object identifier

ExHyperlinkFlags (4120)

This atom contains information about an OLE hyperlink object.

ExHyperLinkAtom Fields

Offset	Type	Name	Contents
0	uint4	flags	Bit 1: If set, Hyperlink was created through Insert Hyperlink dialog Bit 2: If set, Hyperlink is to Custom Show Bit 3: If set, Custom Show is set to return to Slide

ExLink (4046)

A container for OLE linked objects. It contains:

1. ExLinkAtom (4049)
2. ExOleObjAtom (4035)
3. CString (4026), Instance MenuName (1) used for menus and the Links dialog box.
4. CString (4026), Instance ProgID (2) that stores the OLE Programmatic Identifier. A ProgID is a string that uniquely identifies a given object.
5. CString (4026), Instance ClipboardName (3) that appears in the paste special dialog.
6. MetaFile(4033), optional

ExLinkAtom (4049)

This atom contains information about an OLE linked object.

ExLinkAtom Fields

Offset	Type	Name	Contents
0	uint4	slideID	Contains the slide Id the link refers to
4	uint4	updateMode	Stores the way the link is updated. This can be changed with the links dialog in the edit menu. The valid values are: 1 - automatic 3 - manual
8	bool1	unavailable	Set if the linked object is not available

ExMCIMovie (4103)

A container to store data relating to an MCI movie. It contains:

1. ExVideo (4101)

ExMediaAtom (4100)

An atom containing information about media external objects

ExMediaAtom Fields

Offset	Type	Name	Contents
0	uint4	exObjId	Unique external object identifier
4	uint2	flags	Bit1: Loop continuously Bit2: Rewind after play Bit3: Media is a narration

ExMIDIAudio (4109)

A container to store data relating to a MIDI audio. It contains:

1. ExMediaAtom (4100)
2. CString (4026), Instance 0: Path of the multimedia file

ExObjList (1033)

A container for all ExternalObjects in a document. It contains:

1. ExObjListAtom (1034)
2. ExAviMovie (4102), optional
3. ExCDAudio (4110), optional
4. ExControl (4078), optional
5. ExEmbed (4044), optional
6. ExHyperlink (4055), optional
7. ExLink (4046), optional
8. ExMCIMovie (4103), optional
9. ExMIDIAudio (4109), optional
10. ExQuickTimeMovie (4074), optional
11. ExSubscription (4076), optional
12. ExWAVAudioEmbedded (4111), optional
13. ExWAVAudioLink (4112), optional

ExObjListAtom (1034)

An atom containing information about the list of external objects

ExObjListAtom Fields

Offset	Type	Name	Contents
0	sint4	objectIdSeed	Hodlds the next unique identifier for the OLE objects

ExObjRefAtom (3009)

This atom is saved from the OEShape container and refers to external objects that are serialized in the ExObjList: It contains:

ExObjRefAtom Fields

Offset	Type	Name	Contents
0	uint4	exObjId	The unique Id of the external object

ExOleObjAtom (4035)

Atom that stores information for OLE objects.

ExOleObjAtom Fields

Offset	Type	Name	Contents
0	uint4	drawAspect	Corresponds to the DVASPECT enumeration (see http://msdn2.microsoft.com/en-us/library/ms690318.aspx
4	sint4	type	Specifies whether the object is embedded or linked. Valid values are: 0: Embedded 1: Linked 2: Control
8	sint4	objID	Unique identifier for the OLE object
12	sint4	subType	This specifies the type of ole object. *See subType Values table below.*
16	uint4	objStgDataRef	Reference to persist object
20	bool1	isBlank	Set if the object's image is blank

SubType Values

Value	Meaning
0	Default object
1	Microsoft Clipart Gallery
2	Microsoft Word table
3	Microsoft Excel
4	Microsoft Graph
5	Microsoft Organization Chart

6	Microsoft Equation Editor
7	Microsoft Wordart object
8	Sound
9	Imager
10	PowerPoint presentation
11	PowerPoint slide
12	Microsoft Project
13	Microsoft Note-It Ole
14	Microsoft Excel chart
15	Media Player object
16	WordPad

ExOleObjStg (4113)

A variable length container which contains the OLE object data. The data can be LZW compressed (Instance 1), in which case the first 4 bytes contain the length of the uncompressed data The data corresponds to the Istorage data for this ole object. The uncompressed data is a docfile,.

ExQuickTimeMovie (4074)

A container for Macintosh QuickTime movies. Quicktime movies are not supported on Windows, and cannot be played in PowerPoint for Windows. They appear only as pictures, and are stored only for fidelity in round-tripping. It contains:

1. ExVideo (4101)
2. ExQuickTimeMovieData (4075), Instance 0, optional
3. ExQuickTimeMovieData (4075), Instance 1, optional

ExQuickTimeMovieData (4075)

This exists for round-tripping QuickTime movies. A record header with this type is followed by a record consisting of native Macintosh QuickTime movie data.

ExVideo (4101)

A container for Video external object related information. It contains:

1. ExMediaAtom (4100)
2. CString (4026), Instance 0: Path of the multimedia file.

ExWAVAudioEmbedded (4111)

A container for information about WAV audio who's content is included in the presentation. It contains:

1. ExMediaAtom (4100)
2. ExWavAudioEmbeddedAtom (4115), optional
3. Sound (2022), optional

ExWAVAudioEmbeddedAtom (4115)

ExWAVAudioEmbeddedAtom fields

Offset	Type	Name	Content
0	sint4	soundId	persistent reference to an object in the sound

			collection
4	sint4	soundLength	length of the sound clip in milliseconds

ExWAVAudioLink (4112)

A container for information about WAV audio who's content is not included in the presentation. It contains:

1. ExMediaAtom (4100)
2. CString (4115), optional: Path of the WAV audio

FilterPrivacyFlags10 (14000)

An atom containing information about privacy settings. It contains:

FilterPrivacyFlags10 fields

Offset	Type	Name	Content
0	sint4	flags	Bit 1: If set, personal information gets removed upon save

FontCollection (2005)

A container holding information about all the fonts in the presentation. It contains:

1. FontEntityAtom (4023), optional
2. FontEmbedData (4024), optional

FontCollection10 (2006)

A container holding additional information about fonts in the presentation. It contains:

1. FontEntityAtom (4023), optional
2. FontEmbedData (4024), optional

FontEmbedData (4024)

An atom containing data about an embedded font, Instance contains the font index.

FontEmbedFlags10 (13000)

An atom containing additional flags about an embedded font, It contains:

FontEmbedFlags10 fields

Offset	Type	Name	Content
0	sint4	flags	Bit 1: Embedded font is subsetted Bit 2: Subsetting has been confirmed

FontEntityAtom (4023)

This atom corresponds in part to a Windows Logical Font (LOGFONT) structure. It keeps information needed to define the attributes of a font, such as height, width, etc. For more information, consult the Windows API Programmer's reference.

FontEntityAtom's fields

Offset	Type	Name	Content
0	uint2[32]	lfFaceName	Corresponds to the lfFacename field of the

			LOGFONT structure
64	ubyte1	lfCharSet	Corresponds to the lfCharSet field of the LOGFONT structure
65	ubyte1	flags	Bit 1: If set, font is subsetted
66	ubyte1	fontType	Bit 1: Raster Font Bit 2: Device Font Bit 3: TrueType Font
67	ubyte1	lfPitchAndFamily	Corresponds to the lfPitchAndFamily field of the LOGFONT structure

FooterMCAtom (4090)

FooterMCAtom is an atom that stores the position of the footer meta character in the text. It needs no more information because this meta character is replaced by the footer string stored in the header and footer structure of the slide. The FooterMCAtom is only used in the footer placeholder on the slide, title, notes, and handout masters.

FooterDateMCAtom's fields

Offset	Type	Name	Content
0	sint4	position	The position of the character in a text.

GenericDateMCAtom (4088)

GenericDateMCAtom is an atom that stores the position of the generic date character in the text. It needs no more information because this meta character is replaced by the date string stored in the header and footer structure of the slide. The GenericDateMC is only used in one of the header and footer placeholders on slide, title, notes, and handout masters.

GenericDateMCAtom's fields

Offset	Type	Name	Content
0	sint4	position	The position of the character in a text.

GPointAtom (3034)

This atom keeps the master coordinates of a point. This atom does not occur in the file by itself but will always be part of another atom.

GPointAtom Fields

Offset	Type	Name	Contents
0	sint4	x	x coordinates
4	sint4	y	y coordinates

GRatioAtom (3031)

A Ratio Atom keeps the ratio of one quantity to another. This atom does not occur in the file by itself but will always be part of another atom.

GPointAtom Fields

Offset	Type	Name	Contents
0	sint4	numer	Numerator
4	sint4	denom	Denominator

GridSpacingAtom10 (1037)

An atom containing information about grid spacing. It contains:

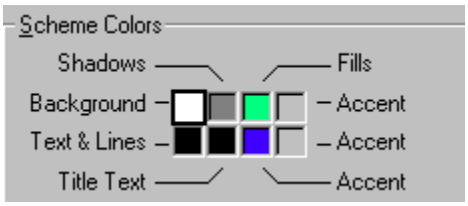
GridSpacingAtom Fields

Offset	Type	Name	Contents
0	sint4	x	Spacing along the X axis in master coordinates
4	sint4	y	Spacing along the Y axid in master coordinates

GrColorAtom (10002)

This atom does not occur in the file by itself, but it occurs inside other atoms. It contains an index into the Scheme Collection or an RGB color as indicated by its index field.

GRColorAtom Fields

Offset	Type	Name	Contents
0	ubyte1	red	Red value (0 - 255)
1	ubyte1	green	Green value (0 - 255)
2	ubyte1	blue	Blue value (0 - 255)
3	ubyte1	index	If this field has a value of OxFE, then the color is an RGB value. If not, it contains an index into the color scheme, with each value describing a color in the Scheme Colors dialog :  See Scheme Colors table below for valid values. This field can have a value of 0xFF if the color is undefined.

Scheme Color Values

Value	Meaning
0	Background
1	Text and Lines
2	Shadows

3	Title Text
4	Fills
5	Accent 1
6	Accent 2
7	Accent 3

GScalingAtom (10001)

This atom does not occur in a file by itself, but it occurs inside other atoms. It represents a scale using two ratios.

GScalingAtom Fields

Offset	Type	Name	Contents
0	PSR_GRatioAtom	x	x axis scaling
8	PSR_GRatioAtom	y	y axis scaling

GuideAtom (1019)

This atom stores information about the guides in a slide.

GuideAtom Fields

Offset	Type	Name	Contents
0	sint4	type	Type of the guide:. 0: Horizontal 1: Vertical
4	sint4	pos	Position of the guide in master coordinates. X coordinate if it's vertical, and Y coordinate if it's horizontal.

Handout (4041)

This is a container that keeps the information about the handout master. It contains

1. PPDrawing (1036)
2. ColorSchemeAtom (1013), Instance SlideScheme (1)
3. CString (4026), Instance SlideName (3), optional
4. ProgTags (5000), optional
5. Comment10 (12000), optional
6. LinkedSlideAtom10 (12007), optional
7. LinkedShapeAtom10 (12006), optional
8. SlideFlags10 (12010), optional
9. SlideTimeAtom10 (12011), optional
10. HashCodeAtom (11008), optional
11. BuildList (11010), optional
12. Theme (1038), optional
13. ColorMapping (1039), optional
14. HeaderFooterDefaults12 (1060), optional

HashCodeAtom (11008)

An atom preceding animation data.

HashCodeAtom fields

Offset	Type	Name	Content
0	uint4	hash	Hash code of animation data

HeaderMCAtom (4089)

HeaderMCAtom is an atom that stores the position of the header meta character in the text. It needs no more information because this meta character is replaced by the header string stored in the header and footer structure of the slide. The HeaderMCAtom is only used in the header placeholder on the slide, title, notes, and handout masters.

HeaderDateMCAtom fields

Offset	Type	Name	Content
0	sint4	position	The position of the character in a text.

HeaderFooterDefaults12 (1060)

Added in PowerPoint 2007.

This slide-level record is used to round-trip the PowerPoint 2007 introduced header/footer defaults. Those are flags on the master slides that control the instantiation of headers/footers when new slides/notes are added to the presentation. The flags are packed in single ubyte1.

HeaderFooterDefaults12 Fields

Offset	Type	Name	Contents
0	ubyte1	headerFooterFlags	Bit 1: Date Bit 2: Footer Bit 3: Header Bit 4: Slide number

HeadersFooters (4057)

A container for information related to Headers and Footers. It contains:

1. HeadersFootersAtom.(4058)
2. CString (4026), Instance UserDate (0), optional: Stores the user's date. This is the date that the user wants in the footers, instead of today's date.
3. CString (4026), Instance Header (1), optional: Stores the Header's contents.
4. CString (4026), Instance Footer (2), optional: Stores the Footer's contents.

HeadersFootersAtom (4058)

HeadersFootersAtom stores the basic information of the header and footer structure. It contains:

HeadersFootersAtom fields

Offset	Type	Name	Content
0	sint2	formatId	one of the 13 possible formats for the date. 0-12. See the Date and Time Dialog for details.
2	uint2	flags	Content of the Header/Footer: Bit 1: Date

			Bit 2: Today Date Bit 3: User Date Bit 4: Slide number Bit 5: Header Bit 6: Footer

HTMLDocInfoAtom (6011)

This atom keeps information about HTML save settings. It contains.

HTMLDocInfoAtom Fields

Offset	Type	Name	Contents
0	uint4	unused	unused
4	uint4	encoding	Specifies the code page, e.g. UTF8
8	sint2	frameColorType	Color of Slide navigation controls: 0: Browser colors 1: Presentation Text colors 2: Presentation Accent colors 3: White text on black 4: Black text on white
10	sint2	screenSize	Target Screen resolution: 0: 544x376 (WebTV) 1: 640x480 2: 720x512 3: 800x600 4: 1024x768 5: 1152x882 6: 1152x900 7: 1200x1024 8: 1600x1200 9: 1800x1440 10: 1920x1200
12	ubyte1	outputType	Target Browser: 1: IE3, Netscape 3 2: IE4+, Netscape 4+ 4: Both
13	ubyte1	flags	Bit 1: Show frame, if set Bit 2: Resize graphics, if set Bit 3: Organize in folders, if set Bit 4: Use long filenames, if set Bit 5: Rely on VML, if set Bit 6: Allow PNG, if set Bit 7: Show Slide animations, if set

HTMLPublishInfo (6013)

A container for information about Publish to HTML settings. It contains:

1. CString (4026), Instance FileName (0), optional: Name of the published presentation
2. CString(4026), Instance NamedShow (1), optional: Name of the Custom show
3. HTMLPublishInfoAtom (6012)

HTMLPublishInfoAtom (6012)

This atom keeps information about Publish to HTML settings. It contains:

HTMLPublishInfoAtom Fields

Offset	Type	Name	Contents
0	sint4	startSlide	Specifies start Slide, if Slide range is selected
4	sint4	endSlide	Specifies end Slide, if Slide range is selected
8	ubyte1	outputType	Color of Slide navigation controls: 0: Browser colors 1: Presentation Text colors 2: Presentation Accent colors 3: White text on black 4: Black text on white
9	ubyte1	flags	Bit 1: Use Slide range, if set Bit 2: Use named show, if set Bit 3: Open in browser, if set Bit 4: Show speaker notes, if set

InteractiveInfo (4082)

Interactive settings for mouse-over (Instance MouseOver (1)) and mouse-down (Instance MouseClick (0)) on an object in slideshow. It contains:

1. InteractiveInfoAtom (4083)
2. CString (4026), Instance MacroName (2), optional: Macro name
3. Sound (2022), optional. Only when serializing to Clipboard
4. ExHyperLink (4055), optional. Only when serializing to Clipboard

InteractiveInfoAtom (4083)

Interactive settings for mouse-over and mouse-down on an object in slideshow

InteractiveInfoAtom Fields

Offset	Type	Name	Contents
0	uint4	soundRef	a reference to a sound in the sound collection, or NULL.
4	uint4	exHyperlinkID	a persistent unique identifier to an external hyperlink object (only valid when action == HyperlinkAction).
8	ubyte1	action	See Action Table
9	ubyte1	oleVerb	Only valid when action ==

			OLEAction. OLE verb to use, 0 = first verb, 1 = second verb, etc.
10	ubyte1	jump	See Jump Table
11	ubyte1	flags	Bit 1: Animated. If 1, then button is animated Bit 2: Stop sound. If 1, then stop current sound when button is pressed. Bit 3: CustomShowReturn. If 1, and this is a jump to custom show, then return to this slide after custom show. Bit 4: If set, Interaction has been visited
12	ubyte1	hyperlinkType	a value from the LinkTo enum, such as LT_URL (only valid when action == HyperlinkAction).

Action Table:

Action	Value
NoAction	0
MacroAction	1
RunProgramAction	2
JumpAction	3
HyperlinkAction	4
OLEAction	5
MediaAction	6
CustomShowAction	7

Jump Table:

Jump	Value
NoJump	0
NextSlide,	1
PreviousSlide,	2
FirstSlide,	3
LastSlide,	4
LastSlideViewed,	5
EndShow	6
SlideId	7

LevelInfoAtom (11018)

An atom preceding per-Level Animation information for Paragraph Builds. It contains:

LevelInfoAtom Fields

Offset	Type	Name	Contents
0	uint4	level	Build level the Animation Information is for

LinkedShapeAtom10 (12006)

An atom containing collaboration information for Shapes. It contains:

LinkedShapeAtom10 Fields

Offset	Type	Name	Contents
0	sint4	shapeIndex	Shape ID
4	sint4	linkedIndex	ID of the linked Shape

LinkedSlideAtom10 (12007)

An atom containing collaboration information for Slides. It contains:

LinkedSlideAtom10 Fields

Offset	Type	Name	Contents
0	sint4	slideIndex	Slide ID
4	sint4	size	Number of LinkedShapeAtom10 following

List (1016)

A generic container for holding a variable number of containers or atoms, The following instances are defined:

DocInfoList (0)

This list can be part of a Document (1000) container. It contains:

1. SlideViewInfo (1018), optional
2. OutlineViewInfo (1031), optional
3. NotesTextViewInfo (43), optional
4. NormalViewSetInfo (44), optional
5. VBAInfo (1023), optional
6. ProgTags (5000), optional, multiple

MainMaster (2000)

This container represents the master slide in a presentation. As such, most of its contents are the ones that a Slide container would have, such as :

1. SlideAtom (1007)
2. ColorSchemeAtom (1013), Instance SchemeListElement (6). optional
3. TxMasterStyleAtom (4003), optional
4. SSSlideInfoAtom (1017), optional
5. HeadersFooters (4057), optional
6. ColorSchemeAtom (1013), Instance SlideScheme (1)
7. TxMasterStyle10Atom (4018), optional
8. PPDrawing (1036)
9. ColorSchemeAtom (1013), Instance SlideScheme (1)
10. CString (4026), Instance SlideName (3), optional
11. ProgTags (5000), optional, multiple
12. Comment10 (12000), optional
13. LinkedSlideAtom10 (12007), optional

14. LinkedShapeAtom10 (12006), optional
15. SlideFlags10 (12010), optional
16. SlideTimeAtom10 (12011), optional
17. HashCodeAtom (11008), optional
18. BuildList (11010), optional
19. TxMasterStyle9 (4013), optional
20. CString (4026), Instance TemplateName (2), optional
21. Theme (1038), optional
22. ColorMapping (1039), optional
23. OriginalMainMasterId (1052), optional
24. CompositeMasterId (1053), optional
25. RoundTripContentMasterInfo12 (1054), optional
26. RoundTripOArtTextStyles12 (1059), optional
27. HeaderFooterDefaults12 (1060), optional
28. AnimationAtom12 (11019), optional
29. AnimationHashAtom12 (11021), optional

MasterTextPropAtom (4002)

Same as PST_StyleTextPropAtom but used for the master text. Since the attributes of a master text by definition always reflect the attributes of the style, we simply store a runlist with demotion levels. This atom is of variable length, and consists of a series of paragraph formatting runs which cover the entire master text. For each of these runs, the paragraph formatting mask is zero. Only the demotion levels are used.

MasterTextPropAtom Fields

Type	Contents
uint4	Length of paragraph formatting run.
PF Run	Paragraph formatting run, with mask = 0 (see PST_SytleTextPropAtom).
Repeat until runs have been emitted for the entire text.	

MetaFile (4033)

This is an atom that occurs inside an ExEmbed or an ExLinkcontainer and is used for icons for linked or embedded OLE objects only. It contains a picture in a presentation stored as a 16-bit Windows metafile. It consists of a METAFILEPICT structure (more information can be found at http://msdn2.microsoft.com/en-us/library/ms649017(VS.85).aspx), followed by the variable length data of the metafile.

MsoCryptSession (12052)

An atom indicating an encrypted document (see Office Open XML specification (Ecma Internation Standard 376) for further details)

msofbtClientData

This container is not part of PPT's binary file format. It is part of and described in detail in the Office Drawing Binary File Format specification. It is mentioned here because it is the container for PPT specific Shape data. It contains:

1. OEShapeAtom (3035), optional
2. OEShapeFlagsAtom (3036), optional
3. ExObjRefAtom (3009), optional

4. AnimationInfo (4116), optional
5. InteractiveInfo (4082), Instance MouseClick (0), optional
6. InteractiveInfo (4082), Instance MouseOver (1), optional
7. OEPlaceholderAtom (3011), optional
8. RecolorInfoAtom (4071), optional
9. ProgTags (5000), optional
10. StyleTextProp11Atom (4022), optional
11. StyleTextProp10Atom (4017), optional
12. OEShapeHighPrecisionAnchor (12018), optional
13. StyleTextProp9Atom (4012), optional
14. RoundTripSahpeId12 (1055), optional
15. RoundTripHFPlaceholder12 (1056), optional
16. RoundTripShapeCheckSumForCustomLayouts12 (1062), optional
17. OEPlaceholderNewPlaceholderId12 (3037), optional

NamedShow (1041)

This atom represents one Custom Show. It contains:

1. CString (4026), representing the name of the Custom Show
2. NamedShowSlides (1042)

NamedShows (1040)

The NamedShows container holds a list of all Custom Shows in the presentation. It contains:

1. NamedShow (1041), optional

NamedShowSlides (1042)

An atom containing a list of slides in the Custom Show. It contains a variable number of slide id's (DWORD)

Notes (1008)

The Notes container is very similar to the Slide (1006) container and it represents the Notes pages of a presentation. It contains:

1. NotesAtom (1009)
2. PPDrawing (1036)
3. ColorSchemeAtom (1013), Instance SlideScheme (1)
4. CString (4026), Instance SlideName (3), optional
5. ProgTags (5000), optional
6. Comment10 (12000), optional
7. LinkedSlideAtom10 (12007), optional
8. LinkedShapeAtom10 (12006), optional
9. SlideFlags10 (12010), optional
10. SlideTimeAtom10 (12011), optional
11. HashCodeAtom (11008), optional
12. BuildList (11010), optional
13. Theme (1038), optional
14. ColorMapping (1039), optional
15. HeaderFooterDefaults12 (1060), optional

16. RoundTripNotesMasterTextStyles12 (1063), optional

NotesAtom (1009)

A NotesAtom stores the id of the slide that owns the notes.

NotesAtom Fields

Offset	Type	Name	Contents
0	sint4	slideID	Number that identifies the slide
4	uint2	flags	Bit 1: follow master objects Bit 2: follow master scheme Bit 3: follow master background

NormalViewSetInfo (1044)

This container keeps information about the normal view set. It contains:

NormalViewSetInfoAtom Fields

Offset	Type	Name	Contents
0	GRatioAtom	leftPortion	Position of the vertical Splitter Bar if the bar's state is Restored (1)
8	GRatioAtom	topPortion	Position of the horizontal Splitter Bar if the bar's state id Restored (1)
16	ubyte1	vertBarState	State of the vertical Spliter Bar: 0: Minimized (Top of area) 1: Restored (Normal position) 2: Maximized (Bottom of area)
17	ubyte1	horizBarState	State of the horizontal Splitter Bar: 0: Mnimized (left of area) 1: Restored (Normal position 2: Maximized (Right of area)
18	ubyte1	preferSingleSet	Bit 1: If set, show Slide in full window
19	ubyte1	showOutlineIcons	Bit 1: If set, show Outline Bit 2: If set, vertical Splitter Bar is snapped

NormalViewSetInfoAtom (1045)

This atom keeps information about the normal view set. It contains:

1. ViewInfoAtom (1021)

NotesTextViewInfo (1043)

This container keeps information about the view settings for Notes view. It contains:

1. ViewInfoAtom (1021)

OEPlaceholderAtom (3011)

Atom that describes the placeholder.

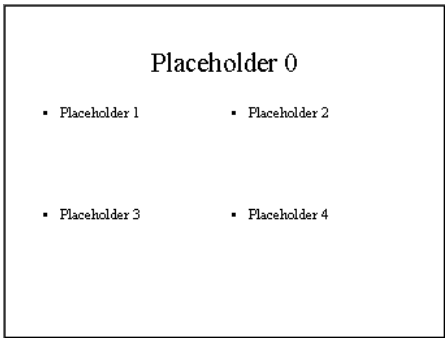
OEPlaceholderAtom Fields

Offset	Type	Name	Contents
0	uint4	placementId	The placement Id is a number assigned to the placeholder. It goes from -1 to the number of placeholders. *See note below.*
4	ubyte1	placeholderId	Type of placeholder. *See the Placeholder ID Values table below for valid values*.
5	ubyte1	size	Size of the placeholder, which can be: 0 - full size 1 - half size 2 - quart of the slide

Placeholder ID Values

Value	Type of Placeholder
0	None
1	Master title
2	Master body
3	Master centered title
4	Master subtitle
5	Master notes slide image
6	Master notes body
7	Master date
8	Master slide number
9	Master footer
10	Master header
11	Notes slide image
12	Notes body
13	Title
14	Body
15	Centered title
16	Subtitle
17	Vertical text title
18	Vertical text body
19	Object (no matter the size)
20	Graph
21	Table
22	Clip Art
23	Organization Chart
24	Media Clip

Note: The placementId is given in order from top to bottom, right to left. As an example, if we used the 4 object slide layout, the placement Ids would be as in the following picture:



There is a special case when the placeholder does not have a position in the layout. This occurs when the user has moved the placeholder from its original position. In this case the placeholder ID is -1.

OEPlaceholderNewPlaceholderId12 (3037)

Added in PowerPoint 2007.

A shape-level record used to round-trip the new placeholder ids for PowerPoint 2007 (picture placeholder & vertical object placeholder). Consists of a single ubyte1 that stores the PowerPoint 2007 placeholder id.

OEPlaceholderNewPlaceholderId12 Fields

Offset	Type	Name	Contents
0	ubyte1	newPlaceholderId	Placeholder Id

OEShapeAtom (3035)

Atom that contains information that describes shape client data.

OEShapeAtom Fields

Offset	Type	Name	Contents
0	ubyte1	flags	Bit 1: Always on top

OEShapeFlagsAtom (3036)

Atom that contains additional information that describes shape client data.

OEShapeAtom Fields

Offset	Type	Name	Contents
0	ubyte1	flags	Bit 3: Part of a Photo Album

OEShapeHighPrecisionAnchor (12018)

An atom containing information that describes higher precision Shape anchoring.

OEShapeAtom Fields

Offset	Type	Name	Contents
0	sint4	left	
4	sint4	top	
8	sint4	right	

12	sint4	bottom

OriginalMainMasterId (1052)

Added in PowerPoint 2007.

A slide-level fixed-length record with single uint4. The presence of this record indicates that the slide is a main master in PowerPoint 2007. The id is used to match the main master with a composite master (if such exists).

OriginalMainMasterId12Atom Fields

Offset	Type	Name	Contents
0	uint4	mainMasterId	Main master id

OutlineTextProps9 (4014)

A container for formatting information about Outline Texts of a Slide. For each Outline Text that needs formatting information, it contains:

1. OutlineTextPropsHeaderExAtom (4015)
2. StyleTextProp9Atom (4012)

OutlineTextProps10 (4019)

A container for formatting information about Outline Texts of a Slide. For each Outline Text that needs formatting information, it contains:

1. OutlineTextPropsHeaderExAtom (4015)
2. StyleTextProp10Atom (4017)

OutlineTextProps11 (4021)

A container for formatting information about Outline Texts of a Slide. For each Outline Text that needs formatting information, it contains:

1. OutlineTextPropsHeaderExAtom (4015)
2. StyleTextProp11Atom (4022)

OutlineTextPropsHeaderExAtom (4015)

An atom identifying an outline text. It contains

OutlineTextPropsHeaderExAtom Fields.

Offset	Type	Name	Contents
0	sint4	slideId	Slide the Outline Text belongs to
4	uint4	textType	Specifies the Outline Text type 0: Title 1: Body 2: Notes 3: Outline 4: Other 5: Center Body 6: Center Title 7: Half Body 8: Quarter Body

	9: Table

OutlineTextRefAtom (3998)

Appears in a slide to indicate a text that is already contained in the document, in a PST_SlideListWithText container. Sometimes slide texts are not contained within the slide container to be able to delay loading a slide and still display the title and body text in outline view

OutlineTextRefAtom Fields.

Offset	Type	Name	Contents
0	sint4	index	the text's index within the slide (0 for title, 1..n for the nth body)

OutlineViewInfo (1031)

This container keeps information about the view settings for Outline view. It contains:

1. ViewInfoAtom (1021)

ParaBuild (11016)

A container for animation information about Paragraph Builds. It contains:

1. BuildAtom (11011)
2. ParaBuildAtom (11013)
3. LevelInfoAtom (11018), optional, multiple

ParaBuildAtom (11017)

An atom for animation information about Paragraph Builds. It contains:

ParaBuildAtom Fields

Offset	Type	Name	Contents
0	uint4	paraBuildType	Paragraph Build Type: 0: All at once 1: Build by nth Level 2: Custom 3: As a whole
4	uint4	buildLevel	Specifies the level, if paraBuildType = Build by nth Level (1)
8	bool1	fAnimBackground	Animate background, if true
9	bool1	fReverse	Animation reverse, if true
10	bool1	fUserSetAnimBackground	User has set fAnimBackground (so don't change it), if true
11	bool1	fAutomatic	Automatic Build, if true
12	uint4	tDelay	Delay of Build in ms, if fAutomatic

PersistPtrFullBlock (6001)

Complete list of persists for this version. (For more information, see UserEditAtom Element Descriptions)

PersistPtrIncrementalBlock (6002)

Incremental diffs on persists. (For more information, see UserEditAtom Element Descriptions)

PhotoAlbumInfoAtom (14002)

An atom for information about a Photo Album. It contains

PhotoAlbumInfoAtom Fields

Offset	Type	Name	Contents
0	bool1	isBlackWhite	All picture are black and white, if set
1	bool1	hasCaption	All pictures have captions below, if set
2	ubyte1	layout	0: Fit to Slide 1: 1 picture 2: 2 pictures 3: 4 pictures 4: 1 picture with title 5: 2 pictures with title 6: 4 pictures with title
3	sint2	frameShape	0: Rectangle 1: Rounded rectangle 2: Beveled 3: Oval 4: Corner tabs 5: Square tabs 6: Plaque tabs

PPDrawing (1036)

Contains Office Drawing information. See the Office Drawing Format Specification for more information.

PPDrawingGroup (1035)

Contains Office Drawing information. See the Office Drawing Format Specification for more information.

PresAdvisoryFlags9 (6010)

An atom indicating disabled Presentation Advisory rules. It contains:

PrintOptions Fields

Offset	Type	Name	Contents
0	uint4	flags	Bitfield indicating which rule is disabled. If a bit is set, the corresponding rule is disabled. If the bit is cleared, it is enabled.

PrintOptions (6000)

PrintOptions is a record that stores default settings for how the PowerPoint presentation should be printed.

PrintOptions Fields

Offset	Type	Name	Contents
0	ubyte1	PrintWhat	What to print by default when printing the presentation. Valid values are from 0-6. See PrintWhat field values table below for valid values.
1	ubyte1	ColorMode	Default color mode to use when printing the presentation. Valid values are from 0-2. See ColorMode field values table below for valid values.
2	bool1	PrintHidden	True if hidden slides should be printed by default when printing the presentation.
3	bool1	ScaleToFitPaper	True if presentation should be scaled to fit paper when printing, by default.
4	bool1	FrameSlides	True if slides should be framed by default when printing the presentation.

PrintWhat Field Values

Value	Meaning
0	Slides (without animations, if any are present)
1	Slides (with animations, if any are present)
2	Handouts (2 slides per page)
3	Handouts (3 slides per page)
4	Handouts (6 slides per page)
5	Notes pages
6	Outline view
7	Handouts (4 slides per page)
8	Handouts (9 slides per page)
9	Handouts (1 slides per page)

ColorMode Field Values

Value	Meaning
0	Black and white
1	Gray-scale
2	Color

ProgBinaryTag (5002)

A name/value pair within a given Programmable Tag. It contains:

1. CString (4026), Instance TagName (0)
2. BinaryTagData (5003)

ProgStringTag (5001)

A name/value pair within a given Programmable Tag. It contains:

1. CString (4026), Instance TagName (0)
2. CString (4026), Instance TagValue (1)

ProgTags (5000)

A container for Programmable Tags. Prgrammable Tags are Name/Value pairs that contain data to be interpreted by various clients, e.g. they contain version specific extension to the binary file format for Document, Slide and Shape data. It contains:

1. ProgBinaryTag (5002), optional
2. ProgStringTag (5001), optional

RecolorInfoAtom (4071)

This atom keeps the recoloring information used to recolor pictures. It contains

RecolorInfoAtom fields

Offset	Type	Name	Content
0	uint2	flags	Bit 1: Should recoloring by applied Bit 2: Were there too many colors Bit 3: Were there too many fills Bit 4: Were any colors ignored Bit 5: Monochrome recolor Bit 6: Cannot modify recolor entries
2	sint2	nColors	Number of color entries
4	sint2	nFills	Number of fill entries
6	uint2[3]	monoColor	RGB color used for monochrome recoloring: Index 0: Red Index 1: Green Index 2: Blue
12	ColorEntry[]	entries	Variable number of color entries (see below). The actual number of entries is determined by nColors + nFills

ColorEntry fields

Offset	Type	Name	Content
0	sint2	doRecolor	F
2	uint2[3]	toColor	RGB color to recolor to
8	ubyte1	toIndex	Scheme index to recolor to
9	ubyte1	(unused)	
10	uint2	recolorType	Always 0
12	uint2[3]	color	RGB color to recolor from

RoundTripContentMasterId12 (1058)

Added in PowerPoint 2007.

A slide-level fixed-length record. It consists of main master id & content master instance id. This is used to round-trip the slide-master relation for slides following content masters in PowerPoint 2007. This is present only for slides that followed a content master in PowerPoint 2007 which is stored with the main master and thus the slide now follows that main master in the binary format. This is needed since multiple content masters can be stored with the main master during conversion.

RoundTripContentMasterId12 Fields

Offset	Type	Name	Contents
0	uint4	mainMasterId	Round-trip id of the main master
4	uint2	contentMasterInstanceId	Instance id of the content master (unique for main master)

RoundTripContentMasterInfo12 (1054)

Added in PowerPoint 2007.

A slide-level variable length record that contains a slide layout content master persisted in xml format. The purpose of this record is to round-trip placeholder information (shape properties, text styles, etc.). This record can be found in on composite masters (only one such record per composite master is allowed) or on main masters (multiple instances of this record are allowed).

The data is actually a package in Office Open XML format, which can simply be opened as a zip file. The zip file contains an xml file with the root element “sldLayout.”

For more information about the xml data representing a content masters, refer to the Office Open XML PresentationML documentation.

RoundTripCustomTableStyles12 (1064)

Added in PowerPoint 2007.

A variable length container which contains the Table Styles used by tables in the presentation. The purpose of this record is that when we open the file back in PowerPoint 2007 we can correctly restore the Table Styles for styling/rendering of the tables in PowerPoint 2007.

The data is actually a package in PowerPoint 2007 open xml format, which can be simply opened as a zip file. The main part in the package contains the table styles, starting with a <tblStyle> element that conforms to the schema defined by CT_TableStyleList.

For more info about the xml data representing table styles, refer to the Office Open XML DrawingML documentation.

RoundTripHFPlaceholder12 (1056)

Added in PowerPoint 2007.

A shape-level fixed-length record with single ubyte1. The presence of the record means that the shape is a header/footer placeholder. The byte stores the original placeholder id. This is needed because we don’t support slide/notes headers/footer placeholder shapes in PowerPoint 2003, but we do in PowerPoint 2007.

RoundTripHFPlaceholder12 Fields

Offset	Type	Name	Contents
0	ubyte1	placeholderId	Original placeholder id

RoundTripNotesMasterTextStyles12 (1063)

Added in PowerPoint 2007.

A variable length container which contains the Notes Master's OfficeArt text style. The purpose of this record is that when we open the file back in PowerPoint 2007 we can correctly restore the text styles in the Notes master.

The data is actually a package in Office Open XMLformat, which can be simply opened as a zip file. Data with root element "txstyles" is stored in this package.

For more information about the xml data representing slide master text styles, refer to the Office Open XML PresentationML documentation.

RoundTripOArtTextStyles12 (1059)

Added in PowerPoint 2007.

A variable length container which contains the Main Master's OfficeArt text style. The purpose of this record is that when we open the file back in PowerPoint 2007 we can correctly restore the text styles in the main master, which is important for rendering text on each slide following this master.

The data is actually a package in Office Open XML format, which can be simply opened as a zip file. The zip file contains an xml file with the root element "txstyles", used to save the OfficeArt text styles in the main master.

For more information about the xml data representing slide master text styles, refer to the Office Open XML (Ecma Internation Standard 376) PresentationML documentation.

RoundTripShapeCheckSumForCustomLayouts12 (1062)

Added in PowerPoint 2007.

This shape-level record consists of two dwords used for text & shape check-sums. During conversion to binary format some main master shapes are duplicated. When the shapes from a PowerPoint 2007 document are converted to PowerPoint 2003 shapes, the code runs check-sums on those and adds this record. When the document is loaded back in PowerPoint 2007, the code calculates the check-sums again and compares with the round-trip ones from this record to determine whether the shape has been modified by the user in PowerPoint 2003 or earlier versions.

RoundTripShapeCheckSumForCustomLayouts12 Fields

Offset	Type	Name	Contents
0	unsigned	shapeCheckSum	Checksum for the shape properties
4	unsigned	textCheckSum	Checksum for the shape text

RoundTripShapeId12 (1055)

Added in PowerPoint 2007.

A shape-level fixed-length record with single uint4 representing the shape id. This record is designed to preserve the relationship between shapes in two cases:

1. Slide placeholders and their master placeholders.
2. Original main master shapes & their duplicates on the composite master

In the first case, the id stored in the slide placeholder is the drawing element id of the master placeholder. In the second case, both the main master shape and its duplicates store a shape id, and the ids are the same.

RoundTripShapeId12Fields

Offset	Type	Name	Contents
0	uint4	shapeId	Shape id

RTFDateTimeMCAtom (4117)

RTFDateTimeMCAtom is a 64 uint2 long string for storing a Word-type field string that describes a date or time. For more information about Word's field strings, consult the Word Technical Reference. The field string is parsed and turned into a metacharacter like SlideNumberMCAtom.

RTFDateTimeMCAtom fields

Offset	Type	Name	Content
0	sint4	position	The position of the character in a text.
4	uint2[64]	format	The field string

Slide (1006)

Slide is a container that marks the beginning of a PowerPoint slide. The psrReference field of a SlidePersistAtom (1011) points to this Slide container. A Slide container can also be part of a ProgTags (5000) container. It contains:

1. SlideAtom (1007)
2. SSSlideInfoAtom (1017), optional
3. HeadersFooters (4057), optional
4. PPDrawing (1036)
5. ColorSchemeAtom (1013), Instance SlideScheme (1)
6. CString (4026), Instance SlideName (3), optional
7. ProgTags (5000), optional
8. Comment10 (12000), optional
9. LinkedSlideAtom10 (12007), optional
10. LinkedShapeAtom10 (12006), optional
11. SlideFlags10 (12010), optional
12. SlideTimeAtom10 (12011), optional
13. HashCodeAtom (11008), optional
14. BuildList (11010), optional
15. RoundTripContentMasterId12 (1058), optional
16. AnimationAtom12 (11019), optional
17. AnimationHashAtom12 (11021), optional
18. SlideSyncInfo12 (14100), optional

SlideAtom: (1007)

This atom stores the slide id and the slide master id.

SlideAtom Fields

Offset	Type	Name	Contents
0	SSlideLayout Atom (1015)	layout	Slide layout descriptor
12	sint4	masterId	This number identifies the master of the slide. It's null for a master slide
16	sint4	notesId	id referencing the corresponding notes slide. 0 if slide has no notes slide.
20	uint2	Flags	Bit 1: Follow master objects Bit 2: Follow master scheme Bit 3: Follow master background

SlideFlags10 (12010)

An atom containing additional flags for Slides. It contains:

LinkedShapeAtomAtom10 Fields

Offset	Type	Name	Contents
0	uint4	flags	Bit 1: Slide is preserved Bit 2: Slide follows master animations

SlideListEntryAtom10 (12016)

SlideListEntryAtom10 Fields

Offset	Type	Name	Contents
0	uint4	id	Slide ID
4	uint4	dwHighDateTime	Slide timestamp. See SlideTimeTimAtom10 (12011)
8	uint4	dwLowDateTime	Slide timestamp. See SlideTimeTimAtom10 (12011)

SlideListTable10 (12017)

A container for information about collaboration data. It contains:

1. SlideListTableSize (12015)
2. SlideLIstEntryAtom10 (12016), multiple, if any

SlideListTableSize (12015)

An atom containing information about a SlideListTable10 (12017). It contains:

SlideListTableSize Fields

Offset	Type	Name	Contents
0	sint4	size	Number of SlideListEntryAtom10's (12016) following

SlideListWithText (4080)

A container that contains the title and body texts a collection of Slides in the presentation. Instance specifies which collection. The actual placeholder shapes in the slides can then use a PST_OutlineTextRefAtom to reference these texts instead of containing them again.

SlideListWithText Instances

0	Collection of Slides
1	Collection of Master Slides
2	Collection of Notes Slides

It contains:

1. SlidePersistAtom (1011)
2. TextHeaderAtom (3999)

SlideNumberMCAtom (4056)

SlideNumberMCAtom is an atom that stores the position of the slide number meta character in the text. The slide number meta character is replaced by the current slide number during PowerPoint's runtime.

SlideNumberMCAtom fields

Offset	Type	Name	Content
0	sint4	position	The position of the character in a text.

SlidePersistAtom (1011)

SlidePersistAtom contains the information for the slide stub objects in the slide lists. The real slide data is stored in a different persist object which can be loaded\saved incrementally. The document saves all SlidePersistObjects in its persist stream so if you launch the number of slides and it's titles are available without loading all the slides.

SlidePersistAtom fields

Offset	Type	Name	Contents
0	uint4	psrReference	logical reference to the slide persist object
4	uint4	flags	Bit 2: Slide outline view is collapsed Bit 2: Slide contains shapes other than placeholders
8	sint4	numberTexts	number of placeholder texts stored with the persist object. Allows to display outline view without loading the slide persist objects
12	sint4	slideId	Unique slide identifier, used for OLE link monikers for example
16	uint4	Reserved	Unused field, always 0

SlideSyncInfo12 (14100)

A container for information about a slide that synchronizes to a slide on a slide library on a server. It contains:

1. CString (4026), Instance ServerID (0): Unique identifier for the slide in the slide library on the server
2. CString (4026), Instance SlideLibURL (1): URL of the slide library
3. SlideSyncInoAtom12 (14101)

SlideSyncInfoAtom12 (14101)

A slide-level fixed length record. Stores timestamps for slides that sync with a version of the slide stored on a server.

SlideSyncInfoAtom12 Fields

Offset	Type	Name	Contents
0	PSR_DateTimeAtom	dateTimeModified	Last modified time on server for this slide
16	PSR_DateTimeAtom	dateTimeInserted	Time the slide was inserted in the presentation

SlideTimeAtom10 (12011)

An atom containing a unique timestamp when the Slide was created. It contains:

LinkedShapeAtomAtom10 Fields

Offset	Type	Name	Contents
0	uint4	dwHighDateTime	Corresponds to the dwHighDateTime field of a System FILETIME structure
0	uint4	dwLowDateTime	Corresponds to the dwLowDateTime field of a System FILETIME structure

SlideViewInfo (1018)

A container that keeps the state of the grid, guide, and view scale. It's instances are:
SlideViewInfo (0): Slide view info for any slide that is not a Notes slide
NotesViewInfo (1): Slide view info for a Notes Slide

It's contents are:
1. SlideViewInfoAtom (1022)
2. ViewInfoAtom (1021), optional
3. GuideAtom (1019), optional

SlideViewInfoAtom (1022)

This atom keeps information about the guides and the grid, its fields are:

SlideViewInfoAtom Fields

Offset	Type	Name	Contents
0	bool1	showGuides	Set if the guides are visible.
1	bool1	snapToGrid	Set if snap to grid is on.
2	bool1	SnapToShape	Set if snap to shape is on.

SmartTagStore11 (14003)

This atom contains property information about the smart tags in the document. It consists of a uint4 which indicates the number of smart tag properties which have been written out, followed by the data for the smart tag properties. The format for this data is defined by the smart tags shared team and should be documented separately by them.

SorterViewInfo (1032)

This container keeps information about the view settings for Slide Sorter view. It contains:

1. ViewInfoAtom (1021)

Sound (2022)

A container holding information about a sound. It contains::

1. CString (4026), Instance 0: Name of sound (e.g. “crash”)
2. CString (4026), Instance 1: Type of sound (e.g. “.wav”)
3. CString (4026), Instance 2: Reference id of sound in sound collection
4. CString (4026), Instance 3, optional: Built-in id of sound, for sounds we ship. This is the id that’s in the reg file.
5. SoundData (2023), optional

SoundCollAtom (2021)

Contains the next unique identifier for external objects

SoundCollAtom Fields

Offset	Type	Name	Contents
0	sint4	objectIdSeed	Next unique identifier for external objects

SoundCollection (2020) & Instance Sounds (5)

Is a container for all sound related atoms and containers. It contains:

1. SoundCollAtom (2021)
2. Sound (2022), for each sound, if any

SoundData (2023)

Variable number of bytes. This is the sound file. It’s contents is dependent on the type of the sound.

SrKinsoku (4040)

A container for the Japanese word wrap feature. It contains:

1. SrKinsokuAtom (4050)
2. CString (4026), Instance Leading (0) that keeps the punctuation that is forbidden at the end of the line.
3. CString (4026), Instance Following (1) that keeps the punctuation that is forbidden at the beginning of the line.

SrKinsokuAtom (4050)

Atom that keeps in a four-byte signed integer the Kinsoku level that is displayed on the Kinsoku dialog. The level can be:

SrKinsokuAtom Fields

Offset	Type	Name	Contents
0	sint4	level	0: Level 1 1: Level 2 2: Custom level

SSDocInfoAtom (1025)

Atom that keeps Slide Show settings for the whole presentation.

SSDocInfoAtom Fields

Offset	Type	Name	Contents
0	GrColorAtom	penColor	Color for the on screen notation pen
4	sint4	restartTime	Time for auto restart of slide show in kiosk mode in millisec.
8	sint2	startSlide	First slide in slideshow
10	sint2	endSlide	Last slide in slideshow
12	uint2[32]	namedShow	Named show identifier
76	uint2	flags	Bit 1: Auto advance Bit 2: Skip builds Bit3: Use slide range Bit 4: Use named show Bit 5: Browse mode on Bit 6: Kiosk mode on Bit 7: Skip narration Bit 8: loop continously Bit 9: show scrollbar

SSlideLayoutAtom (1015)

Stores the slide's geometric layout, and the placeholders' ID.

SSlideLayoutAtom Fields

Offset	Type	Name	Contents
0	sint4	geom	Stores the geometric layout of the slide, this value can go from 0 to 18, and it identifies the position and number of placeholders. *See the Slide Layout table on the next page.*
4	ubyte1[8]	placeholderId	This field has an ID that identifies each of the placeholders on the slide. To see the meaning of each slide ID, refer to the PlaceholderID Values table under the OEPlaceholderAtom entry.

Slide Layout Table

Value	Meaning
0	The slide is a title slide
1	Title and body slide
2	Title master slide
3	(Not used)
4	Master notes layout
5	Notes title/body layout
6	Handout layout, therefore it doesn't have placeholders except header, footer, and date
7	Only title placeholder
8	Body of the slide has 2 columns and a title
9	Slide's body has 2 rows and a title
10	Body contains 2 columns, right column has 2 rows
11	Body contains 2 columns, left column has 2 rows
12	Body contains 2 rows, bottom row has 2 columns
13	Body contains 2 rows, top row has 2 columns
14	4 objects
15	Big object
16	Blank slide
17	Vertical title on the right, body on the left
18	Vertical title on the right, body on the left split into 2 rows

SSSlideInfoAtom (1017)

This atom keeps the information for the slide's transitions. The TransType field and the direction field together define a build effect.

SSSlideInfoAtom Fields

Offset	Type	Name	Contents
0	sint4	slideTime	How long to show the slide in ticks
4	uint4	soundRef	Index to a sound in the soundCollection
8	uint2	effect	High order byte: Type of transition. *See the Transition Type table below.* Low order byte: Direction of the transition. *See Direction table below*
10	uint2	flags	Flags that determine the type of build. *See Build Flags table below*
12	ubyte1	speed	Speed of the transition *See Transition Speed table below*

Transition Types

Flag	Meaning
0	No transition
1	Random
2	Blinds
3	Checker
4	Cover
5	Dissolve
6	Fade
7	Pull
8	Random bars
9	Strips
10	Wipe
11	Zoom
13	Split

Direction Values for Transitions

Type of transition	Value for direction	Meaning
Random & Dissolve	0	Anywhere
Wipes & Covers	0	Left
	1	Up
	2	Right
	3	Down
	4	Left up
	5	Right up
	6	Left down
	7	Right down
Strips	0	Up left
	1	Up right
	2	Down left
	3	Down right
	4	Right up
	5	Left down
	6	Right down
Zoom	0	Out
	1	In
Blinds & Stripes	0	Horizontal
	1	Vertical
Cuts	0	No black
	1	To black
	2	Best cut
Splits	0	Horizontal
	1	out

	2	Horizontal in
	3	Vertical out
		Vertical in
Flash	0	Fast
	1	Medium
	2	Slow

BuildFlags Field Values

Value	Meaning
0	Advance on mouse click
2	Hidden slide
4	The slide has sound
6	Loop until next sound
8	Stop the previous sound
10	Auto advance
12	Cursor always visible

Transition Speed Values

Value	Meaning
0	Slow (1000ms)
1	Medium (750ms)
2	Fast (500ms)
3	Very Fast (300ms)
4	Fastest (75ms)
5	Very Slow (5000ms)

StyleTextPropAtom (4001)

The paragraph and character properties for this text. This atom is of variable length and is organized in two run lists specifying exceptions from the style.

StyleTextPropAtom Fields

Type	Contents
uint4	Length of paragraph formatting run.
PF Run	Paragraph formatting run. *See Paragraph Formatting Run Fields below.*
Repeat until runs have been emitted for the entire text. Then,	
uint4	Length of character formatting run.
CF Run	Character formatting run. *See Character Formatting Run Fields below.*
Repeat until runs have been emitted for the entire text.	

Paragraph Formatting Run Fields

When?	Type	Contents
Always	uint2	Indent level of this run.
	uint4	Paragraph formatting mask of this run; the fields indicated appear immediately following.

		Bit 0: buHasBullet Bit 1: buHasTypeface Bit 2: buHasColor Bit 3: buHasSize Bit 4: buTypeface Bit 5: buSize Bit 6: buColor Bit 7: buChar Bit 8: pfLeftMargin Bit 9: Unused, must be zero. Bit 10: pfIndent Bit 11: pfAlignment Bit 12: pfLineSpacing Bit 13: pfSpaceBefore Bit 14: pfSpaceAfter Bit 15: pfDefaultTabSize Bit 16: pfBaseLine Bit 17: pfCharWrap Bit 18: pfWordWrap Bit 19: pfOverflow Bit 20: pfTabStops Bit 21: pfTextDirection Bits 22-31: Unused, must be zero.
buHasBullet buHasTypeface buHasColor buHasSize	uint2	Bullet flags. The entire field appears if any flag is an exception from the master, but only the value of the exceptional bits is considered. All other bits should be zero. For the buHas flags, the bullet follows the first character of the paragraph if it does not have a defined style of its own. Bit 0: buHasBullet – Is a bullet present? Bit 1: buHasTypeface – Does the bullet have a defined typeface? Bit 2: buHasColor – Does the bullet have a defined color? Bit 3: buHasSize – Does the bullet have a defined size? Bits 4-7: Unused, must be zero.
buChar	uint2	Unicode character of the bullet. Unicode PUA from U+F000-U+F0FF is used for characters from symbol fonts.
buTypeface	uint2	Index into font list (see PST_FontCollection) of bullet typeface. Only valid if buHasTypeface is set.
buSize	sint2	Size of bullet. If buSize >= 0, buSize is a percentage of the size of the first character of the paragraph. If buSize < 0, the absolute value of buSize is the point size of

		the bullet. Only valid if buHasSize is set.
buColor	GrColorAtom	Color of bullet. Only valid if buHasColor is set.
pfAlignment	sint2	Paragraph alignment. *See Paragraph Alignment table below.*
pfLineSpacing	sint2	Spacing between lines. If pfLineSpacing >= 0, pfLineSpacing is a percentage of normal line height. If pfLineSpacing < 0, the absolute value of pfLineSpacing is the spacing in master coordinates.
pfSpaceBefore	sint2	Spacing before a paragraph. If pfSpaceBefore >= 0, pfSpaceBefore is a percentage of normal line height. If pfSpaceBefore < 0, the absolute value of pfSpaceBefore is the spacing in master coordinates.
pfSpaceAfter	sint2	Spacing after a paragraph. If pfSpaceAfter >= 0, pfSpaceAfter is a percentage of normal line height. If pfSpaceAfter < 0, the absolute value of pfSpaceAfter is the spacing in master coordinates.
pfLeftMargin	sint2	Paragraph's distance from shape's left margin, in master coordinates.
pfIndent	sint2	First line of paragraph's distance from shape's left margin, in master coordinates.
pfDefaultTabSize	sint2	Default distance between tab stops, in master coordinates.
pfTabStops	Tab Stops	Location of tab stops. *See Tab Stops table below.*
pfBaseLine	uint2	Font alignment. *See Font Alignment table below.*
pfCharWrap pfWordWrap pfOverflow	uint2	East Asian line break flags. The entire field appears if any flag is an exception from the master, but only the value of the exceptional bits is considered. All other bits should be zero. Bit 0: pfCharWrap – Does the paragraph use Asian rules for controlling first and last characters? Bit 1: pfWordWrap – Does the paragraph allow Latin text to wrap in the middle of a word? Bit 2: pfOverflow – Does the paragraph allow hanging punctuation? Bits 3-7: Unused, must be zero.
pfTextDirection	uint2	Text direction. *See Text Direction table below.*

Character Formatting Run Fields

When?	Type	Contents

Always	uint4	Character formatting mask of this run; the fields indicated appear immediately following. Bits 0-15: cfStyle Bit 16: cfTypeface Bit 17: cfSize Bit 18: cfColor Bit 19: cfPosition Bit 20: Unused, must be zero. Bit 21: cfFEOldTypeface Bit 22: cfANSITypeface Bit 23: cfSymbolTypeface Bits 24-31: Unused, must be zero.
cfStyle	uint2	Style flags. The entire field appears if any flag is an exception from the master, but only the value of the exceptional bits is considered. All other bits should be zero. Bit 0: Is the text bold? Bit 1: Is the text italic? Bit 2: Is the text underlined? Bit 3: Unused, must be zero. Bit 4: Does the text have a shadow? Bit 5: Should smart quotes be rendered using an East Asian font? Bit 6: Unused, must be zero. Bit 7: Should numbers in East Asian vertical text be rendered horizontally? Bit 8: Unused, must be zero. Bit 9: Is this text embossed? Bits 10-13: Extension nibble. This number is the index modulo 16 of the PowerPoint 2000 extended paragraph and character formats and special info for this text (see PST_StyleTextProp9Atom). Bits 14-15: Unused, must be zero.
cfTypeface	uint2	Index into font list (see PST_FontCollection) of ASCII typeface.
cfFEOldTypeface	uint2	Index into font list (see PST_FontCollection) of legacy East Asian/complex scripts typeface. Used for legacy Office support only, not used by PowerPoint 2003.
cfANSITyepface	uint2	Index into font list (see PST_FontCollection) of ANSI typeface.
cfSymbolTypeface	uint2	Index into font list (see PST_FontCollection) of symbol typeface.
cfSize	uint2	Size of font, in points.
cfPosition	uint2	Offset from baseline, as a percentage of font size. If cfPosition > 0, the text is

		superscripted by the percentage given. If cfPosition < 0, the text is subscripted by the absolute value of the percentage given. If cfPosition = 0, the text is positioned on the baseline. Any text which is super/subscripted is reduced to 70% of its normal size.
cfColor	GrColorAtom	Color of text.

Tab Stops Fields

Type	Contents
uint2	Number of tab stops.
Then, for each tab stop,	
uint2	Distance of tab from shape's left margin in master coordinates.
uint2	Tab alignment. *See Tab Alignment table below.*

Paragraph Alignment

Flag	Meaning
0	Left
1	Center
2	Right
3	Justify
4	Distributed
5	Thai Distributed
6	Justify Low

Font Alignment

Flag	Meaning
0	Roman
1	Hanging
2	Centered
3	Upholding Fixed

Text Direction

Flag	Meaning
0	Left to Right
1	Right to Left

Tab Alignment

Flag	Meaning
0	Left
1	Center
2	Right
3	Decimal

StyleTextProp9Atom (4012)

The PowerPoint 2000 extended paragraph and character properties and special info for this text. This atom is of variable length and is organized as runs specifying exceptions from the

style. Each set of three runs is applied to the text according to the extension nibble set in cfStyle in the character properties in the PST_StyleTextPropAtom. Runs are applied beginning from the start of the text and are only applied in order, so as to permit the use of modulo 16 indices. If no text has an index corresponding to a specific set of extension runs, they are discarded.

StyleTextProp9Atom Fields

Type	Contents
PF2000 Run	PowerPoint 2000 extended paragraph formatting run. *See PowerPoint 2000 Extended Paragraph Formatting Run Fields table below.*
CF2000 Run	PowerPoint 2000 extended character formatting run. *See PowerPoint 2000 Extended Character Formatting Run Fields table below.*
SI2000 Run	PowerPoint 2000 extended special info run. *See PowerPoint 2000 Extended Special Info Run Fields table below.*
Repeat until runs have been emitted for the entire text.	

PowerPoint 2000 Extended Paragraph Formatting Run Fields

When?	Type	Contents
Always	uint4	PowerPoint 2000 extended paragraph formatting mask of this run; the fields indicated appear immediately following. Bits 0-22: Unused, must be zero. Bit 23: buBlip Bit 24: buAnmScheme Bit 25: buHasAnm Bit 26: pfPP10Ext Bits 27-31: Unused, must be zero.
buBlip	uint2	Index into picure bullet list (see PST_BlipCollection) of picture bullet. -1 is used to represent no bullet.
buHasAnm	uint2	Autonumbering scheme flags. Bit 0: Does this paragraph have an autonumbered bullet? Bits 1-15: Unused, must be zero.
buAnmScheme	TxAnmListProps	Autonumbering scheme.
pfPP10Ext	uint4	Unused, must be zero.

PowerPoint 2000 Extended Character Formatting Run Fields

When?	Type	Contents
Always	uint4	PowerPoint 2000 extended character formatting mask of this run; the fields indicated appear immediately following. Bits 0-19: Unused, must be zero. Bit 20: cfPP10Ext Bits 21-31: Unused, must be zero.
cfPP10Ext	uint4	Bits 0-3: Extension nibble. This number is the index modulo 16 of the PowerPoint 2002 extended character formats for this

		text (see PST_StyleTextProp10Atom). Bits 4-31: Unused, must be zero.

PowerPoint 2000 Extended Special Info Run Fields

When?	Type	Contents
Always	uint4	PowerPoint 2000 extended special info mask of this run; the fields indicated appear immediately following. Bits 0-4: Unused, must be zero. Bit 5: pp10Ext Bit 6: fBidi Bits 7-31: Unused, must be zero.
pp10Ext	uint4	Bits 0-3: Extension nibble. This number is the index modulo 16 of the PowerPoint 2003 extended special info for this text (see PST_StyleTextProp11Atom). Bits 4-30: Unused, must be zero. Bit 31: Does this text have incorrect Japanese grammar?
fBidi	uint2	Bit 0: Is this text to be laid out with a BiDi level greater than zero according to the Unicode algorithm? Bits 1-15: Unused, must be zero.

StyleTextProp10Atom (4017)

The PowerPoint 2002 extended character properties for this text. This atom is of variable length and is organized as runs specifying exceptions from the style. Each run is applied to the text according to the extension nibble set in cfPP10Ext in the PowerPoint 2000 extended character properties in the PST_StyleTextProp9Atom. Runs are applied beginning from the start of the text and are only applied in order, so as to permit the use of modulo 16 indices. If no text has an index corresponding to a specific extension run, it is discarded.

StyleTextProp10Atom Fields

Type	Contents
CF2002 Run	PowerPoint 2002 extended character formatting run. *See PowerPoint 2002 Extended Character Formatting Run Fields table below.*
Repeat until runs have been emitted for the entire text.	

PowerPoint 2002 Extended Character Formatting Run Fields

When?	Type	Contents
Always	uint4	PowerPoint 2002 extended character formatting mask of this run; the fields indicated appear immediately following. Bits 0-23: Unused, must be zero. Bit 24: cfFENewTypeface Bit 25: cfCSTypeface Bit 26: cfPP11Ext Bits 27-31: Unused, must be zero.
cfFENewTypeface	uint2	Index into PowerPoint 2002 extended font

		list (see PST_FontCollection10) of East Asian typeface.
cfCSTypeface	uint2	Index into PowerPoint 2002 extended font list (see PST_FontCollection10) of complex scripts typeface.
cfPP11Ext	uint4	Unused, must be zero.

StyleTextProp11Atom (4022)

The PowerPoint 2003 extended special info for this text. This atom is of variable length and is organized as runs specifying exceptions from the style. Each run is applied to the text according to the extension nibble set in pp10Ext in the PowerPoint 2000 extended special info in the PST_StyleTextProp9Atom. Runs are applied beginning from the start of the text and are only applied in order, so as to permit the use of modulo 16 indices. If no text has an index corresponding to a specific extension run, it is discarded.

StyleTextProp11Atom Fields

Type	Contents
SI2003 Run	PowerPoint 2003 extended special info run. *See PowerPoint 2003 Extended Special Info Run Fields table below.*
Repeat until runs have been emitted for the entire text.	

PowerPoint 2003 Extended Special Info Run Fields

When?	Type	Contents
Always	uint4	PowerPoint 2002 extended character formatting mask of this run; the fields indicated appear immediately following. Bits 0-8: Unused, must be zero. Bit 9: smartTags Bit 10: pp12Ext Bits 11-31: Unused, must be zero.
smartTags	SmartTags	Smart tags attached to this range of text. *See SmartTags Fields table below.*
pp12Ext	uint4	Unused, must be zero.

SmartTags Fields

Type	Contents
uint4	Number of smart tags on this text.
Then, for each smart tag,	
uint4	Index of this smart tag's data within the PST_SmartTagStore11.

Summary (1026)

A container for the presentation's summary information. It contains:

1. BookmarkCollection (2019)

Theme (1038)

Added in PowerPoint 2007.

A variable length container which contains the Main Master's Theme. The purpose of this record is so that when we open the file back in PowerPoint 2007 we can correctly restore the Theme for the main master.

The data is actually a package in Office Open XML format, which can be simply opened as a zip file. Data with root element "theme" and "themeOverride" may be stored in xml files inside this package.

For more information about the xml theme data, refer to the Office Open XML DrawingML documentation.

TextBookmarkAtom (4007)

Bookmark ("property") within text

TextBookmarkAtom Fields

Offset	Type	Name	Contents
0	uint4	begin	Beginning character position
4	uint4	end	End character position
8	uint4	bookmarkID	Bookmark ID

TextBytesAtom (4008)

The actual characters of the text, not including the trailing return character, stored as bytes. Each byte is the low byte of a character in the Unicode character set, with the high byte considered equal to zero. This atom is of variable length and depends on the length of the text.

TextCharsAtom (4000)

The actual characters of the text, not including the trailing return character, stored in the Unicode character set. Most Unicode characters are two bytes; some characters are 'surrogate' characters which take four bytes. Characters in symbol fonts are stored using the Unicode private use area U+F000-U+F0FF. Metacharacters are stored as asterisks. This atom is of variable length and depends on the length of the text.

TextDefaults9Atom (4016)

Used only within the PST_Environment container to store text default PowerPoint 2000 extended character and paragraph properties for new texts. This atom is of variable length, and consists of PowerPoint 2000 extended character and paragraph formatting runs. The masks for these runs indicate the differences between the defaults for new texts and the 'other' text style at indent level zero.

TextDefaults9Atom Fields

Type	Contents
CF2000 Run	PowerPoint 2000 extended character formatting run describing differences between defaults and the 'other' text style at indent level zero (see PST_StyleTextProp9Atom).
PF2000 Run	PowerPoint 2000 extended paragraph formatting run describing differences between defaults and the 'other' text style at indent level zero (see PST_StyleTextProp9Atom).

TextDefaults10Atom (4020)

Used only within the PST_Environment container to store text default PowerPoint 2002 extended character properties for new texts. This atom is of variable length, and consists of a PowerPoint 2002 extended character formatting run. The mask for this run indicates the differences between the defaults for new texts and the 'other' text style at indent level zero.

TextDefaults9Atom Fields

Type	Contents
CF2002 Run	PowerPoint 2002 extended character formatting run describing differences between defaults and the 'other' text style at indent level zero (see PST_StyleTextProp10Atom).

TextHeaderAtom (3999)

Appears in the beginning of a series of atoms belonging to the same text.

TextHeaderAtom Fields

Offset	Type	Name	Contents
0	uint4	txType	Type of text. *See the Text Type table below.*

Text Types

Flag	Meaning
0	Title
1	Body
2	Notes
3	Outline
4	Other (Text in a shape)
5	Center body (subtitle in title slide)
6	Center title (title in title slide)
7	Half body (body in two-column slide)
8	Quarter body (body in four-body slide)

TextRulerAtom (4006)

Ruler of a text as it differs from the style's ruler settings. This atom is of variable length.

TextRulerAtom Fields

When?	Type	Contents

Always	uint4	Ruler mask of this run; the fields indicated are exceptions from the master style and appear immediately following. Bit 0: defaultTabSize Bit 1: numLevels Bit 2: tabStops Bit 3: leftMargin0 Bit 4: leftMargin1 Bit 5: leftMargin2 Bit 6: leftMargin3 Bit 7: leftMargin4 Bit 8: indent0 Bit 9: indent1 Bit 10: indent2 Bit 11: indent3 Bit 12: indent4 Bits 13-31: Unused, must be zero.
numLevels	uint2	Number of indent levels (maximum 5).
defaultTabSize	uint2	Default distance between tab stops, in master coordinates.
tabStops	Tab Stops	Location of tab stops (see PST_StyleTextPropAtom).
leftMargin0	uint2	Paragraph's distance from shape's left margin, in master coordinates, at style level 0.
indent0	uint2	First line of paragraph's distance from shape's left margin, in master coordinates, at indent level 0.
leftMargin1	uint2	Paragraph's distance from shape's left margin, in master coordinates, at style level 1.
indent1	uint2	First line of paragraph's distance from shape's left margin, in master coordinates, at indent level 1.
leftMargin2	uint2	Paragraph's distance from shape's left margin, in master coordinates, at style level 2.
indent2	uint2	First line of paragraph's distance from shape's left margin, in master coordinates, at indent level 2.
leftMargin3	uint2	Paragraph's distance from shape's left margin, in master coordinates, at style level 3.
indent3	uint2	First line of paragraph's distance from shape's left margin, in master coordinates, at indent level 3.
leftMargin4	uint2	Paragraph's distance from shape's left margin, in master coordinates, at style level 4.

indent4	uint2	First line of paragraph's distance from shape's left margin, in master coordinates, at indent level 4.

TextSpecInfoAtom (4010)

The special info runs contained in this text. Special info runs consist of character properties which don't follow styles. This atom is of variable length.

TextSpecInfoAtom Fields

Type	Contents
uint4	Length of special info run.
SI Run	Special info run. *See Special Info Run Fields table below.*
Repeat until runs have been emitted for the entire text.	

Special Info Run Fields

When?	Type	Contents
Always	uint4	Special info mask of this run; the fields indicated appear immediately following. Bit 0: spellInfo Bit 1: langId (always set with bit 2) Bit 2: altLangId (always set with bit 1) Bits 3-31: Unused, must be zero.
spellInfo	uint2	Spelling status of this text. *See Spell Info table below.*
langId	uint2	Windows LANGID for this text.
altLangId	uint2	Alternate Windows LANGID of this text; must be a valid non-East Asian LANGID if the text has an East Asian language, otherwise may be an East Asian LANGID or language neutral (zero).

Spell Info Types

Flag	Meaning
0	Unchecked
1	Previously incorrect, needs rechecking
2	Correct
3	Incorrect

TxCFExceptionAtom (4004)

Used only within the PST_Environment container to store text default character properties for new texts. This atom is of variable length, and consists of a character formatting run. The mask for this run indicates the differences between the defaults for new texts and the 'other' text style at indent level zero.

TxCFExceptionAtom Fields

Type	Contents
CF Run	Character formatting run describing differences between defaults and the 'other' text style at indent level zero (see

PST_StyleTextPropAtom).

TxInteractiveInfoAtom (4063)

An interactive info in a text. Instance specifies the type of the interactive event: MouseClick (0) or MouseOver (1). These atoms always follow a corresponding InteractiveInfo (4082) atom containing the actual interactive info data.

Offset	Type	Name	Contents
0	uint4	begin	Beginning character position
4	uint4	end	Ending character position

TxMasterStyleAtom (4003)

PowerPoint text styles. The atom instance value is the text type and corresponds to the *TxStyle* field in the PST_TextHeaderAtom. The text styles are located in the PST_MainMaster container, except for the “other” style, which is in the PST_Environment container. This atom is of variable length, and consists of a character and a paragraph formatting run for each indent level defined in a style. If this style is a derived style, the masks contain only such bits as differ from the base style this style is derived from. The center title style is derived from the title style, and the center, half, and quarter body styles are derived from the body style. All other styles are base styles and have all defined bits set in the mask; they contain a complete description of the formatting.

TxMasterStyleAtom Fields

Type	Contents
uint2	Number of indent levels in this style (maximum 5).
Then, for each indent level:	
PF Run	Paragraph formatting run (see PST_StyleTextPropAtom).
CF Run	Character formatting run (see PST_StyleTextPropAtom).

TxMasterStyle9Atom (4013)

PowerPoint 2000 extended text styles. The atom instance value is the text type and corresponds to the *TxStyle* field in the PST_TextHeaderAtom. The text styles are located in the PST_MainMaster container, except for the “other” style, which is in the PST_Environment container. This atom is of variable length, and consists of a PowerPoint 2000 extended character and paragraph formatting run for each indent level defined in a style. For the purposes of the extended text styles no style is considered to be derived from another style. Mask bits which are unset point to a property or feature which is not set.

TxMasterStyleAtom Fields

Type	Contents
uint2	Number of indent levels in this style (maximum 5).
Then, for each indent level:	
PF2000 Run	Paragraph formatting run (see PST_StyleTextProp9Atom).
CF2000 Run	Character formatting run (see PST_StyleTextProp9Atom).

TxMasterStyle10Atom (4018)

PowerPoint 2002 extended text styles. The atom instance value is the text type and corresponds to the *TxStyle* field in the PST_TextHeaderAtom. The text styles are located in the PST_MainMaster container, except for the “other” style, which is in the PST_Environment container. This atom is of variable length, and consists of a PowerPoint 2002 extended character formatting run for each indent level defined in a style. For the purposes of the extended text styles no style is considered to be derived from another style. Mask bits which are unset point to a property or feature which is not set.

TxMasterStyleAtom Fields

Type	Contents
uint2	Number of indent levels in this style (maximum 5).
Then, for each indent level:	
CF2002 Run	PowerPoint 2002 extended character formatting run (see PST_StyleTextProp10Atom).

TxPFExceptionAtom (4005)

Used only within the PST_Environment container to store text default paragraph properties for new texts. This atom is of variable length, and consists of a paragraph formatting run. The mask for this run indicates the differences between the defaults for new texts and the ‘other’ text style at indent level zero. The indent level for this run must be written out as zero.

TxPFExceptionAtom Fields

Type	Contents
PF Run	Paragraph formatting run describing differences between defaults and the ‘other’ text style at indent level zero (see PST_StyleTextPropAtom).

TxSpecialInfoAtom (4009)

Used only within the PST_Environment container to store default special info (see PST_TextSpecInfoAtom for a definition of “special info”) for new texts. This atom is of variable length, and consists of a special info run with all defined bits set.

TxSpecialInfoAtom Fields

Type	Contents
SI Run	Special info run describing default special info (see PST_TextSpecInfoAtom).

UserEditAtom (4085)

See UserEditAtom in “Current User Stream” section.

Offset	Type	Name	Contents
0	sint4	lastSlideID	Id of slide currently selected in view
4	uint4	version	Major and minor app version that did the save
8	uint4	offsetLastEdit	File offset of UsereditAtom of the previous incremental save. 0 after a full save
12	uint4	offsetPersistDirectory	File offset to persist pointers for this save operation

16	uint4	documentRef	Persist reference to the document persist object
20	uint4	maxPersistWritten	Seed value for persist object id management
24	sint2	lastViewType	View type see table below

Last View Field Values

Value	Meaning
0	None
1	Slide
2	Slide Master
3	Notes
4	Handout Page
5	Notes Master
6	Outline Master
7	Outline View
8	Sorter View
9	Visual Basic Editor
10	Title Master
11	SlideShow
12	SlideShow Fullscreen
13	Notes Text
14	Print Preview
15	Thumbnails
16	Master Thumbnails

VBAInfo (1023)

A container for VBA (Visual Basic for Applications) information. It contains:

1. VBAInfoAtom (1024)

VBAInfoAtom (1024)

Contains information about a VBA Storage

VBAInfoAtom Fields

Offset	Type	Name	Contents
0	uint4	objStgDataRef	Logical reference to the VBA persist object
4	uint4	hasMacros	0 if the VBA Storage is empty 1 if the VBA Storage contains data
8	uint4	version	VBAInfoAtom (2)

ViewInfoAtom (1021)

Contains information about the scale at which the slide is seen.

ViewInfoAtom Fields

Offset	Type	Name	Contents
0	PSR_GScalingAtom	curScale	Keeps the current scale
16	PSR_GScalingAtom	prevScale	Keeps the previous scale
32	PSR_GPointAtom	viewSize	Keeps the size of the view in master coordinates
40	PSR_GPointAtom	origin	Keeps the origin in master coordinates
48	bool1	varScale	Set if zoom to fit is set
49	bool1	draftMode	Not used

VisualPageAtom (11009)

An atom containing information about animation data for a Slide. It contains

VisualShapeAtom Fields

Offset	Type	Name	Contents
0	uint4	type	Type of Visual Element, see table below Always Slide (1)

VisualShapeAtom (11003)

An atom containing information about animation data for a Shape. It contains

VisualShapeAtom Fields

Offset	Type	Name	Contents
0	uint4	type	Type of the Visual Element, see table below
4	uint4	refType	Additional, PPT specific, type information 0: Uninitialized 1: Shape 2: Sound 3: Invalid Text Range
8	uint4	id	If refTpe = Sound (2) ID of the sound object else Shape ID
12	sint4	data0	If type = Chart Element (5) Chart Element type, see table below Else Text Range start position
16	sint4	data1	If type = Chart Element (5) level Else

	Text Range end position

Visual Element Type Values

0	Shape
1	Slide
2	Text Range
3	Audio
4	Video
5	Chart Element
6	Shape only (no text)
7	Uninitialized
8	All Text Range

Chart Element Type Values

0	None
1	Series
2	Category
3	Element in Series
4	Element in Category
5	Custom

Apendix A: Records Ordered by Number

Name	*Type*
Unknown	0
SubContainerCompleted	1
IRRAtom	2
PSS	3
SubContainerException	4
ClientSignal1	6
ClientSignal2	7
PowerPointStateInfoAtom	10
Document	1000
DocumentAtom	1001
EndDocument	1002
SlidePersist	1003
SlideBase	1004
SlideBaseAtom	1005
Slide	1006
SlideAtom	1007
Notes	1008
NotesAtom	1009
Environment	1010
SlidePersistAtom	1011
Scheme	1012
SchemeAtom	1013
DocViewInfo	1014
SslideLayoutAtom	1015
MainMaster	1016
SSSlideInfoAtom	1017
SlideViewInfo	1018
GuideAtom	1019
ViewInfo	1020
ViewInfoAtom	1021
SlideViewInfoAtom	1022
VBAInfo	1023
VBAInfoAtom	1024
SSDocInfoAtom	1025
Summary	1026
Texture	1027
VBASlideInfo	1028
VBASlideInfoAtom	1029
DocRoutingSlip	1030
OutlineViewInfo	1031
SorterViewInfo	1032
ExObjList	1033
ExObjListAtom	1034
PPDrawingGroup	1035

PPDrawing	1036
Theme	1038
ColorMapping	1039
NamedShows	1040
NamedShow	1041
NamedShowSlides	1042
OriginalMainMasterId	1052
CompositeMasterId	1053
RoundTripContentMasterInfo12	1054
RoundTripShapeId12	1055
RoundTripHFPlaceholder12	1056
RoundTripContentMasterId12	1058
RoundTripOArtTextStyles12	1059
HeaderFooterDefaults12	1060
DocFlags12	1061
RoundTripShapeCheckSumForCustomLayouts12	1062
RoundTripNotesMasterTextStyles12	1063
RoundTripCustomTableStyles12	1064
List	2000
FontCollection	2005
ListPlaceholder	2017
BookmarkCollection	2019
SoundCollection	2020
SoundCollAtom	2021
Sound	2022
SoundData	2023
BookmarkSeedAtom	2025
GuideList	2026
RunArray	2028
RunArrayAtom	2029
ArrayElementAtom	2030
Int4ArrayAtom	2031
ColorSchemeAtom	2032
OEShape	3008
ExObjRefAtom	3009
OEPlaceholderAtom	3011
GrColor	3020
GrectAtom	3025
GratioAtom	3031
Gscaling	3032
GpointAtom	3034
OEShapeAtom	3035
OEPlaceholderNewPlaceholderId12	3037
OutlineTextRefAtom	3998
TextHeaderAtom	3999
TextCharsAtom	4000
StyleTextPropAtom	4001
BaseTextPropAtom	4002
TxMasterStyleAtom	4003

TxCFStyleAtom	4004
TxPFStyleAtom	4005
TextRulerAtom	4006
TextBookmarkAtom	4007
TextBytesAtom	4008
TxSIStyleAtom	4009
TextSpecInfoAtom	4010
DefaultRulerAtom	4011
FontEntityAtom	4023
FontEmbedData	4024
TypeFace	4025
CString	4026
ExternalObject	4027
MetaFile	4033
ExOleObj	4034
ExOleObjAtom	4035
ExPlainLinkAtom	4036
CorePict	4037
CorePictAtom	4038
ExPlainAtom	4039
SrKinsoku	4040
Handout	4041
ExEmbed	4044
ExEmbedAtom	4045
ExLink	4046
ExLinkAtom_old	4047
BookmarkEntityAtom	4048
ExLinkAtom	4049
SrKinsokuAtom	4050
ExHyperlinkAtom	4051
ExPlain	4053
ExPlainLink	4054
ExHyperlink	4055
SlideNumberMCAtom	4056
HeadersFooters	4057
HeadersFootersAtom	4058
RecolorEntryAtom	4062
TxInteractiveInfoAtom	4063
EmFormatAtom	4065
CharFormatAtom	4066
ParaFormatAtom	4067
MasterText	4068
RecolorInfoAtom	4071
ExQuickTime	4073
ExQuickTimeMovie	4074
ExQuickTimeMovieData	4075
ExSubscription	4076
ExSubscriptionSection	4077
ExControl	4078

ExControlAtom	4091
SlideListWithText	4080
AnimationInfoAtom	4081
InteractiveInfo	4082
InteractiveInfoAtom	4083
SlideList	4084
UserEditAtom	4085
CurrentUserAtom	4086
DateTimeMCAtom	4087
GenericDateMCAtom	4088
HeaderMCAtom	4089
FooterMCAtom	4090
ExMediaAtom	4100
ExVideo	4101
ExAviMovie	4102
ExMCIMovie	4103
ExMIDIAudio	4109
ExCDAudio	4110
ExWAVAudioEmbedded	4111
ExWAVAudioLink	4112
ExOleObjStg	4113
ExCDAudioAtom	4114
ExWAVAudioEmbeddedAtom	4115
AnimationInfo	4116
RTFDateTimeMCAtom	4117
ProgTags	5000
ProgStringTag	5001
ProgBinaryTag	5002
BinaryTagData	5003
PrintOptions	6000
PersistPtrFullBlock	6001
PersistPtrIncrementalBlock	6002
RulerIndentAtom	10000
GscalingAtom	10001
GrColorAtom	10002
GLPointAtom	10003
GlineAtom	10004
AnimationAtom12	11019
AnimationHashAtom12	11021
SlideSyncInfo12	14100
SlideSyncInfoAtom12	14101

Appendix B: Miscellaneous Enumerated Types and Structures

```
#pragma pack(4)
//====================== Types enumeration
==================================
//=====================================================================
=======

enum psrTypeCode                        // enumerates record types that
are saved
{
   PST_UNKNOWN                    = 0,  // should never occur in file
   PST_SubContainerCompleted      = 1,  // should never occur in file
   PST_IRRAtom                    = 2,  // Indexed Record Reference
   PST_PSS                        = 3,  // start of stream
   PST_SubContainerException      = 4,  // should never occur in file
   PST_ClientSignal1 = 6,               // should never occur in file
   PST_ClientSignal2 = 7,               // should never occur in file


   /* Application Saved State Information */
   PST_PowerPointStateInfoAtom = 10,

   /* Document & Slide */
   PST_Document             = 1000,
   PST_DocumentAtom         = 1001,
   PST_EndDocument          = 1002,
   // unused 1003
   PST_SlideBase            = 1004,
   PST_SlideBaseAtom        = 1005,
   PST_Slide                = 1006,
   PST_SlideAtom            = 1007,
   PST_Notes                = 1008,
   PST_NotesAtom            = 1009,
   PST_Environment          = 1010,
//   PST_DLook                = 1011,
   PST_Scheme               = 1012,
   PST_SchemeAtom           = 1013,
   PST_DocViewInfo          = 1014,
   PST_SSlideLayoutAtom     = 1015,
   PST_MainMaster           = 1016,
   PST_SSSlideInfoAtom      = 1017,
   PST_SlideViewInfo        = 1018,
   PST_GuideAtom            = 1019,
   PST_ViewInfo             = 1020,
   PST_ViewInfoAtom         = 1021,
   PST_SlideViewInfoAtom    = 1022,
   PST_VBAInfo              = 1023,
   PST_VBAInfoAtom          = 1024,
   PST_SSDocInfoAtom        = 1025,
   PST_Summary              = 1026,
//   PST_Texture              = 1027,
   PST_VBASlideInfo         = 1028,
   PST_VBASlideInfoAtom     = 1029,
```

```
   PST_DocRoutingSlip      = 1030,
   PST_OutlineViewInfo     = 1031,
   PST_SorterViewInfo      = 1032,
   PST_ExObjList           = 1033,     /* new for PP96 */
   PST_ExObjListAtom       = 1034,     /* new for PP96 */
   PST_PPDrawingGroup      = 1035,     /* new for PP96 */
   PST_PPDrawing           = 1036,     /* new for PP96 */

   PST_NamedShows          = 1040,
   PST_NamedShow           = 1041,
   PST_NamedShowSlides     = 1042,

   /* Collections & lists */
   PST_List                = 2000,
   PST_FontCollection      = 2005,
   PST_ListPlaceholder     = 2017,
   PST_BookmarkCollection  = 2019,
   PST_SoundCollection     = 2020,
   PST_SoundCollAtom       = 2021,
   PST_Sound               = 2022,
   PST_SoundData           = 2023,
   PST_BookmarkSeedAtom    = 2025,
   PST_GuideList           = 2026,
   // ...
   PST_RunArray            = 2028,
   PST_RunArrayAtom        = 2029,     // for compatibility with pre-
d255 files
   PST_ArrayElementAtom    = 2030,     // variable length atom.  no
PSR_ defined
   PST_Int4ArrayAtom       = 2031,     // variable length atom.  no
PSR_ defined
   PST_ColorSchemeAtom     = 2032,     // contains 8 colors

   /* Slide Elements */
   PST_OEShape             = 3008,
   PST_ExObjRefAtom        = 3009,
   PST_OEPlaceholderAtom   = 3011,
   PST_GrColor             = 3020,
   PST_GRectAtom           = 3025,
   PST_GRatioAtom          = 3031,
   PST_GScaling            = 3032,
   PST_GPointAtom          = 3034,

   /* Text, Rulers, External */
   PST_TextCharsAtom       = 4000,
   PST_StyleTextPropAtom   = 4001,
   PST_BaseTextPropAtom    = 4002,
   PST_TxMasterStyleAtom   = 4003,
   PST_TxCFStyleAtom       = 4004,
   PST_TxPFStyleAtom       = 4005,
   // ...
   PST_FontEntityAtom      = 4023,
   PST_FontEmbedData       = 4024,
   PST_TypeFace            = 4025,
   PST_CString             = 4026,
   PST_ExternalObject      = 4027,
   PST_MetaFile            = 4033,
```

```
   PST_ExOleObj            = 4034,
   PST_ExOleObjAtom        = 4035,
   PST_ExPlainLinkAtom     = 4036,
   PST_CorePict            = 4037,
   PST_CorePictAtom        = 4038,
   PST_ExPlainAtom         = 4039,
   PST_SrKinsoku           = 4040,
   PST_Handout             = 4041,
   PST_ExEmbed             = 4044,
   PST_ExEmbedAtom         = 4045,
   PST_ExLink              = 4046,
   PST_ExLinkAtom          = 4047,
   PST_BookmarkEntityAtom  = 4048,
   PST_SrKinsokuAtom       = 4050,
   PST_ExHyperlinkAtom     = 4051,
   PST_ExPlain             = 4053,
   PST_ExPlainLink         = 4054,
   PST_ExHyperlink         = 4055,
   PST_SlideNumberMCAtom   = 4056,
   PST_HeadersFooters      = 4057,
   PST_HeadersFootersAtom  = 4058,
   PST_RecolorEntryAtom    = 4062,
   PST_PowerText           = 4064,
   PST_EmFormatAtom        = 4065,
   PST_CharFormatAtom      = 4066,
   PST_ParaFormatAtom      = 4067,
   PST_MasterText          = 4068,
   PST_RulerEntity         = 4069,
   PST_RulerTabArrayAtom   = 4070,        // variable length atom.  no
PSR_ defined
   PST_RecolorInfoAtom     = 4071,     // recolor info after d303
   PST_ExQuickTime         = 4073,
   PST_ExQuickTimeMovie    = 4074,
   PST_ExQuickTimeMovieData= 4075,
   PST_ExSubscription      = 4076,
   PST_ExSubscriptionSection= 4077,
   PST_ExControl           = 4078,     // new for PP96
   PST_ExControlAtom       = 4079,     // new for PP96
   PST_AnimationInfoAtom   = 4081,
   PST_InteractiveInfo     = 4082,
   PST_InteractiveInfoAtom = 4083,
   PST_SlideList           = 4084,
   PST_UserEditAtom        = 4085,
   PST_CurrentUserAtom     = 4086,
   PST_DateTimeMCAtom      = 4087,     // header and footer meta
characters.
   PST_GenericDateMCAtom   = 4088,
   PST_HeaderMCAtom        = 4089,
   PST_FooterMCAtom        = 4090,
   PST_ExMediaAtom         = 4100,    // External Media
   PST_ExVideo             = 4101,
   PST_ExAviMovie          = 4102,


   //  The 5000 block is used for client data in our
   PST_ProgTags            = 5000,        // Programmable tags
   PST_ProgTagsAtom        = 5001,
```

```
   PST_ProgTag             = 5002,


   PST_PrintOptions        = 6000,        // Per-document print options
   PST_PersistPtrFullBlock = 6001,              // Complete list of
persist's for this ver.
   PST_PersistPtrIncrementalBlock = 6002, // Incremental diffs on
persists

   PST_GScalingAtom        = 10001,       // Does not occur in file
(nested within another record)
   PST_GrColorAtom         = 10002,       // Does not occur in file
(nested within another record)


   PST_LAST
};



//====================== Instance enumeration
================================
//=========================================================================
=======

//
enum PSSInstanceCode
{
   // exoleobj
  INS_StgName           = 0,
  INS_MenuName          = 1,
  INS_ProgID            = 2,
  INS_ClipboardName     = 3,

   // SrKinsoku
  INS_Leading           = 0,
  INS_Following         = 1,
  INS_DocKinsoku        = 2,
  INS_SrKinsokuLevel    = 3,

   // VBAInfo
  INS_StorageName       = 1,
  INS_MacroName         = 2,
   // look
  INS_LookName          = 1,

   // object array
  INS_ObArrayElement    = 0,

   // doc
  INS_DocSlideList      = 0,
  INS_DocMasterList     = 1,

  INS_DocInfoList       = 0,
  INS_DocSlideShowInfo  = 0,
  INS_Handout           = 0,
  INS_Summary           = 0,
```

```
// slide
INS_SlideSlideShowInfo = 0,
INS_SlideScheme       = 1,      // SlideMaster's scheme
INS_TemplateName      = 2,

INS_GroupElementList  = 0,
INS_ListElement       = 1,
INS_OEInfoListElement = 2,
INS_SlideElementListElement = 3,
INS_OElements         = 4,
INS_InfoListElement   = 5,
INS_SchemeListElement = 6,
INS_GuideListElement  = 7,
INS_SlideBackground   = 8,

 // environment
INS_DocEnvironment    = 0,
INS_DefaultAttribs    = 1,
INS_Pictures          = 2,
INS_PicFonts          = 3,
INS_MruColors         = 4,

 // text/external
INS_SSPlayInfo        = 1,

// slideshow
INS_AnimationInfo     = 0,
INS_InteractiveInfo   = 1,
INS_SlideNotes        = 3,
INS_DocNotes          = 4,
INS_Sounds            = 5,
INS_SSOEInfo          = 6,

// Named shows
INS_NamedShows        = 0,
INS_NamedShowName     = 1,
INS_NamedShowSlides   = 2,

 // HeadersFooters
INS_UserDate          = 0,
INS_Header            = 1,
INS_Footer            = 2,
INS_SlideHeadersFooters = 3,
INS_NotesHeadersFooters = 4,

 // Summary Info
INS_BookmarkCollection = 0,
INS_BookmarkValue     = 1,
INS_BookmarkSeedAtom  = 2,

// Textures
INS_TextureName       = 0,

// TagName
INS_TagName           = 0,
INS_TagValue          = 1,
```

```
   // DocInfoList
  INS_SlideViewInfo     = 0,
  INS_NotesViewInfo     = 1,
  INS_HandoutViewInfo   = 2,
  INS_SlideShowWindowViewInfo = 3,

  // ExControl
  INS_StreamName        = 0,

  // Placeholder
  INS_PlaceholderInfo   = 0,

  // InteractiveInfo trigger
  INS_MouseClick        = 0,
  INS_MouseOver         = 1,
};



//====================== Versions
=============================================
//=================================================================
=======

#define VER_PowerPointStateInfoAtom 0

/* Document & Slide */
#define VER_DocumentAtom         0
#define VER_SlideBaseAtom        0
#define VER_SlideAtom            0
#define VER_NotesAtom            0
#define VER_SchemeAtom           0
#define VER_SSlideLayoutAtom     0
#define VER_SSSlideInfoAtom      0
#define VER_GuideAtom            0
#define VER_ViewInfoAtom         0
#define VER_SlideViewInfoAtom    0
#define VER_VBAInfoAtom          0
#define VER_VBASlideInfoAtom     0
#define VER_SSDocInfoAtom        0
#define VER_Summary              0
#define VER_TxStylesAtom         0
#define VER_ExObjListAtom        0     /* new for PP96 */
#define VER_SoundCollAtom        0     /* new for PP96 */

/* Collections & lists */
#define VER_RunArrayAtom         0
#define VER_ArrayElementAtom     0
#define VER_Int4ArrayAtom        0
#define VER_ColorSchemeAtom      0     /* new for PP96 */

/* Slide Elements */
#define VER_ExObjRefAtom         0
#define VER_OEPlaceholderAtom    0
#define VER_GRectAtom            0
```

```
/* Text, Rulers, External */
#define VER_TextCharsAtom        0
#define VER_StyleTextPropAtom    0
#define VER_BaseTextPropAtom     0
#define VER_TxMasterStyleAtom    0
#define VER_TxCFStyleAtom        0
#define VER_TxPFStyleAtom        0
#define VER_FontEntityAtom       0
#define VER_ExOleObjAtom         0
#define VER_ExEmbedAtom          0
#define VER_ExLinkAtom           0
#define VER_ExControlAtom        0
#define VER_ExPlainAtom          0
#define VER_ExPlainLinkAtom      0
#define VER_ExHyperlinkAtom      0
#define VER_CorePictAtom         0
#define VER_BookmarkEntityAtom   0
#define VER_SrKinsokuAtom        0
#define VER_TxStyleEntryAtom     0
#define VER_BookmarkSeedAtom     0
#define VER_HeadersFooters       0
#define VER_HeadersFootersAtom   0
#define VER_SlideNumberMCAtom    0
#define VER_DateTimeMCAtom       0
#define VER_GenericDateMCAtom    0
#define VER_HeaderMCAtom         0
#define VER_FooterMCAtom         0

#define VER_RecolorEntryAtom     0
#define VER_RecolorInfoAtom      0
#define VER_EmFormatAtom         0
#define VER_ParaFormatAtom       0
#define VER_CharFormatAtom       0
#define VER_RulerTabArrayAtom    0
#define VER_AnimationInfoAtom    0
#define VER_InteractiveInfoAtom  0
#define VER_CStringAtom          0
#define VER_SlideListAtom        0
#define VER_UserEditAtom         0
#define VER_CurrentUserAtom      0


/* External Media */
#define VER_ExMediaAtom          0

/* Programmable Tags */
#define VER_ProgTagsAtom         0

/* Print Options */
#define VER_PrintOptions         0


//====================== Persistent Storage Records
==========================
//=====================================================================
=======
```

```
typedef sint4  PSR_GCoord;
typedef sint4  PSR_GLCoord;

struct PSR_GPointAtom
{
   sint4   x;
   sint4   y;
};

struct PSR_GRatioAtom
{
   sint4 numer;
   sint4 denom;
};

struct PSR_GScalingAtom
{
 PSR_GRatioAtom x;
 PSR_GRatioAtom y;
};

enum
{
   F_SCALE   = 16,
   F_DEG90   = 90 * F_SCALE,
   F_DEG360  = 360 * F_SCALE,
};


struct PSR_GRectAtom
{
   sint4  left;
   sint4  top;
   sint4  right;
   sint4  bottom;
};

struct PSR_GrColorAtom
{
   ubyte1 red;
   ubyte1 green;
   ubyte1 blue;
   ubyte1 pad;
};


struct PSR_EmFormatAtom
{
   sint4 ref;
};

// Font
#define PSR_LF_FACESIZE    32

struct PSR_FontEntityAtom
{
```

```
 // members of logfont
 uint2   lfFaceName[PSR_LF_FACESIZE];
 ubyte1  lfCharSet;
 ubyte1  lfClipPrecision;
 ubyte1  lfQuality;
 ubyte1  lfPitchAndFamily;
};

#define PSR_BOOKMARKNAMESIZE  32

struct PSR_BookmarkEntityAtom
{
 uint4   bookmarkID;
 uint2   bookmarkName[PSR_BOOKMARKNAMESIZE];
};

struct PSR_BookmarkSeedAtom
{
 uint4   bookmarkID;
};

typedef PSR_GPointAtom PSR_TxCtrOfRotAtom;

/*********************************************************************
********
   classes related to OElement
*********************************************************************
*******/

typedef sint4 FEAlignment;

enum                        // can OR one horizontal and one vertical
{  // horizontal alignments
   FE_ALIGN_LEFT        = 0x0001,      // left edges
   FE_ALIGN_CENTER      = 0x0002,      // horizontal center
   FE_ALIGN_RIGHT       = 0x0003,      // right edges
   FE_ALIGN_HORIZONTAL  = 0x000F,      // mask for horizontal component

   // vertical alignments
   FE_ALIGN_TOP         = 0x0010,      // top edges
   FE_ALIGN_MIDDLE      = 0x0020,      // vertical center
   FE_ALIGN_BOTTOM      = 0x0030,      // bottom edges
   FE_ALIGN_VERTICAL    = 0x00F0,      // mask for vertical component
};

typedef ubyte1 FELineStyle;

typedef ubyte1 FEArrowStyle;    // arrowhead style is on or off
enum
{
   FE_ARROW_NONE = 0,
   FE_ARROW_NORMAL,
   FE_ARROW_ROUND,
   FE_ARROW_DIAMOND
};

typedef ubyte1 FELineStyle;
```

```
#define   F_LT_SOLIDLINE   0                  // solid colored line
#define   F_LT_DASH1       1                      // Dash Pattern 1
#define   F_LT_DASH2       2                      // Dash Pattern 2
#define   F_LT_DASH3       3                            // Dash Pattern
3
#define   F_LT_DASH4       4                      // Dash Pattern 4

typedef ubyte1 FEShadowType;
#define F_ST_COLOREDSHADOW         (1)   // solid, colored shadow
#define F_ST_TRANSPARENTSHADOW     (2)   // transparent, colored shadow
#define F_ST_EMBOSSEDSHADOW        (3)   // double-shadow with embossed
effect

typedef ubyte1 FEFillType;

#define F_FT_SOLIDFILL             (1)   // solid colored fill
#define F_FT_BACKGROUNDFILL        (2)   // automatic fill with slide
background
#define F_FT_TRANSPARENTFILL       (3)   // transparent fill
#define F_FT_PATTERNEDFILL         (4)   // patterned fill
#define F_FT_SHADEDFILL            (5)   // shaded fill (fade)
#define F_FT_TEXTUREDFILL          (6)   // textured bitmap fill
#define F_FT_PICTUREFILL           (7)   // fill with a picture

enum
{  FDimX        = 0,
   FAnimateX    = 2,
   FLastFlag    = 2
};


typedef ubyte1 FEPlaceholderId;
enum
{  FE_PLACE_NONE = 0,
   FE_PLACE_MASTER_TITLE,
   FE_PLACE_MASTER_BODY,
   FE_PLACE_MASTER_CENTERTITLE,
   FE_PLACE_MASTER_SUBTITLE,
   FE_PLACE_MASTER_NOTES_SLIDEIMAGE,
   FE_PLACE_MASTER_NOTES_BODY,
   FE_PLACE_MASTER_DATE,
   FE_PLACE_MASTER_SLIDENUMBER,
   FE_PLACE_MASTER_FOOTER,
   FE_PLACE_MASTER_HEADER,
   FE_PLACE_NOTES_SLIDEIMAGE,
   FE_PLACE_NOTES_BODY,
   FE_PLACE_TITLE,
   FE_PLACE_BODY,
   FE_PLACE_CENTERTITLE,
   FE_PLACE_SUBTITLE,
   FE_PLACE_V_TITLE,
   FE_PLACE_V_BODY,
   FE_PLACE_OBJECT,
   FE_PLACE_GRAPH,
   FE_PLACE_TABLE,
   FE_PLACE_CLIPART,
```

```
   FE_PLACE_ORGCHART,
   FE_PLACE_MEDIA,
   FE_PLACE_FIRST = FE_PLACE_MASTER_TITLE,
   FE_PLACE_LAST = FE_PLACE_MEDIA
};

typedef ubyte1 FEPlaceholderSize;
enum
{  FE_SIZE_FULL,
   FE_SIZE_HALF,
   FE_SIZE_QUART
};


typedef sint4 FLayout ;
enum
{
   F_GEOM_TITLE_SLIDE,    // title moved down, center aligned body
below it
   F_GEOM_TITLE_BODY,     // standard title/body layout copied from
master
   F_GEOM_TITLE_ONLY,     // title only, no body placeholder
   F_GEOM_2_COLUMNS,      // body split into 2 columns
   F_GEOM_2_ROWS,         // body split into 2 rows
   F_GEOM_COLUMN_2_ROWS,  // body split into 2 columns, right column
has 2 rows
   F_GEOM_2_ROWS_COLUMN,  // body split into 2 columns, left column has
2 rows
   F_GEOM_ROW_2_COLUMNS,  // body split into 2 rows, bottom row has 2
columns
   F_GEOM_2_COLUMNS_ROW,  // body split into 2 rows, top row has 2
columns
   F_GEOM_4_OBJECTS,      // body split into 4 objects
   F_GEOM_BIG_OBJECT,     // title and body combined into one big
object
   F_GEOM_BLANK           // neither title nor body
};

enum
{  F_ManualAdvanceX = 0,
   F_HiddenX        = 2,
   F_SoundX         = 4,
   F_LastFlag       = 4
};

enum
{ F_Layout= 0,
 F_Look,
 F_Notes
};

#define MAX_OBJECTS_IN_LAYOUT 8     // no layout has more than 5
objects
struct PSR_SSlideLayoutAtom
{
   sint4 geom;
   ubyte1 placeholderId[ MAX_OBJECTS_IN_LAYOUT ];
```

```
};

struct PSR_DocumentAtom
{
   PSR_GPointAtom      slideSize;      // slide size in master coords
   PSR_GPointAtom      notesSize;      // notes page size in master
coords
   PSR_GRatioAtom      serverZoom;
   uint2               firstSlideNum;
   sint2               slideSizeType;  // size type: A4, screen,
custom, etc.
   bool1               saveWithFonts;
   bool1               omitTitlePlace; // omit placeholders on title
slide
   bool1               rightToLeft;    // right-to-left document
(Middle East)
   bool1               showComments;   // are comments visible
};


struct PSR_UserEditAtom
{
   sint4  lastSlideID;    // slideID
   uint4  version;        // This is major/minor/build which did the
edit
   uint4  offsetLastEdit; // File offset of last edit
   uint4  offsetPersistDirectory; // Offset to PersistPtrs for
                                  // this file version.
   uint4  documentRef;
   uint4  maxPersistWritten;      // Addr of last persist ref written
to the file (max seen so far).
   sint2  lastViewType;   // enum view type
};

// This is written to the current user stream.  It is a variable length
// record, whose true size includes a sequence of bytes after this
structure
// which is the current User's name.
// NOTE: We don't support incremental records of different machine
types,
//       thus saving a file on the other plaform will involve a full
save.
struct PSR_CurrentUserAtom
{
   uint4  size;
   uint4  magic;  // Magic number to ensure this is a PowerPoint file.
   uint4  offsetToCurrentEdit;  // Offset in main stream to current
edit field.
   uint2     lenUserName;
   uint2  docFileVersion;
   ubyte1 majorVersion;
   ubyte1 minorVersion;
};

struct PSR_ExObjListAtom
{
   sint4  objectIdSeed;   // next unique identifier for ole objects
```

```
};

struct PSR_SoundCollAtom
{
   sint4  objectIdSeed;   // next unique identifier for ole objects
};

const int MST_FLAG_OBJECTS = 0x01;
const int MST_FLAG_SCHEME = 0x02;
const int MST_FLAG_BACKGROUND = 0x04;

struct PSR_SlideBaseAtom
{
   PSR_GRectAtom    rect;       // size in master coordinates

   uint2  flags; // Replaces below
};


struct PSR_SlideAtom
{
   PSR_SlideBaseAtom base;       // base attributes
   PSR_SSlideLayoutAtom layout;
   sint4       slideId;
   sint4       masterId;         // Id of master slide
};

struct PSR_NotesAtom
{
   sint4 slideID;
};

struct PSR_ExObjRefAtom
{
   uint4 exObjId;
};

struct PSR_OEPlaceholderAtom
{
   uint4    placementId;               // the placeholders position.
   ubyte1   placeholderId;             // Place holder number
   ubyte1   size;                      // the placeholders size.
};


//====================== Text ========================================
//=================================================================
=======
// Containers:
//    PSR_SrKinsokuAtom

const int S_HEADERFOOTER_DATE =        0x01;
const int S_HEADERFOOTER_TODAYDATE =   0x02;
const int S_HEADERFOOTER_USERDATE =    0x04;
const int S_HEADERFOOTER_SLIDENUMBER = 0x08;
const int S_HEADERFOOTER_HEADER =      0x10;
const int S_HEADERFOOTER_FOOTER =      0x20;
```

```
struct PSR_HeadersFootersAtom
{
   sint2 formatId;
   uint2 flags;    // date, todayDate, userDate, slideNumber, header,
footer
};

struct PSR_SlideNumberMCAtom
{  sint4 position; // position in text
};

struct PSR_DateTimeMCAtom
{  sint4  position; // position in text
   ubyte1 index;    // the date/time index
};

struct PSR_GenericDateMCAtom
{  sint4 position; // position in text
};

struct PSR_HeaderMCAtom
{  sint4 position; // position in text
};

struct PSR_FooterMCAtom
{  sint4 position; // position in text
};

struct PSR_SrKinsokuAtom
{
   sint4 level;
};

struct PSR_ExPlainAtom
{
    sint4  objID; // persistent unique identifier for external object
};

struct PSR_ExPlainLinkAtom
{
    sint4  objID; // persistent unique identifier for external object
};

struct PSR_ExHyperlinkAtom
{
    sint4  objID; // persistent unique identifier for external object
};

// ExOleObject
struct PSR_ExOleObjAtom
{
   uint4  drawAspect;
   sint4  type;  // whether embedded or linked ?
   sint4  objID; // persistent unique identifier for external object
   sint4  subType;
   bool1  isBlank; // true if object has no presentation data
```

```
};

// ExEmbed
struct PSR_ExEmbedAtom
{
   sint4        followColorScheme;
   bool1        cantLockServerB;
   bool1        noSizeToServerB;
   bool1        isTable;
};

// ExLink
struct PSR_ExLinkAtom
{
   uint4        updateMode;
   bool1        unavailable;
};

// ExControl
struct PSR_ExControlAtom
{  bool1        useIStream;
};


struct PSR_RecolorEntryAtom
{
   PSR_GrColorAtom toColor;
   PSR_GrColorAtom fromColor;
   bool1           doRecolor;
};

struct PSR_RecolorInfoAtom
{
 PSR_GrColorAtom monoColor;
 sint4  nColors;
 sint4  nFills;
 uint2  flags;
};


struct PSR_CorePictAtom
{
 PSR_GRectAtom  frame;      // frame of the picture.
 bool1          isVirtual;  // Is memory handle virtual?
};


#define PSR_NAMEDSHOW_SIZE    32

struct PSR_SSDocInfoAtom
{
    PSR_GrColorAtom  penColor;
    sint4            restartTime;
    sint2            startSlide;
    sint2            endSlide;
    uint2            namedShow[PSR_NAMEDSHOW_SIZE];
    ubyte1           flags;
```

```
};

struct PSR_SSSlideInfoAtom
{
   sint4        slideTime;     // how long to show the slide in ticks
   uint4        soundRef;
   uint2        effect;        // type of transition (2 character
signature)
   uint2        flags;         // set of flags that determine type of
build
   ubyte1       speed;         // speed of transition
};  // slide show info

struct PSR_AnimationInfoAtom
{
  PSR_GrColorAtom  dimColor;
  uint4            flags;
  uint4            soundRef;
  uint2            orderID;
  uint2            delayTime;
  uint2            slideCount;
  ubyte1           buildType;
  ubyte1           flyMethod;
  ubyte1           flyDirection;
  ubyte1           afterEffect;
  ubyte1           subEffect;
  ubyte1           oleVerb;
};

struct PSR_InteractiveInfoAtom
{
 uint4      soundRef;
 uint4      exHyperlinkID;
 ubyte1     action;
 ubyte1     oleVerb;
 ubyte1     jump;
 ubyte1     flags;
};


// External Media related Atoms
struct PSR_ExMediaAtom
{
   uint4    exObjId;    // All objects derived from ExternalObject must
save/load their id
   uint2    flags;
};



// View Info

struct PSR_ViewInfoAtom
{
 PSR_GScalingAtom  curScale;
 PSR_GScalingAtom  prevScale;
 PSR_GPointAtom    viewSize;
```

```
 PSR_GPointAtom    origin;
 bool1             varScale;
 bool1             draftMode;
};

struct PSR_GuideAtom
{
 sint4   type;  // guide type
 sint4   pos;   // position in master coordinates
                // x if vertical; y if horizontal
};

// DocViewInfo

struct PSR_SlideViewInfoAtom
{
   bool1 showGuides;
   bool1 snapToGrid;
   bool1 snapToShape;
};

// VBA
struct PSR_VBAInfoAtom
{
 uint4   state;  // Project State
 uint4   autoLoad; // Bring project into running state immediately
 uint4   version; // version number, 0 and 1: old VBA, 2: new VBA^3
};

struct PSR_VBASlideInfoAtom
{
 uint4   state;  // Project State
};

// VBAProject

struct PSR_SchemeAtom
{
   uint4 tableSize;
};

struct PSR_ColorSchemeAtom
{
   uint4 color[8]; // 8 COLORREFs in color scheme
};


// Indexed Record Reference Atom

struct PSR_IRRAtom
{
   uint4 indexID;          // Which index to use   indexToUse =
indexMap.Lookup(indexID)
   uint4 indexKey;         // location = index.Lookup(indexKey)
};
```

```
struct PSR_PowerPointStateInfoAtom
{
   uint4 curViewType;
   uint4 curSlideId;
};


struct PSR_ProgTagsAtom
{
   uint4 nTags;
};


// Per-document options:

struct PSR_PrintOptions
{
   ubyte1   printWhat;
   bool1    printHidden;
   bool1    printBlackWhite;
   bool1    printPureBlackWhite;
   bool1    scaleToFitPaper;
   bool1    frameSlides;
};
```

Appendix C:

```
//
// Sample code to read the text out of a PowerPoint '97
presentation.
//

#include <ole2.h>
#include <stdio.h>
#include <time.h>

// Stolen from app\sertypes.h
// system dependent sizes
// system dependent sizes
typedef signed long     sint4;            // signed 4-byte
integral value
typedef signed short    sint2;            // signed 4-byte
integral value
typedef unsigned long   uint4;            // unsigned 4-byte
integral value
typedef unsigned short  uint2;            //          2-byte
typedef char            bool1;            // 1-byte boolean
typedef unsigned char   ubyte1;           // unsigned byte
value
typedef uint2           psrType;
typedef uint4           psrSize;          // each record is
preceeded by
                                          // pssTypeType and
pssSizeType.
typedef uint2          psrInstance;
typedef uint2          psrVersion;
typedef uint4          psrReference;     // Saved object
reference


#define PSFLAG_CONTAINER 0xFF             // If the version
field of a record
                                          //  header takes on
this value, the
                                          //  record header
marks the start of
                                          //  a container.
// PowerPoint97 Record Header
typedef unsigned long DWord;

int AssertionFailed( const char* file, int line, const char*
expr )
/*=================*/
{// AR: Message box the assert
   return( TRUE );
} /* AssertionFailed */


#define Assert( expr )
\
{
\
   static char _str[] = #expr;
\

\
   if( !(int)(expr) )
\
```

```
      AssertionFailed( __FILE__, __LINE__, _str );
\
} /* Assert */

static BOOL ReadText( WCHAR* buffer, unsigned long bufferSize,
unsigned long* pSizeRet );
// Returns TRUE if more text exists.  Fills buffer upto
bufferSize.  Actual size used is
// pSizeRet.


struct RecordHeader
{
   psrVersion     recVer      : 4;                  // may be
PSFLAG_CONTAINER
   psrInstance    recInstance : 12;
   psrType        recType;
   psrSize        recLen;
};


struct PSR_CurrentUserAtom
{
   uint4  size;
   uint4  magic;  // Magic number to ensure this is a
PowerPoint file.
   uint4  offsetToCurrentEdit;  // Offset in main stream to
current edit field.
   uint2  lenUserName;
   uint2  docFileVersion;
   ubyte1 majorVersion;
   ubyte1 minorVersion;
};

struct PSR_UserEditAtom
{
   sint4  lastSlideID;    // slideID
   uint4  version;        // This is major/minor/build which
did the edit
   uint4  offsetLastEdit; // File offset of last edit
   uint4  offsetPersistDirectory; // Offset to PersistPtrs for
                             // this file version.
   uint4  documentRef;
   uint4  maxPersistWritten;      // Addr of last persist ref
written to the file (max seen so far).
   sint2  lastViewType;   // enum view type
};

struct PSR_SlidePersistAtom
{
   uint4  psrReference;
   uint4  flags;
   sint4  numberTexts;
   sint4  slideId;
   uint4  reserved;
};


#define CURRENT_USER_STREAM      L"Current User"
#define DOCUMENT_STREAM          L"PowerPoint Document"
#define HEADER_MAGIC_NUM         -476987297

const int PST_UserEditAtom       = 4085;
```

```
const int PST_PersistPtrIncrementalBlock = 6002; // Incremental
diffs on persists
const int PST_SlidePersistAtom    = 1011;
const int PST_TextCharsAtom       = 4000;  // Unicode in text
const int PST_TextBytesAtom       = 4008;  // non-unicode text

class PPSPersistDirectory;

struct ParseContext
{
   ParseContext(ParseContext *pNext) : m_pNext(pNext),
m_nCur(0) {}

   RecordHeader  m_rh;
   uint4         m_nCur;
   ParseContext *m_pNext;
};

const int SLIDELISTCHUNKSIZE=32;

struct SlideListChunk
{
   SlideListChunk( SlideListChunk* next, psrReference newOne )
:
      pNext( next ), numInChunk(1) { refs[0] = newOne; }
   SlideListChunk *pNext;
   DWord numInChunk;
   psrReference refs[SLIDELISTCHUNKSIZE];
};

class FileReader
{
public:
   FileReader(IStorage *pStg);
   ~FileReader();

   BOOL ReadText( WCHAR *pBuff, ULONG size, ULONG *pSizeRet );
   // Reads next size chars from file.  Returns TRUE if there
is more
   // text to read.

   BOOL IsPowerPoint() { return m_isPP; } // Returns true if
this is a PowerPoint '97 file.

   void ReadPersistDirectory();
   void PPSReadUserEditAtom( DWord offset, PSR_UserEditAtom&
userEdit );
   void ReadSlideList();

protected:
   BOOL ReadCurrentUser(IStream *pStm);
   void *ReadRecord( RecordHeader& rh );

   BOOL Parse();
   IStream *GetDocStream();
   BOOL DoesClientRead( psrType type ) { return FALSE; }
   void ReleaseRecord( RecordHeader& rh, void* diskRecBuf );
   DWord ParseForSlideLists();
   void AddSlideToList( psrReference refToAdd );
   BOOL StartParse( DWord offset );
   BOOL FillBufferWithText();
   BOOL FindNextSlide( DWord& offset );

private:
```

```
   PSR_CurrentUserAtom m_currentUser;
   IStream *           m_pDocStream;
   IStorage *          m_pPowerPointStg;
   BOOL                 m_isPP;
   ParseContext*        m_pParseContexts;

   WCHAR*               m_pCurText;
   unsigned long        m_curTextPos;
   unsigned long        m_curTextLength;

   PSR_UserEditAtom*    m_pLastUserEdit;
   PPSPersistDirectory* m_pPersistDirectory;
   SlideListChunk*      m_pFirstChunk;
   int                  m_curSlideNum;

   WCHAR*               m_pClientBuf;
   unsigned long        m_clientBufSize;
   unsigned long        m_clientBufPos;
   ULONG*               m_pSizeRet;
};


FileReader::FileReader(IStorage *pStg) :
   m_pPowerPointStg(pStg),
   m_isPP(FALSE),
   m_pParseContexts(NULL),
   m_curTextPos(0),
   m_pLastUserEdit( NULL ),
   m_pPersistDirectory( NULL ),
   m_pDocStream( NULL ),
   m_pFirstChunk( NULL ),
   m_curSlideNum(0),
   m_pCurText( NULL ),
   m_pClientBuf( NULL ),
   m_clientBufSize( 0 ),
   m_clientBufPos( 0 )
{
   IStream *pStm = NULL;
   m_pPowerPointStg->AddRef();
   HRESULT hr = pStg->OpenStream( CURRENT_USER_STREAM, NULL,
STGM_READ | STGM_DIRECT | STGM_SHARE_EXCLUSIVE, NULL, &pStm );
   if( SUCCEEDED(hr) && ReadCurrentUser(pStm) )
      m_isPP = TRUE;
   pStm->Release();
}

FileReader::~FileReader()
{
   m_pPowerPointStg->Release();
}

BOOL FileReader::FillBufferWithText()
{
   unsigned long amtToCopy = min( (m_curTextLength -
m_curTextPos), (m_clientBufSize - m_clientBufPos) );
   unsigned long loop = amtToCopy;
   while( loop-- )
      m_pClientBuf[ m_clientBufPos++ ] = m_pCurText[
m_curTextPos++ ];
   if( m_curTextPos == m_curTextLength )
   {
      delete [] m_pCurText;
      m_pCurText = NULL;
      m_curTextPos = 0;
```

```
      m_curTextLength = 0;
   }
   *m_pSizeRet += amtToCopy;
   return (m_clientBufSize == m_clientBufPos); // If client's
buffer is full return TRUE.
}

void FileReader::AddSlideToList( psrReference refToAdd )
{
   if( m_pFirstChunk == NULL )
      m_pFirstChunk = new SlideListChunk(NULL, refToAdd);
   else
   {
      if( m_pFirstChunk->numInChunk+1 > SLIDELISTCHUNKSIZE )
         m_pFirstChunk = new SlideListChunk(m_pFirstChunk,
refToAdd);
      else
      {
         m_pFirstChunk->refs[m_pFirstChunk->numInChunk] =
refToAdd;
         m_pFirstChunk->numInChunk++;
      }
   }
}

IStream *FileReader::GetDocStream()
{
   if( m_pDocStream == NULL )
   {
      if( !m_isPP )
         return NULL;
      HRESULT hr = m_pPowerPointStg->OpenStream(
DOCUMENT_STREAM, NULL, STGM_READ | STGM_DIRECT |
STGM_SHARE_EXCLUSIVE, NULL, &m_pDocStream );
      if (FAILED(hr))
      {
            fprintf(stderr,"Error (%d) opening PowerPoint
Document Stream.\n",(int)hr);
         return NULL;
      }
   }
   return m_pDocStream;

}

BOOL FileReader::ReadCurrentUser(IStream *pStm)
{
   ULONG nRd=0;
   RecordHeader rh;
   BOOL isPP = FALSE;
   if( SUCCEEDED( pStm->Read(&rh, sizeof(rh), &nRd) ) )
   {
      if( SUCCEEDED( pStm->Read(&m_currentUser,
sizeof(PSR_CurrentUserAtom), &nRd) ) )
      {
         if( nRd != sizeof(PSR_CurrentUserAtom) )
            return FALSE;
      }
      isPP = ( m_currentUser.size == sizeof( m_currentUser )
)&&
             ( m_currentUser.magic == HEADER_MAGIC_NUM )&&
             ( m_currentUser.lenUserName <= 255        );
   }
```

```
   return isPP;
}


class PPSDirEntry
{
   PPSDirEntry()
    : m_pNext( NULL ), m_pOffsets( NULL ), m_tableSize( 0 ){}

   PPSDirEntry* m_pNext;
   DWord*       m_pOffsets;
   DWord        m_tableSize;
public:
   ~PPSDirEntry(){ delete m_pOffsets; m_pOffsets = NULL; }

friend class PPSPersistDirectory;
}; // class PPSDirEntry

class PPSPersistDirectory
{
public:
   PPSPersistDirectory();

   ~PPSPersistDirectory();

   void  AddEntry( DWord cOffsets, DWord* pOffsets );
   DWord GetPersistObjStreamPos( DWord ref );
   DWord NumberOfAlreadySavedPersists();

private:
   PPSDirEntry* m_pFirstDirEntry;
};


PPSPersistDirectory::PPSPersistDirectory() : m_pFirstDirEntry(
NULL ){}

PPSPersistDirectory::~PPSPersistDirectory()
{
   while( m_pFirstDirEntry )
   {
      PPSDirEntry* pDirEntry = m_pFirstDirEntry;
      m_pFirstDirEntry = m_pFirstDirEntry->m_pNext;
      delete pDirEntry;
   }
}

void PPSPersistDirectory::AddEntry( DWord cOffsets, DWord*
pOffsets )
{
   PPSDirEntry* pDirEntry = new PPSDirEntry();

   pDirEntry->m_tableSize = cOffsets;
   pDirEntry->m_pOffsets = new DWord[cOffsets];
   memcpy( pDirEntry->m_pOffsets, pOffsets, cOffsets * sizeof(
DWord ) );

   // append to the end of the entry list
   PPSDirEntry** ppDirEntry = &m_pFirstDirEntry;
   while( NULL != *ppDirEntry )
      ppDirEntry = &(*ppDirEntry)->m_pNext;
   *ppDirEntry = pDirEntry;
}
```

```
DWord PPSPersistDirectory::GetPersistObjStreamPos( DWord ref )
{
   PPSDirEntry* pEntry = m_pFirstDirEntry;
   while( pEntry )
   {
      DWord* pOffsets = pEntry->m_pOffsets;
      while( (DWord)( (char*)pOffsets - (char*)pEntry-
>m_pOffsets ) < pEntry->m_tableSize * sizeof( DWord ) )
      {
         DWord nRefs = pOffsets[0] >> 20;
         DWord base = pOffsets[0] & 0xFFFFF; // 1-based
         if( ( base <= ref )&&( ref < base + nRefs ) )
            return pOffsets[ 1 + ref - base ];
         pOffsets += nRefs + 1;
      }
      pEntry = pEntry->m_pNext;
   }
   return (DWord) -1;
}

DWord PPSPersistDirectory::NumberOfAlreadySavedPersists()
{
   DWord count = 0;
   PPSDirEntry* pEntry = m_pFirstDirEntry;
   while( pEntry )
   {
      DWord* pOffsets = pEntry->m_pOffsets;
      while( (DWord)( pEntry->m_pOffsets - pOffsets ) < pEntry-
>m_tableSize * sizeof( DWord ) )
      {
         DWord nRefs = pOffsets[0] >> 20;
         count += nRefs;
         pOffsets += nRefs + 1;
      }
      pEntry = pEntry->m_pNext;
   }
   return count;
}

void FileReader::PPSReadUserEditAtom( DWord offset,
PSR_UserEditAtom& userEdit )
{
   LARGE_INTEGER li;
   li.LowPart = offset;
   li.HighPart = 0;
   GetDocStream()->Seek(li,STREAM_SEEK_SET, NULL);
   RecordHeader rh;
   GetDocStream()->Read(&rh, sizeof(rh), NULL);
   Assert( rh.recType == PST_UserEditAtom );
   Assert( rh.recLen == sizeof( PSR_UserEditAtom ) );
   li.LowPart = offset;
   GetDocStream()->Read(&userEdit, sizeof(userEdit), NULL);
}


void *FileReader::ReadRecord( RecordHeader& rh )
// Return values:
// NULL and rh.recVer == PSFLAG_CONTAINER: no record was read
in.
//    record header indicated start of container.
// NULL and rh.recVer != PSFLAG_CONTAINER: client must read in
record.
{
   IStream *pStm = GetDocStream();
```

```
   // read record header, verify
   pStm->Read(&rh, sizeof(rh), NULL);  //AR: Check Error

   // if client will read, do not read in record
   if( DoesClientRead( rh.recType ) )
      return NULL;

   // If container, return NULL
   if(rh.recVer == PSFLAG_CONTAINER)
      return NULL;


   // Allocate buffer for disk record. Client must call
ReleaseRecord() or
   // pass the atom up to CObject::ConstructContents() which
will
   // then release it.
   void* buffer = new char[rh.recLen];

   // read in record
   pStm->Read(buffer, rh.recLen, NULL);

   // NOTE: ByteSwapping & versioning not done by this simple
reader.
   return (buffer);
}

void FileReader::ReleaseRecord( RecordHeader& rh, void*
diskRecBuf )
{
   if(rh.recType && rh.recVer!=PSFLAG_CONTAINER)
      delete [] (char*)diskRecBuf;
   rh.recType = 0;         // consume the record so that record
doesn't
                           // get processed again.
}

void FileReader::ReadPersistDirectory()
{
   if( NULL != m_pLastUserEdit )
      return; // already read

   PSR_UserEditAtom userEdit;
   DWord offsetToEdit = m_currentUser.offsetToCurrentEdit;

   while( 0 < offsetToEdit )
   {
      PPSReadUserEditAtom( offsetToEdit, userEdit );
      if( NULL == m_pLastUserEdit )
      {
         m_pPersistDirectory = new PPSPersistDirectory();
         m_pLastUserEdit     = new PSR_UserEditAtom;
         *m_pLastUserEdit = userEdit;
      }
      LARGE_INTEGER li;
      li.LowPart = userEdit.offsetPersistDirectory;
      li.HighPart = 0;
      GetDocStream()->Seek(li,STREAM_SEEK_SET, NULL); // AR:
check that seek succeeded.
      RecordHeader rh;
      DWord *pDiskRecord = (DWord*) ReadRecord(rh);
      Assert( PST_PersistPtrIncrementalBlock == rh.recType );
      m_pPersistDirectory->AddEntry( rh.recLen / sizeof( DWord
), pDiskRecord );
```

```
      ReleaseRecord( rh, pDiskRecord );
      offsetToEdit = userEdit.offsetLastEdit;
   }
} // PPStorage::ReadPersistDirectory

void FileReader::ReadSlideList()
{
   Assert( m_pLastUserEdit != NULL );
   DWord offsetToDoc = m_pPersistDirectory-
>GetPersistObjStreamPos( m_pLastUserEdit->documentRef );
   LARGE_INTEGER li;
   li.LowPart = offsetToDoc;
   li.HighPart = 0;
   GetDocStream()->Seek(li,STREAM_SEEK_SET, NULL);
   ParseForSlideLists();
}

DWord FileReader::ParseForSlideLists()
{
   IStream *pStm = GetDocStream();

   RecordHeader rh;
   DWord nRd=0;
   // Stack based parsing for SlideLists
   pStm->Read(&rh, sizeof(rh), &nRd);
   if( ( rh.recVer != PSFLAG_CONTAINER ) && ( (rh.recVer &
0x0F)!=0x0F ) )
   {
      if( rh.recType == PST_SlidePersistAtom )
      {
         PSR_SlidePersistAtom spa;
         Assert( sizeof(spa) == rh.recLen );
         pStm->Read(&spa, sizeof(spa), &nRd);
         AddSlideToList( spa.psrReference );
      }
      else
      {
         LARGE_INTEGER li;
         li.LowPart = rh.recLen;
         li.HighPart = 0;
         pStm->Seek(li,STREAM_SEEK_CUR, NULL);
      }
      nRd += rh.recLen;
   }
   else
   {
      DWord nCur = 0;
      while( nCur < rh.recLen )
      {
         nCur += ParseForSlideLists();
      }
      nRd += nCur;
   }
   return nRd;

}

BOOL FileReader::ReadText( WCHAR *pBuff, ULONG size, ULONG
*pSizeRet )
{
   DWord offset;
   *pSizeRet = 0;
   m_pSizeRet = pSizeRet;
   m_pClientBuf = pBuff;
```

```
   m_clientBufSize = size;
   m_clientBufPos = 0;

   for( ;; )
   {
      if( ( m_pParseContexts == NULL ) )
      {
         if( FindNextSlide(offset) )
         {
            if( StartParse( offset ) )
               return TRUE;
         }
         else
            return FALSE; // DONE parsing, no more slides
      }
      else
      {
         if( m_pClientBuf )
         {
            if( FillBufferWithText() ) // Use existing text
first.
               return TRUE;
         }
         if( Parse() ) // restart parse where we left off.
            return TRUE;
      }
   }
}

BOOL FileReader::StartParse( DWord offset )
{
   LARGE_INTEGER li;
   li.LowPart = offset;
   li.HighPart = 0;
   GetDocStream()->Seek(li,STREAM_SEEK_SET, NULL);
   m_pParseContexts = new ParseContext( NULL );
   GetDocStream()->Read(&m_pParseContexts->m_rh,
sizeof(RecordHeader), NULL);
   return Parse();
}

BOOL FileReader::Parse()
{
   IStream *pStm = GetDocStream();

   RecordHeader rh;
   DWord nRd=0;
   Assert( m_pParseContexts );
   // Restarting a parse might complete a container so we test
this initially.
   if( m_pParseContexts->m_nCur >= m_pParseContexts-
>m_rh.recLen )
   {
      Assert(  m_pParseContexts->m_nCur == m_pParseContexts-
>m_rh.recLen );
      ParseContext* pParseContext = m_pParseContexts;
      m_pParseContexts = m_pParseContexts->m_pNext;
      delete pParseContext;
   }

   do
   {
      pStm->Read(&rh, sizeof(RecordHeader), NULL);
```

```
      if( ( rh.recVer != PSFLAG_CONTAINER ) && ( (rh.recVer &
0x0F)!=0x0F ) )
      {
         if( rh.recType == PST_TextCharsAtom )
         {
            m_curTextPos = 0;
            m_curTextLength = rh.recLen/2;
            Assert( m_pCurText == NULL );
            m_pCurText = new WCHAR[rh.recLen/2];
            pStm->Read(m_pCurText, rh.recLen, &nRd);
            wprintf( L"-%s-\n", m_pCurText );
            if( FillBufferWithText() )
               return TRUE;   // Stop parsing if buffer is
full, and return control to client
         }
         else if( rh.recType == PST_TextBytesAtom )
         {
            Assert( m_pCurText == NULL );
            m_curTextPos = 0;
            m_curTextLength = rh.recLen;
            m_pCurText = new WCHAR[rh.recLen];
            pStm->Read(m_pCurText, rh.recLen, &nRd);
            char *pHack = (char *) m_pCurText;
            unsigned int back2 = rh.recLen*2-1;
            unsigned int back1 = rh.recLen-1;
            for(unsigned int i=0;i<rh.recLen;i++)
            {
               pHack[back2-1] = pHack[back1];
               pHack[back2] = 0;
               back2 -=2;
               back1--;
            }
            if( FillBufferWithText() )
               return TRUE;   // Stop parsing if buffer is
full, and return control to client
         }
         else
         {
            LARGE_INTEGER li;
            ULARGE_INTEGER ul;
            li.LowPart = rh.recLen;
            li.HighPart = 0;
            pStm->Seek(li,STREAM_SEEK_CUR,&ul);
         }
         m_pParseContexts->m_nCur += rh.recLen;
         m_pParseContexts->m_nCur += sizeof( RecordHeader ); //
Atom rh's add towards containing container's size.
      }
      else
      {
         m_pParseContexts = new ParseContext( m_pParseContexts
);
         m_pParseContexts->m_rh = rh;
      }
      if( m_pParseContexts->m_nCur >= m_pParseContexts-
>m_rh.recLen )
      {
         Assert(  m_pParseContexts->m_nCur == m_pParseContexts-
>m_rh.recLen );
         ParseContext* pParseContext = m_pParseContexts;
         m_pParseContexts = m_pParseContexts->m_pNext;
         delete pParseContext;
      }
```

```
   } while( m_pParseContexts && ( m_pParseContexts->m_nCur <
m_pParseContexts->m_rh.recLen ) );

   return FALSE;
}

BOOL FileReader::FindNextSlide( DWord& offset )
{
   if( m_curSlideNum == 0 )
   {
      Assert( m_pLastUserEdit != NULL );
      offset = m_pPersistDirectory->GetPersistObjStreamPos(
m_pLastUserEdit->documentRef );
      m_curSlideNum++;
      return TRUE;
   }
   else
   {
      uint4 curSlideNum = m_curSlideNum++;
      SlideListChunk *pCur = m_pFirstChunk;
      while( pCur && ( curSlideNum > pCur->numInChunk ) )
      {
         curSlideNum -= pCur->numInChunk;
         pCur = pCur->pNext;
      }
      if( pCur == NULL )
         return FALSE;
      offset = m_pPersistDirectory->GetPersistObjStreamPos(
pCur->refs[curSlideNum-1] );
      return TRUE;
   }
}

static BOOL ReadText( void** ppContext, IStorage* pStgFrom,
WCHAR* buffer, unsigned long bufferSize, unsigned long*
pSizeRet )
{
   FileReader* pFI = NULL;
   if( *ppContext == NULL )
   {
      pFI = new FileReader( pStgFrom );
      *ppContext = pFI;
      if( !pFI->IsPowerPoint() )
      {
         delete pFI;
         *pSizeRet = 0;
         return FALSE;
      }
      pFI->ReadPersistDirectory();
      pFI->ReadSlideList();
   }
   else
   {
      pFI = (FileReader *)*ppContext;
   }
   BOOL bRet = pFI->ReadText(buffer, bufferSize, pSizeRet);
   if( !bRet )
   {
      delete pFI;
      *ppContext = NULL;
   }
   return bRet;
}
```

```
void main(int argc, char **argv)
{
   OLECHAR wc[256];
   HRESULT hr;
   IStorage *pStgFrom = NULL;

   if (argc < 2)
   {
         fprintf(stderr,"Usage dblock <file to be read>\n");
         exit(0);
   }
   MultiByteToWideChar( CP_ACP, MB_PRECOMPOSED, argv[1], -1,
wc, 255);
   hr = StgOpenStorage(wc, NULL, STGM_READ | STGM_DIRECT |
               STGM_SHARE_DENY_WRITE, NULL, 0, &pStgFrom);
   if (FAILED(hr))
   {
         fprintf(stderr,"Error (%d) opening docfile:
%s\n",(int)hr,argv[1]);
   }
   else
   {
      WCHAR wcBuf[6];
      ULONG sizeUsed;
      BOOL fContinue = TRUE;
      void *pContext = NULL;
      while( fContinue )
      {
         fContinue = ReadText( &pContext, pStgFrom, wcBuf, 5,
&sizeUsed );
         wcBuf[sizeUsed] = 0;
         wprintf(L"-%s-\n", wcBuf);
      }
   }
}
```