[MS-ONESTORE]:

OneNote Revision Store File Format

Revision Summary

Date	Revision History	Revision Class	Comments
7/13/2009	0.1	Major	Initial Availability
8/28/2009	0.2	Editorial	Revised and edited the technical content
11/6/2009	0.3	Editorial	Revised and edited the technical content
2/19/2010	1.0	Major	Updated and revised the technical content
3/31/2010	1.01	Editorial	Revised and edited the technical content
4/30/2010	1.02	Editorial	Revised and edited the technical content
6/7/2010	1.03	Major	Updated and revised the technical content
6/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
7/23/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
9/27/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	None	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	1.04	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	1.04	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	1.5	Minor	Clarified the meaning of the technical content.
4/11/2012	1.5	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	1.6	Minor	Clarified the meaning of the technical content.
10/8/2012	2.0	Major	Significantly changed the technical content.
2/11/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
7/30/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
11/18/2013	2.0	None	No changes to the meaning, language, or formatting of the technical content.
2/10/2014	2.0	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	2.1	Minor	Clarified the meaning of the technical content.
7/31/2014	2.2	Minor	Clarified the meaning of the technical content.
10/30/2014	2.2	None	No changes to the meaning, language, or formatting of the

Date	Revision History	Revision Class	Comments
			technical content.
3/16/2015	3.0	Major	Significantly changed the technical content.
6/30/2015	4.0	Major	Significantly changed the technical content.
9/4/2015	5.0	Major	Significantly changed the technical content.
4/14/2016	6.0	Major	Significantly changed the technical content.
7/15/2016	6.0	None	No changes to the meaning, language, or formatting of the technical content.
9/14/2016	6.0	None	No changes to the meaning, language, or formatting of the technical content.
9/29/2016	6.0	None	No changes to the meaning, language, or formatting of the technical content.
10/17/2016	6.0	None	No changes to the meaning, language, or formatting of the technical content.
9/19/2017	6.1	Minor	Clarified the meaning of the technical content.
6/19/2018	7.0	Major	Significantly changed the technical content.
12/11/2018	7.0	None	No changes to the meaning, language, or formatting of the technical content.
6/18/2019	7.1	Minor	Clarified the meaning of the technical content.
2/19/2020	7.2	Minor	Clarified the meaning of the technical content.
4/22/2021	8.0	Major	Significantly changed the technical content.
6/25/2021	9.0	Major	Significantly changed the technical content.
7/20/2021	10.0	Major	Significantly changed the technical content.
8/17/2021	11.0	Major	Significantly changed the technical content.
10/5/2021	11.0	None	No changes to the meaning, language, or formatting of the technical content.
2/15/2022	11.1	Minor	Clarified the meaning of the technical content.
2/21/2023	11.2	Minor	Clarified the meaning of the technical content.
5/16/2023	11.3	Minor	Clarified the meaning of the technical content.
4/16/2024	12.0	Major	Significantly changed the technical content.
8/20/2024	13.0	Major	Significantly changed the technical content.
11/12/2024	13.1	Minor	Clarified the meaning of the technical content.
2/18/2025	13.2	Minor	Clarified the meaning of the technical content.
5/20/2025	13.3	Minor	Clarified the meaning of the technical content.

Table of Contents

1 Introduction

The OneNote Revision Store File Format (.one and .onetoc2) is a collection of structures that specify a revision store and is organized into cross-referenced object spaces (section 2.1.4) that contain objects (section 2.1.5) with property sets (section 2.1.1) and a transaction log (section 2.3.3) to ensure file integrity across asynchronous writes.

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

1.1 Glossary

This document uses the following terms:

curly braced GUID string: The string representation of a 128-bit globally unique identifier (**GUID**) using the form {XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX}, where X denotes a hexadecimal digit. The string representation between the enclosing braces is the standard representation of a GUID as described in [RFC4122] section 3. Unlike a GUIDString, a curly braced GUID string includes enclosing braces.

cyclic redundancy check (CRC): An algorithm used to produce a checksum (a small, fixed number of bits) against a block of data, such as a packet of network traffic or a block of a computer file. The CRC is a broad class of functions used to detect errors after transmission or storage. A CRC is designed to catch random errors, as opposed to intentional errors. If errors might be introduced by a motivated and intelligent adversary, a cryptographic hash function has to be used instead.

file data object: An object that represents a file that was inserted into a OneNote revision store file. It can be stored internally as a data stream in the revision store file, or externally in the onefiles folder.

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

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

onefiles folder: A folder that stores file data objects for a OneNote revision store file. It is located in the same directory as the revision store file and the folder name maps to the name of the revision store file. For example, if the revision store file is named "section.one" the onefiles folder is named "section_onefiles".

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

universally unique identifier (UUID): A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and RPC objects. UUIDs are highly likely to be unique. UUIDs are also known as **globally unique identifiers (GUIDs)** and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. Specifically, the use of this term does not imply or require that the algorithms described in [RFC4122] or [C706] has to be used for generating the UUID.

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

1.2 References

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

1.2.1 Normative References

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

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

[MS-FSSHTTPB] Microsoft Corporation, "Binary Requests for File Synchronization via SOAP Protocol".

[MS-FSSHTTP] Microsoft Corporation, "File Synchronization via SOAP over HTTP Protocol".

[MS-ONE] Microsoft Corporation, "OneNote File Format".

[MS-OSHARED] Microsoft Corporation, "Office Common Data Types and Objects Structures".

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

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

[RFC3309] Stone, J., Stewart, R., and Otis, D., "Stream Control Transmission Protocol (SCTP) Checksum Change", RFC 3309, September 2002, http://www.rfc-editor.org/rfc/rfc3309.txt

[RFC4122] Leach, P., Mealling, M., and Salz, R., "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, July 2005, https://www.rfc-editor.org/info/rfc4122

1.2.2 Informative References

None.

1.3 Overview

This file format is a revision-based file format created to be an effective way to store changes with revisions instead of needing to rewrite the entire file whenever a change is written to the file. Additionally, the revision store is transactional to ensure data integrity as clients read and write data to the revision store. The revision store is used for .one and .onetoc2 files.

1.3.1 File Structure

A revision store file is divided into the structures in the following diagram.

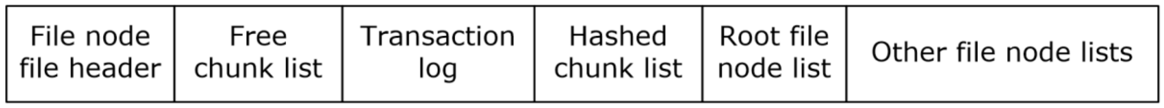


Figure 1: File structure

The header (section 2.3.1) is the first 1024 bytes of the file. It contains references to the other structures in the file as well as metadata about the file.

The free chunk list (section 2.3.2) defines where there are free spaces in the file where data can be written.

The transaction log (section 2.3.3) stores the state and length of each file node list (section 2.4) in the file.

The hashed chunk list (section 2.3.4) stores read-only objects in the file that can be referenced by multiple revisions (section 2.1.8).

The root file node list (section 2.1.14) is the file node list that is the root of the tree of all file node lists in the file.

All of the file node lists that contain user data.

1.3.2 File Node Lists

File node lists are the building blocks organizing all of the data in the file. There are multiple file node lists that form a tree hierarchy beginning with the root file node list (section 2.1.14), as shown in the following diagram.

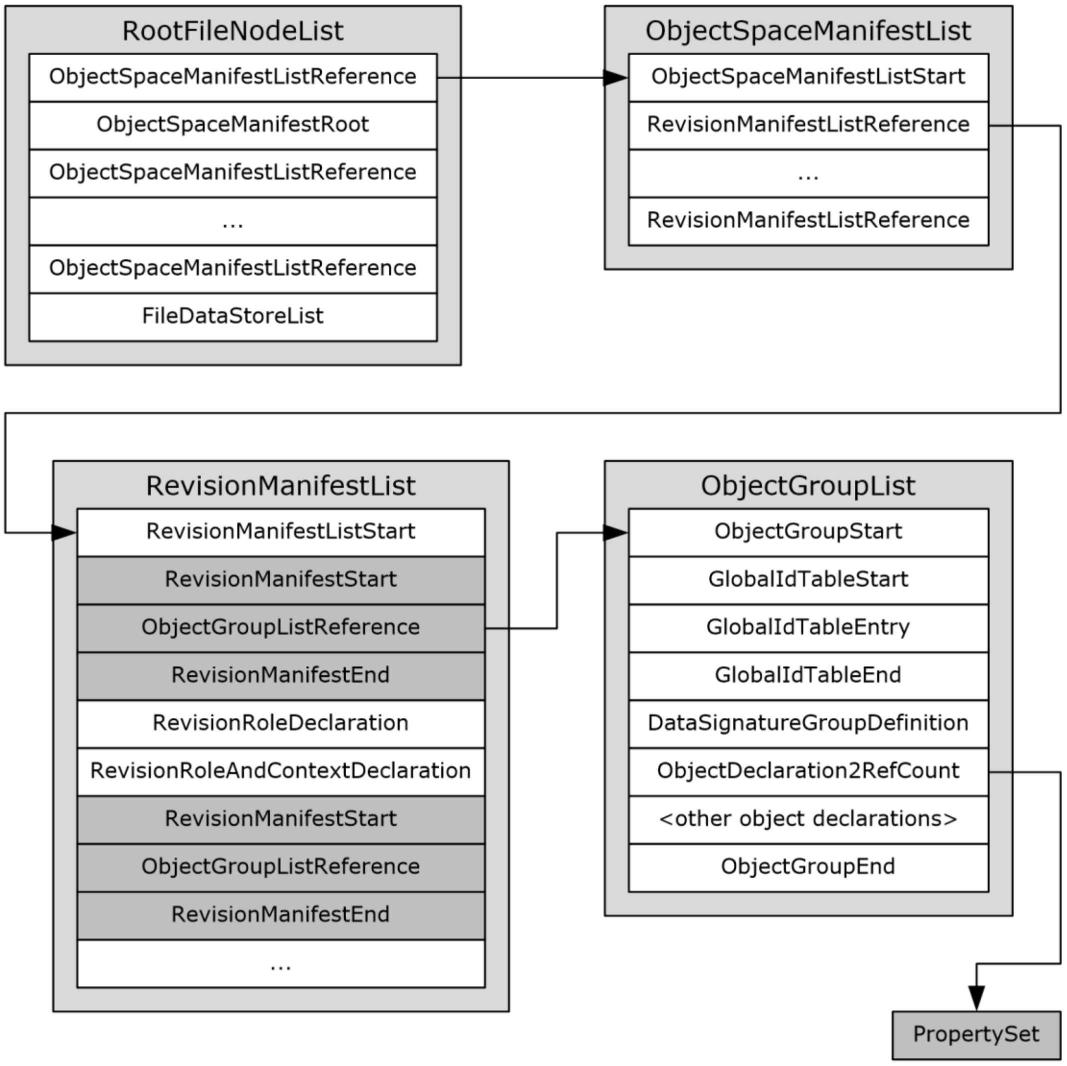


Figure 2: File node list structure

The root file node list enumerates all of the object spaces (section 2.1.4) in the revision store, and the file data store list, if present. It also identifies the root object spaces, as shown in the following diagram.

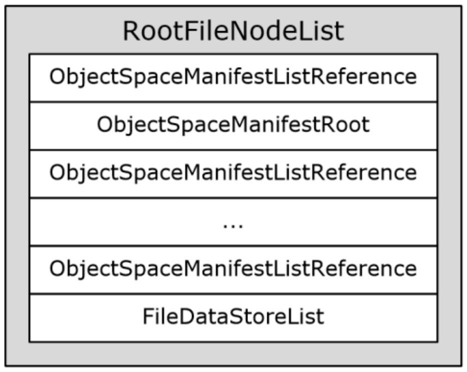

RootFileNodeList
ObjectSpaceManifestListReference
ObjectSpaceManifestRoot
ObjectSpaceManifestListReference
...
ObjectSpaceManifestListReference
FileDataStoreList

Figure 3: Root file node list

1.3.3 Object Space Manifest List

An object space manifest list (section 2.1.6) references the set of revisions that make up an object space (section 2.1.4). An object space is a collection of objects (section 2.1.5) and their properties, as shown in the following diagram.

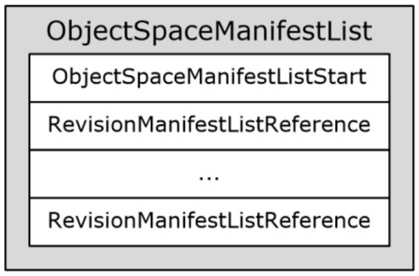

ObjectSpaceManifestList
ObjectSpaceManifestListStart
RevisionManifestListReference
...
RevisionManifestListReference

Figure 4: Object space manifest list

1.3.4 Revision Manifest List

A revision store tracks the state of an object space over time. A revision (section 2.1.8) is a snapshot of the state of an object space at a specific point in time. A revision manifest (section 2.1.9) defines a single revision (section 2.1.8), either as a full set of objects (section 2.1.5) or as a set of changes from another revision (section 2.1.8). A revision manifest list (section 2.1.10) is the collection of all revisions (section 2.1.8) that have been saved for this object space.

In the following figure each of the shaded sequences of boxes represents a revision manifest.

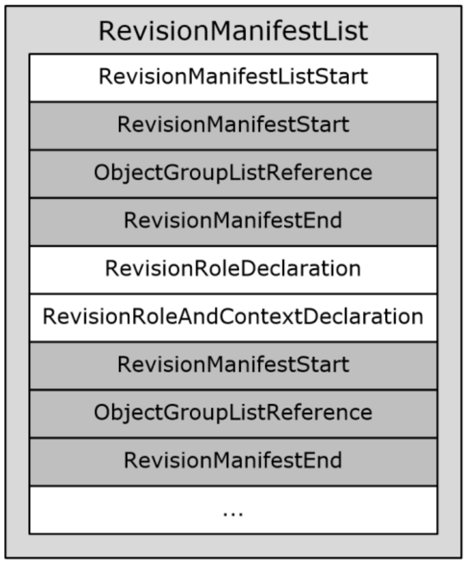


Figure 5: Revision manifest list

1.3.5 Object Group

An object group (section 2.1.13) enumerates a set of objects (section 2.1.5), each of which has an identity and optionally a property set (section 2.1.1), as shown in the following diagram.

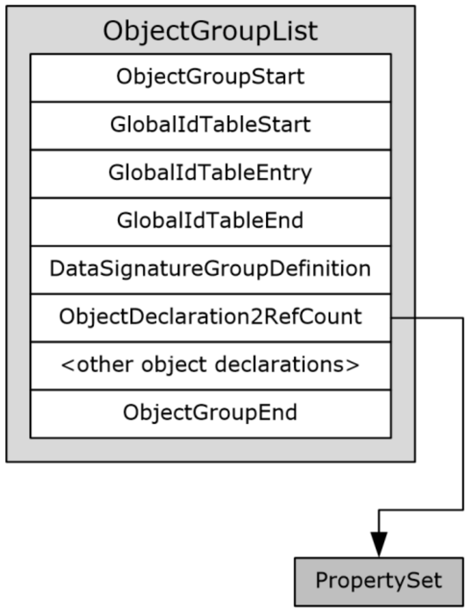


Figure 6: Object group

1.3.6 Transaction Log

The transaction log is used to keep track of each file node list (section 2.4) and how much of the list ought to be read. Each file node list can continue for any length, but the transaction log defines the active length. Content past the end represents incomplete transactions. A complete transaction begins with data appended to the file node lists, followed by new entries in the transaction log to reflect the updated state of the file node lists. The final operation for the transaction is to update the transaction count in the header to activate the new transaction entries.

1.4 Relationship to Protocols and Other Structures

This file format is dependent on the structures defined in the following references:

- [MS-ONE] for specific object type (section 2.6.14) and property identifier (section 2.6.6) values.
- [MS-OSHARED] for the algorithm to compute a **cyclic redundancy check (CRC)** (section 2.1.2).
- [MS-DTYP] for the persistence format for **GUIDs**.
- [MS-FSSHTTP] for data transmission protocol.
- [MS-FSSHTTPB] for transmission protocol data structures and types.

1.5 Applicability Statement

This document specifies a persistence format for a revision store organized into cross-referenced object spaces (section 2.1.4), containing objects (section 2.1.5) with property sets (section 2.1.1), and containing a transaction log (section 2.3.3) to ensure file integrity across asynchronous writes. This persistence format is applicable when the primary presentation format for the contained information is electronic.

This persistence format is applicable for use as a stand-alone document, and for transmission via the File Synchronization via SOAP over HTTP Protocol as described in [MS-FSSHTTP] and [MS-FSSHTTPB].

This persistence format provides interoperability with applications that create or read documents conforming to this structure<1>.

1.6 Versioning and Localization

This document covers versioning issues in the following areas:

Structure versions: The revision store supports the introduction of new **FileNode** structure (section 2.4.3) types, **Object** types (section 2.6.14), and **PropertyID**s (section 2.6.6) by future implementers or this specification. Programs that implement this structure specification that encounter data that identifies itself as a type that is unknown to the application will ignore that data and leave it unchanged when persisting the file again.

If the value of the **Header.ffvOldestCodeThatMayReadThisFile** field is greater than 0x0000002A, programs that implement this structure specification ignore all other data in the file.

Localization: This structure defines no general locale-specific processes or data.

1.7 Vendor-Extensible Fields

None.

2 Structures

2.1 Fundamental Concepts

2.1.1 Property Set

A property set is a collection of properties that specify the attributes of an object (section 2.1.5). The **PropertySet** structure specifies the format of a property set and is contained by an **ObjectSpaceObjectPropSet** structure (section 2.6.1). The meaning of each property in the set is specified in [MS-ONE] section 2.1.12. A **PropertySet** structure can contain references to other objects.

The data for a property that is not an object reference is contained in the **PropertySet.rgData** stream field. The **rgData** stream is read sequentially beginning with the first property in a **PropertySet.rgPrids** array until every property has been read. The number of bytes read for each property is specified by the **PropertyID.type** field.

The data for a property that is a reference to one or more objects (section 2.1.5) is contained in the streams within an **ObjectSpaceObjectPropSet** structure (**OIDs.body**, **OSIDs.body**, **ContextIDs.body**). The streams are read sequentially beginning with the first property in a **PropertySet.rgPrids** array. If the **PropertyID.type** field specifies a single object (0x8, 0xA, 0xC), a single **CompactID** (4 bytes) is read from the corresponding stream in the **ObjectSpaceObjectPropSet** structure. If the **PropertyID.type** field specifies an array of objects (0x9, 0xB, 0xD), an unsigned integer (4 bytes) is read from the **PropertySet.rgData** stream and specifies the number of **CompactID** structures (section 2.2.2) to read from the corresponding stream in the **ObjectSpaceObjectPropSet** structure. The streams for each **PropertyID.type** field are given by the following table.

Value	Stream
0x8 (ObjectID, section 2.6.6)	ObjectSpaceObjectPropSet.OIDs.body
0x9 (ArrayOfObjectIDs, section 2.6.6)	ObjectSpaceObjectPropSet.OIDs.body
0xA (ObjectSpaceID, section 2.6.6)	ObjectSpaceObjectPropSet.OSIDs.body
0xB (ArrayOfObjectSpaceIDs, section 2.6.6)	ObjectSpaceObjectPropSet.OSIDs.body
0xC (ContextID, section 2.6.6)	ObjectSpaceObjectPropSet.ContextIDs.body
0xD (ArrayOfContextIDs, section 2.6.6)	ObjectSpaceObjectPropSet.ContextIDs.body

2.1.2 Cyclic Redundancy Check (CRC) Algorithms

A revision store file contains **cyclic redundancy check (CRC)** values that are used to ensure the integrity of the file. The algorithm used is given by the type of the revision store file.

File format	Algorithm
.one	The CRC is calculated using the algorithm specified by [RFC3309]. The CRC polynomial is: $x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x^{1}+1$ Normal representation for the polynomial is 0x04C11DB7. For the purpose of ordering, the least significant bit of the 32-bit CRC is defined to be

File format	Algorithm
	the coefficient of the x^{31} term. The 32-bit CRC register is initialized to all 1's and once the data is processed, the CRC register is inverted. (1's complement.)
.onetoc2	The CRC is calculated using the algorithm specified by MsoCrc32Compute, as specified in [MS-OSHARED] section 2.4.3.2.

2.1.3 Global Identification Table

A global identification table is an optimization mechanism for compacting a set of **ExtendedGUID** structures (section 2.2.1). Each unique **ExtendedGUID.guid** field in the set is added to the global identification table with a unique index. **CompactID** structures (section 2.2.2) are used after that to represent the **ExtendedGUID** structure by retrieving the value of the **ExtendedGUID.guid** field from the table.

The global identification table MUST have the structure described in the following table.

File format	Valid structure
.one	▪ A **FileNode** structure (section 2.4.3) with a **FileNodeID** field value equal to 0x022 (**GlobalIdTableStart2FND** structure, section 2.4.3). ▪ Zero or more **FileNode** structures with **FileNodeID** field value equal to 0x024 (**GlobalIdTableEntryFNDX** structure, section 2.5.10). ▪ A **FileNode** structure with a **FileNodeID** field value equal to 0x028 (**GlobalIdTableEndFNDX** structure, section 2.4.3).
.onetoc2	▪ A **FileNode** structure (section 2.4.3) with a **FileNodeID** field value equal to 0x021 (**GlobalIdTableStartFNDX** structure, section 2.5.9). ▪ Zero or more **FileNode** structures with **FileNodeID** field values equal to one of the following values: ▪ 0x024 (**GlobalIdTableEntryFNDX** structure, section 2.5.10) ▪ 0x025 (**GlobalIdTableEntry2FNDX** structure, section 2.5.11) ▪ 0x026 (**GlobalIdTableEntry3FNDX** structure, section 2.5.12) ▪ A **FileNode** structure with a **FileNodeID** field value equal to 0x028 (**GlobalIdTableEndFNDX** structure, section 2.4.3).

A global identification table applies to all **FileNode** structures specified immediately following the table until a **FileNode** structure with one of the following **FileNodeID** field values is encountered:

- 0x01C (**RevisionManifestEndFND** structure, section 2.4.3)
- 0x021 (**GlobalIdTableStartFNDX** structure, section 2.5.9)
- 0x022 (**GlobalIdTableStart2FND** structure, section 2.4.3)

2.1.4 Object Space

An object space is a collection of objects (section 2.1.5). The objects in an object space can be revised independently of objects in other object spaces.

Object spaces MUST be referenced from the root file node list (section 2.1.14) by a **FileNode** structure (section 2.4.3) with a **FileNodeID** field value equal to 0x08 (**ObjectSpaceManifestListReferenceFND** structure, section 2.5.2). Object spaces MUST have a unique identifier (OSID), specified by the **ObjectSpaceManifestListReferenceFND**.**gosid** field. Every revision store file MUST have exactly one root object space whose OSID is specified by the **ObjectSpaceManifestRootFND**.**gosidRoot** field.

The content of an object space is specified by the **ObjectSpaceManifestList** structure (section 2.1.6).

2.1.5 Object Space Object

An object space object (object) is a collection of data. Every object's identity is specified by an **ExtendedGUID** structure (section 2.2.1).

When an object with the same identity is declared multiple times, the later declarations are said to revise the object. The following **FileNode** structures can declare or revise an object:

- 0x041 (**ObjectRevisionWithRefCountFNDX** structure, section 2.5.13)
- 0x042 (**ObjectRevisionWithRefCount2FNDX** structure, section 2.5.14)
- 0x02D (**ObjectDeclarationWithRefCountFNDX** structure, section 2.5.23)
- 0x02E (**ObjectDeclarationWithRefCount2FNDX** structure, section 2.5.24)
- 0x0A4 (**ObjectDeclaration2RefCountFND** structure, section 2.5.25)
- 0x0A5 (**ObjectDeclaration2LargeRefCountFND** structure, section 2.5.26)
- 0x0C4 (**ReadOnlyObjectDeclaration2RefCountFND** structure, section 2.5.29)
- 0x0C5 (**ReadOnlyObjectDeclaration2LargeRefCountFND** structure, section 2.5.30)
- 0x072 (**ObjectDeclarationFileData3RefCountFND** structure, section 2.5.27)
- 0x073 (**ObjectDeclarationFileData3LargeRefCountFND** structure, section 2.5.28)

The identity and data for an object is specified by a **FileNode** structure (section 2.4.3). The data specified by the **FileNode.fnd** field MUST contain a **JCID** (section 2.6.14) or an unsigned integer that specifies the value of **JCID.index**. When only index is specified, the other fields of **JCID** MUST be implied as set to: `JCID.IsBinary = "false"`, `JCID.IsPropertySet = "true"`, `JCID.IsGraphNode = "false"`, `JCID.IsFileData = "false"`, and `JCID.IsReadOnly = "false"`.

The meaning of the **JCID** structure is specified by the property set (as defined in [MS-ONE] section 2.1.13) for objects whose data is a property set (section 2.1.1), or file data object (as defined in [MS-ONE] section 2.1.5) for objects whose data is a **file data object**. The **JCID** structure of an object MUST NOT be changed when the object is revised.

If the value of the **JCID.IsPropertySet** field is "true" or if only JCID.index is specified, then the data for the **Object Space Object** structure MUST be an **ObjectSpaceObjectPropSet** structure (section 2.6.1), and the value of the **FileNode.FileNodeID** field MUST be one of the following values:

- 0x02D (**ObjectDeclarationWithRefCountFNDX** structure, section 2.5.23)
- 0x02E (**ObjectDeclarationWithRefCount2FNDX** structure, section 2.5.24)
- 0x0A4 (**ObjectDeclaration2RefCountFND** structure, section 2.5.25)
- 0x0A5 (**ObjectDeclaration2LargeRefCountFND** structure, section 2.5.26)

- 0x0C4 (**ReadOnlyObjectDeclaration2RefCountFND** structure, section 2.5.29)
- 0x0C5 (**ReadOnlyObjectDeclaration2LargeRefCountFND** structure, section 2.5.30)

Objects whose data is an **ObjectSpaceObjectPropSet** structure can contain references to the following:

- Objects in the same object space (section 2.1.4), specified by the **ObjectSpaceObjectPropSet.OIDs** field. Object references MUST NOT form a cycle.
- Object spaces specified by the **ObjectSpaceObjectPropSet.OSIDs** field. Objects MUST NOT reference the object space that contains them. Object space references specifically reference the revision (section 2.1.8) of the object space that is associated with the default context and revision role 0x00000001. Object space references MUST NOT form a cycle.
- Contexts (section 2.1.11) of the same object space, specified by the **ObjectSpaceObjectPropSet.ContextIDs** field. Context references specifically reference the revision that is associated with the identified context and revision role 0x00000001. Context references can form a cycle.

If the value of the **JCID.IsReadOnly** field is "true" then the value of the **FileNode.FileNodeID** field MUST be 0x0C4 (**ReadOnlyObjectDeclaration2RefCountFND** structure, section 2.5.29) or 0x0C5 (**ReadOnlyObjectDeclaration2LargeRefCountFND** structure, section 2.5.30). If an object with **JCID.IsReadOnly** set to "true" is revised, all declarations of this object MUST specify identical data.

If the value of the **JCID.IsFileData** field is "true" then the value of the **FileNode.FileNodeID** field MUST be 0x072 (**ObjectDeclarationFileData3RefCountFND** structure, section 2.5.27) or 0x073 (**ObjectDeclarationFileData3LargeRefCountFND** structure, section 2.5.28). If an object with **JCID.IsFileData** set to "true" is revised, all declarations of this object MUST specify identical data.

All objects in a revision contain a reference count. The reference count for any object that is not a root object (section 2.1.7) is the number of objects that directly reference this object, where the referencing object is reachable from a root object within the revision or is itself a root object. The reference count for a root object is the count calculated by the previous algorithm for non-root objects, plus 1.

The current reference count of an object in a revision is specified by the last **FileNode** structure that revised the object or by a **FileNode** structure with **FileNodeID** equal to 0x84, (**ObjectInfoDependencyOverridesFND**, section 2.5.20) that updates the reference count without revising the object.

2.1.6 Object Space Manifest List

An object space manifest list is a file node list (section 2.4) that specifies the revision manifest lists (section 2.1.10) that exist for the object space (section 2.1.4).

An object space manifest list MUST consist of the following sequence of **FileNode** structures (section 2.4.3):

1. A **FileNode** structure with **FileNodeID** field value equal to 0x00C (**ObjectSpaceManifestListStartFND** structure, section 2.5.3).
2. One or more **FileNode** structures with **FileNodeID** field values equal to 0x010 (**RevisionManifestListReferenceFND** structure, section 2.5.4). If more than one is present, all but the last MUST be ignored.

2.1.7 Root Object

A root object is an object (section 2.1.5) within an object space (section 2.1.4) that is directly referenced by a revision (section 2.1.8). All other objects within the revision MUST be reachable from a root object via object references.

A root object is specified for a revision by either of the following **FileNode** structures (section 2.4.3) within a revision manifest:

- **FileNode** structures with **FileNodeID** field values equal to 0x059 (**RootObjectReference2FNDX** structure, section 2.5.15)
- **FileNode** structures with **FileNodeID** field values equal to 0x05A (**RootObjectReference3FND** structure, section 2.5.16)

A revision can specify more than one root object. Different root objects in a revision MUST specify different values for **RootRole** ([MS-ONE] section 2.1.8). A root object MUST NOT specify more than one **RootRole**.

2.1.8 Revision

A revision specifies the state of an object space (section 2.1.4), which is comprised of a choice of root objects (section 2.1.7), a set of objects (section 2.1.5) reachable from the root objects via object references, and the state of each object.

A revision is identified by an **ExtendedGUID** structure (section 2.2.1). A revision is immutable. Revisions that specify the same **ExtendedGUID** structure identity MUST resolve to an identical set of objects, object states, and root object choices.

2.1.9 Revision Manifest

A revision manifest specifies a set of objects (section 2.1.5), the object states, the choice of root objects (section 2.1.7) for an object space (section 2.1.4), and an identity. Together, these resolve to a revision (section 2.1.8). The revision manifest can also specify the identity of another revision as a dependency. When a dependency revision is specified, the objects, object states, and root object of the dependency revision are also included in this revision, except where they are specified by this revision manifest.

A revision manifest is specified by a sequence of **FileNode** structures (section 2.4.3) within a revision manifest list (section 2.1.10).

The sequence MUST begin with one of the **FileNode** structures described in the following table.

File format	Valid structures
.one	0x01E (**RevisionManifestStart6FND** structure, section 2.5.7) 0x01F (**RevisionManifestStart7FND** structure, section 2.5.8)
.onetoc2	0x01B (**RevisionManifestStart4FND** structure, section 2.5.6)

If the object space is encrypted, then the second **FileNode** in the sequence MUST be a **FileNode** structure with a **FileNodeID** equal to 0x07C (**ObjectDataEncryptionKeyV2FNDX** structure, section 2.5.19).

The remainder of the sequence can contain the **FileNode** structures described in the following table, and MUST NOT contain any other **FileNode** structures.

File format	Valid contents
.one	Zero or more sequences of object group **FileNode** structures: ▪ 0x0B0 (**ObjectGroupListReferenceFND** structure, section 2.5.31) ▪ 0x084 (**ObjectInfoDependencyOverridesFND** structure, section 2.5.20) Where each **ObjectGroupListReferenceFND** structure MUST be followed by an **ObjectInfoDependencyOverridesFND** structure. Zero or one sequence of global identification table **FileNode** structures: ▪ A **FileNode** structure with a **FileNodeID** field value equal to 0x022 (**GlobalIdTableStart2FND** structure, section 2.4.3) ▪ Zero or more **FileNode** structures with a **FileNodeID** field value equal to 0x024 (**GlobalIdTableEntryFNDX** structure, section 2.5.10) ▪ A **FileNode** structure with a **FileNodeID** field value equal to 0x028 (**GlobalIdTableEndFNDX** structure, section 2.4.3) Where the global identification table sequence of **FileNode** structures MUST NOT be followed by any object group sequences of **FileNode** structures. Zero or more **FileNode** structures with **FileNodeID** field values equal to any of: ▪ 0x05A (**RootObjectReference3FND** structure section 2.5.16) ▪ 0x084 (**ObjectInfoDependencyOverridesFND** structure, section 2.5.20)
.onetoc2	Zero or more **FileNode** structures with **FileNodeID** field values equal to any of the following: ▪ 0x059 (**RootObjectReference2FNDX** structure, section 2.5.15) ▪ 0x084 (**ObjectInfoDependencyOverridesFND** structure) Zero or one sequence of global identification table **FileNode** structures: ▪ A **FileNode** structure with a **FileNodeID** field value equal to 0x021 (**GlobalIdTableStartFNDX** structure, section 2.5.9) ▪ Zero or more **FileNode** structures with **FileNodeID** field values equal to one of: ▪ 0x024 (**GlobalIdTableEntryFNDX** structure, section 2.5.10) ▪ 0x025 (**GlobalIdTableEntry2FNDX** structure, section 2.5.11) ▪ 0x026 (**GlobalIdTableEntry3FNDX** structure, section 2.5.12) ▪ A **FileNode** structure with a **FileNodeID** field value equal to 0x028 (**GlobalIdTableEndFNDX** structure, section 2.4.3) Zero or more **FileNode** structures with **FileNodeID** field values equal to the following: ▪ 0x08C (**DataSignatureGroupDefinitionFND** structure, section 2.5.33) ▪ 0x02D (**ObjectDeclarationWithRefCountFNDX** structure, section 2.5.23) ▪ 0x02E (**ObjectDeclarationWithRefCount2FNDX** structure, section 2.5.24) ▪ 0x041 (**ObjectRevisionWithRefCountFNDX** structure, section 2.5.13) ▪ 0x042 (**ObjectRevisionWithRefCount2FNDX** structure, section 2.5.14) that MUST follow a global identification table sequence.

The sequence MUST end with a **FileNode** structure with a **FileNodeID** field value equal to 0x01C (**RevisionManifestEndFND** structure, section 2.4.3).

2.1.10 Revision Manifest List

A revision manifest list is a file node list (section 2.4) that specifies the revisions (section 2.1.8) of an object space (section 2.1.4), and the revision roles (section 2.1.12) and contexts (section 2.1.11) that label those revisions.

All of the revision manifests (section 2.1.9) for an object space MUST appear in a single revision manifest list. An object space manifest list (section 2.1.6) can specify multiple revision manifest lists. In that case, all but the last revision manifest list referenced by the object space manifest list MUST be ignored.

The first **FileNode** structure (section 2.4.3) in a revision manifest list MUST have the **FileNodeID** field value equal to 0x14 (**RevisionManifestListStartFND** structure, section 2.5.5). The remainder of the revision manifest list MUST contain zero or more of the following structures, and MUST NOT contain any others:

- Revision manifests (section 2.1.9).
- **FileNode** structures with a **FileNodeID** field value equal to 0x5C (**RevisionRoleDeclarationFND** structure, section 2.5.17).
- **FileNode** structures with a **FileNodeID** field value equal to 0x5D (**RevisionRoleAndContextDeclarationFND** structure, section 2.5.18).

2.1.11 Context

A context is a label for a revision (section 2.1.8) of an object space (section 2.1.4). It is specified by an **ExtendedGUID** (section 2.2.1). The context with **ExtendedGUID** equal to {{00000000-0000-0000-0000-000000000000}, 0} is called the default context of the object space.

The following **FileNode** structures (section 2.4.3) associate a context and revision role (section 2.1.12) label pair with a specific revision:

- **FileNode** structures with **FileNodeID** field values equal to 0x01B (**RevisionManifestStart4FND** structure, section 2.5.6).
- **FileNode** structures with **FileNodeID** field values equal to 0x01E (**RevisionManifestStart6FND** structure, section 2.5.7).
- **FileNode** structures with **FileNodeID** field values equal to 0x01F (**RevisionManifestStart7FND** structure, section 2.5.8).
- **FileNode** structures with **FileNodeID** field values equal to 0x05C (**RevisionRoleDeclarationFND** structure, section 2.5.17).
- **FileNode** structures with **FileNodeID** field values equal to 0x05D (**RevisionRoleAndContextDeclarationFND** structure, section 2.5.18).

FileNode structures that specify a revision role but not a context use the default context in their label pair. When a **FileNode** structure associates a context and revision role label pair with a revision, that revision is the current revision of that context and revision role label pair. All associations made by **FileNode** structures earlier in the revision manifest list (section 2.1.10) for the same label pair MUST be ignored.

2.1.12 Revision Role

A revision role is a label for a revision (section 2.1.8) of an object space (section 2.1.4). It is specified by a 4-byte integer where the high 2 bytes MUST be set to zero. Revision role SHOULD<2> be 0x00000001, which specifies that the revision applies to the active view of the current object space.

The following **FileNode** structures (section 2.4.3) associate a context (section 2.1.11) and revision role label pair with a specific revision:

- **FileNode** structures with **FileNodeID** field values equal to 0x01B (**RevisionManifestStart4FND** structure, section 2.5.6).
- **FileNode** structures with **FileNodeID** field values equal to 0x01E (**RevisionManifestStart6FND** structure, section 2.5.7).
- **FileNode** structures with **FileNodeID** field values equal to 0x01F (**RevisionManifestStart7FND** structure, section 2.5.8).
- **FileNode** structures with **FileNodeID** field values equal to 0x05C (**RevisionRoleDeclarationFND** structure, section 2.5.17).
- **FileNode** structures with **FileNodeID** field values equal to 0x05D (**RevisionRoleAndContextDeclarationFND** structure, section 2.5.18).

When a **FileNode** structure associates a context and revision role label pair with a revision, all associations made by **FileNode** structures earlier in the revision manifest list (section 2.1.10) for the same label pair MUST be ignored.

2.1.13 Object Group

An object group specifies a subset of objects in a revision manifest (section 2.1.9). An object group MUST NOT be referenced by more than one revision manifest.

An object group MUST be contained within a single file node list (section 2.4). This file node list MUST have the following structure:

- **FileNode** structure (section 2.4.3) with a **FileNodeID** field value equal to 0x0B4 (**ObjectGroupStartFND** structure, section 2.5.32).
- Global identification table (section 2.1.3).
- Zero or more **FileNode** structures with any of the following **FileNodeID** field values:
    - 0x08C (**DataSignatureGroupDefinitionFND** structure, section 2.5.33).
    - 0x0A4 (**ObjectDeclaration2RefCountFND** structure, section 2.5.25).
    - 0x0A5 (**ObjectDeclaration2LargeRefCountFND** structure, section 2.5.26).
    - 0x0C4 (**ReadOnlyObjectDeclaration2RefCountFND** structure, section 2.5.29).
    - 0x0C5 (**ReadOnlyObjectDeclaration2LargeRefCountFND** structure, section 2.5.30).
    - 0x072 (**ObjectDeclarationFileData3RefCountFND** structure, section 2.5.27).
    - 0x073 (**ObjectDeclarationFileData3LargeRefCountFND** structure, section 2.5.28).
- **FileNode** structure with a **FileNodeID** field value equal to 0x0B8 (**ObjectGroupEndFND** structure, section 2.4.3).

2.1.14 Root File Node List

The root file node list is a file node list (section 2.4) that specifies the set of all object spaces (section 2.1.4) contained in this file. It also specifies which object space is the root.

The root file node list MUST begin with the **FileNodeListFragment** structure (section 2.4.1) specified by the **Header.fcrFileNodeListRoot** field (section 2.3.1).

The root file node list MUST consist of the following **FileNode** structures (section 2.4.3), and MUST NOT contain any others:

- One or more **FileNode** structures with **FileNodeID** field values equal to 0x008 (**ObjectSpaceManifestListReferenceFND** structure, section 2.5.2).
- One **FileNode** structure with a **FileNodeID** field value equal to 0x004 (**ObjectSpaceManifestRootFND** structure, section 2.5.1).
- Zero or one **FileNode** structure with **FileNodeID** field values equal to 0x090 (**FileDataStoreListReferenceFND** structure, section 2.5.21).

2.2 Common Types

2.2.1 ExtendedGUID

The **ExtendedGUID** structure is a combination of a **GUID**, as specified by [MS-DTYP], and an unsigned integer. Two **ExtendedGUID** structures specify the same identifier if the values of their **guid** fields are the same and the values of their **n** fields are the same.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
guid (16 bytes)																															
...																															
...																															
n																															

guid (16 bytes): Specifies a GUID, as specified by [MS-DTYP].

n (4 bytes): An unsigned integer that MUST be zero when the **guid** field value is {00000000-0000-0000-0000-000000000000}.

2.2.2 CompactID

If you are reading a revision store file encoded using the File Synchronization via SOAP Over HTTP Protocol, refer to the section 2.7.8. Otherwise, continue reading the rest of this section.

The **CompactID** structure is a combination of two unsigned integers. A **CompactID** structure together with a global identification table (section 2.1.3) specifies an **ExtendedGUID** structure (section 2.2.1).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
n								guidIndex																							

n (8 bits): An unsigned integer that specifies the value of the **ExtendedGUID.n** field.

guidIndex (24 bits): An unsigned integer that specifies the index in the global identification table. The **GUID** that corresponds to this index provides the value for the **ExtendedGUID.guid** field.

2.2.3 StringInStorageBuffer

The **StringInStorageBuffer** structure is a variable-length **Unicode** string.

This structure has the following format.

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

cch (4 bytes): An unsigned integer that specifies the number of characters in the string.

StringData (variable): An array of UTF-16 Unicode characters. The length of the array MUST be equal to the value specified by the **cch** field.

2.2.4 File Chunk Reference

A file chunk reference specifies a reference to data in the file. Each file chunk reference contains an **stp** field and a **cb** field. The **stp** field is a stream pointer that specifies the offset, in bytes, from the beginning of the file where the referenced data is located. The **cb** field specifies the size, in bytes, of the referenced data. The sizes, in bytes, of the **stp** and **cb** fields are specified by the structures in this section.

Special values:

fcrNil: Specifies a file chunk reference where all bits of the **stp** field are set to 1, and all bits of the **cb** field are set to zero.

fcrZero: Specifies a file chunk reference where all bits of the **stp** and **cb** fields are set to zero.

2.2.4.1 FileChunkReference32

A **FileChunkReference32** structure is a file chunk reference (section 2.2.4) where both the **stp** field and the **cb** field are 4 bytes in size.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
stp																															
cb																															

stp (4 bytes): An unsigned integer that specifies the location of the referenced data in the file.

cb (4 bytes): An unsigned integer that specifies the size, in bytes, of the referenced data.

2.2.4.2 FileNodeChunkReference

A **FileNodeChunkReference** structure is a file chunk reference (section 2.2.4) that specifies the location in the file and the size of data referenced by a **FileNode** structure (section 2.4.3). The size of the file chunk reference (section 2.2.4) is specified by the **FileNode.StpFormat** and **FileNode.CbFormat** fields of the **FileNode** structure that contains the **FileNodeChunkReference** structure. The meaning of the referenced data is specified by the **FileNode** structure that contains the **FileNodeChunkReference** structure.

This structure has the following format.

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

stp (variable): An unsigned integer that specifies the location of the referenced data in the file. The size and meaning of the **stp** field is specified by the value of the **FileNode.StpFormat** field.

cb (variable): An unsigned integer that specifies the size, in bytes, of the data. The size and meaning of the **cb** field is specified by the value of **FileNode.CbFormat** field.

2.2.4.3 FileChunkReference64

A **FileChunkReference64** structure is a file chunk reference (section 2.2.4) where both the **stp** field and the **cb** field are 8 bytes in size.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
stp																															
...																															
cb																															

...

stp (8 bytes): An unsigned integer that specifies the location of the referenced data in the file.

cb (8 bytes): An unsigned integer that specifies the size, in bytes, of the referenced data.

2.2.4.4 FileChunkReference64x32

A **FileChunkReference64x32** structure is a file chunk reference (section 2.2.4) where the **stp** field is 8 bytes in size and the **cb** field is 4 bytes in size.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
stp																															
...																															
cb																															

stp (8 bytes): An unsigned integer that specifies the location of the referenced data in the file.

cb (4 bytes): An unsigned integer that specifies the size, in bytes, of the referenced data.

2.3 File Structure

A revision store file MUST begin with a **Header** structure (section 2.3.1). The remainder of the file is partitioned into blocks of bytes, where the size and structure of each block is specified by the field that references it. A block is reachable if it is referenced by the **Header** structure, or if it is referenced by a field in another reachable block. Data outside the **Header** structure and any reachable blocks MUST be ignored.

All structures are aligned on 1-byte boundaries. All integers are signed unless otherwise specified. All fields are **little-endian** unless otherwise specified.

2.3.1 Header

The **Header** structure MUST be at the beginning of the file.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
guidFileType (16 bytes)																															
...																															
...																															
guidFile (16 bytes)																															

...			
...			
guidLegacyFileVersion (16 bytes)			
...			
...			
guidFileFormat (16 bytes)			
...			
...			
ffvLastCodeThatWroteToThisFile			
ffvOldestCodeThatHasWrittenToThisFile			
ffvNewestCodeThatHasWrittenToThisFile			
ffvOldestCodeThatMayReadThisFile			
fcrLegacyFreeChunkList			
...			
fcrLegacyTransactionLog			
...			
cTransactionsInLog			
cbLegacyExpectedFileLength			
rgbPlaceholder			
...			
fcrLegacyFileNodeListRoot			
...			
cbLegacyFreeSpaceInFreeChunkList			
fNeedsDefrag	fRepairedFile	fNeedsGarbageCollect	fHasNoEmbeddedFileObje
guidAncestor (16 bytes)			

...
...
crcName
fcrHashedChunkList
...
...
fcrTransactionLog
...
...
fcrFileNodeListRoot
...
...
fcrFreeChunkList
...
...
cbExpectedFileLength
...
cbFreeSpaceInFreeChunkList
...
guidFileVersion (16 bytes)
...
...
nFileVersionGeneration
...
guidDenyReadFileVersion (16 bytes)

...
...
grfDebugLogFlags
fcrDebugLog
...
...
fcrAllocVerificationFreeChunkList
...
...
bnCreated
bnLastWroteToThisFile
bnOldestWritten
bnNewestWritten
rgbReserved (728 bytes)
...
...

guidFileType (16 bytes): A **GUID**, as specified by [MS-DTYP], that specifies the type of the revision store file. MUST be one of the values from the following table.

File format	Value
.one	{7B5C52E4-D88C-4DA7-AEB1-5378D02996D3}
.onetoc2	{43FF2FA1-EFD9-4C76-9EE2-10EA5722765F}

guidFile (16 bytes): A GUID, as specified by [MS-DTYP], that specifies the identity of this revision store file. SHOULD be globally unique.

guidLegacyFileVersion (16 bytes): MUST be "{00000000-0000-0000-0000-000000000000}" and MUST be ignored.

guidFileFormat (16 bytes): A GUID, as specified by [MS-DTYP], that specifies that the file is a revision store file. MUST be "{109ADD3F-911B-49F5-A5D0-1791EDC8AED8}".

ffvLastCodeThatWroteToThisFile (4 bytes): An unsigned integer. MUST be one of the values in the following table, depending on the file type.

File format	Value
.one	0x0000002A
.onetoc2	0x0000001B

ffvOldestCodeThatHasWrittenToThisFile (4 bytes): An unsigned integer. MUST be one of the values in the following table, depending on the file format of this file.

File format	Value
.one	0x0000002A
.onetoc2	0x0000001B

ffvNewestCodeThatHasWrittenToThisFile (4 bytes): An unsigned integer. MUST be one of the values in the following table, depending on the file format of this file.

File format	Value
.one	0x0000002A
.onetoc2	0x0000001B

ffvOldestCodeThatMayReadThisFile (4 bytes): An unsigned integer. MUST be one of the values in the following table, depending on the file format of this file.

File format	Value
.one	0x0000002A
.onetoc2	0x0000001B

fcrLegacyFreeChunkList (8 bytes): A **FileChunkReference32** structure (section 2.2.4.1) that MUST have a value of "fcrZero" (see section 2.2.4).

fcrLegacyTransactionLog (8 bytes): A **FileChunkReference32** structure that MUST be "fcrNil" (see section 2.2.4).

cTransactionsInLog (4 bytes): An unsigned integer that specifies the number of transactions in the transaction log (section 2.3.3). MUST NOT be zero.

cbLegacyExpectedFileLength (4 bytes): An unsigned integer that MUST be zero, and MUST be ignored.

rgbPlaceholder (8 bytes): An unsigned integer that MUST be zero, and MUST be ignored.

fcrLegacyFileNodeListRoot (8 bytes): A **FileChunkReference32** structure that MUST be "fcrNil".

cbLegacyFreeSpaceInFreeChunkList (4 bytes): An unsigned integer that MUST be zero, and MUST be ignored.

fNeedsDefrag (1 byte): MUST be ignored.

fRepairedFile (1 byte): MUST be ignored.

fNeedsGarbageCollect (1 byte): MUST be ignored.

fHasNoEmbeddedFileObjects (1 byte): An unsigned integer that MUST be zero, and MUST be ignored.

guidAncestor (16 bytes): A GUID that specifies the **Header.guidFile** field of the table of contents file, as specified by [MS-ONE] section 2.1.15, given by the following table:

Table of contents file format	Location of table of contents file
Section file (.one)	Table of contents file is located in the same directory as this file.
Table of contents file (.onetoc2)	Table of contents file is located in the parent directory of this file.

If the GUID is {00000000-0000-0000-0000-000000000000}, this field does not reference a table of contents file.

crcName (4 bytes): An unsigned integer that specifies the CRC value (section 2.1.2) of the name of this revision store file. The name is the **Unicode** representation of the file name with its extension and an additional null character at the end. This **CRC** is calculated using the CRC algorithm for the .one file (section 2.1.2), regardless of this revision store file format.

fcrHashedChunkList (12 bytes): A **FileChunkReference64x32** structure (section 2.2.4.4) that specifies a reference to the first **FileNodeListFragment** in a hashed chunk list (section 2.3.4). If the value of the **FileChunkReference64x32** structure is "fcrZero" or "fcrNil", the hashed chunk list does not exist.

fcrTransactionLog (12 bytes): A **FileChunkReference64x32** structure that specifies a reference to the first **TransactionLogFragment** structure (section 2.3.3.1) in a transaction log (section 2.3.3). The value of the **fcrTransactionLog** field MUST NOT be "fcrZero" and MUST NOT be "fcrNil".

fcrFileNodeListRoot (12 bytes): A **FileChunkReference64x32** structure that specifies a reference to a root file node list (section 2.1.14). The value of the **fcrFileNodeListRoot** field MUST NOT be "fcrZero" and MUST NOT be "fcrNil".

fcrFreeChunkList (12 bytes): A **FileChunkReference64x32** structure that specifies a reference to the first **FreeChunkListFragment** structure (section 2.3.2.1). If the value of the **FileChunkReference64x32** structure is "fcrZero" or "fcrNil", then the free chunk list (section 2.3.2) does not exist.

cbExpectedFileLength (8 bytes): An unsigned integer that specifies the size, in bytes, of this revision store file.

cbFreeSpaceInFreeChunkList (8 bytes): An unsigned integer that SHOULD<3> specify the size, in bytes, of the free space specified by the free chunk list.

guidFileVersion (16 bytes): A GUID, as specified by [MS-DTYP]. When either the value of **cTransactionsInLog** field or the **guidDenyReadFileVersion** field is being changed, **guidFileVersion** MUST be changed to a new GUID.

nFileVersionGeneration (8 bytes): An unsigned integer that specifies the number of times the file has changed. MUST be incremented when the **guidFileVersion** field changes.

guidDenyReadFileVersion (16 bytes): A GUID, as specified by [MS-DTYP]. When the existing contents of the file are being changed, excluding the **Header** structure of the file and unused storage blocks, **guidDenyReadFileVersion** MUST be changed to a new GUID.

grfDebugLogFlags (4 bytes): MUST be zero. MUST be ignored.

fcrDebugLog (12 bytes): A **FileChunkReference64x32** structure that MUST have a value "fcrZero". MUST be ignored.

fcrAllocVerificationFreeChunkList (12 bytes): A **FileChunkReference64x32** structure that MUST be "fcrZero". MUST be ignored.

bnCreated (4 bytes): An unsigned integer that specifies the build number of the application that created this revision store file. SHOULD<4> be ignored.

bnLastWroteToThisFile (4 bytes): An unsigned integer that specifies the build number of the application that last wrote to this revision store file. SHOULD<5> be ignored.

bnOldestWritten (4 bytes): An unsigned integer that specifies the build number of the oldest application that wrote to this revision store file. SHOULD<6> be ignored.

bnNewestWritten (4 bytes): An unsigned integer that specifies the build number of the newest application that wrote to this revision store file. SHOULD<7> be ignored.

rgbReserved (728 bytes): MUST be zero. MUST be ignored.

2.3.2 Free Chunk List

The free chunk list specifies a list of unused storage blocks in the file. The free chunk list consists of a sequence of one or more **FreeChunkListFragment** structures (section 2.3.2.1). The location of the first **FreeChunkListFragment** structure is specified by a **Header.fcrFreeChunkList** (section 2.3.1) field. When writing into any of the storage blocks specified by **fcrFreeChunk** the free chunk list MUST be updated to reflect the change.

2.3.2.1 FreeChunkListFragment

The **FreeChunkListFragment** structure specifies a fragment in the free chunk list (section 2.3.2).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
crc																															
fcrNextChunk																															
...																															
...																															
fcrFreeChunk (variable)																															

...

crc (4 bytes): An unsigned integer that specifies the CRC (section 2.1.2) of a **fcrFreeChunk** field.

fcrNextChunk (12 bytes): A **FileChunkReference64x32** structure (section 2.2.4.4) that specifies a reference to another **FreeChunkListFragment** structure. If the value of the **fcrNextChunk** field is "fcrNil" (see section 2.2.4), then this is the last **FreeChunkListFragment** structure.

fcrFreeChunk (variable): An array of **FileChunkReference64** structures (section 2.2.4.3) where each element in the array specifies a reference to unused parts of the file. The number of elements is given by (*cb* – 16) / 16 where *cb* is the size, in bytes, of this **FreeChunkListFragment** structure; *cb* is specified by the file chunk reference (section 2.2.4) that references this **FreeChunkListFragment** structure.

2.3.3 Transaction Log

To ensure integrity of the files, each write to a file is structured as a transaction.

A transaction consists of the following types of data:

- The **FileNode** structures (section 2.4.3) and file node lists (section 2.4) that specify the actual user data.

  New file node lists can be added by the transaction. **FileNode** structures in existing file node lists MUST NOT be modified or removed. New **FileNode** structures can be added to existing file node lists.

- A set of **TransactionEntry** structures (section 2.3.3.2) that specify which file node lists are added or modified by this transaction and the new number of **FileNode** structures in those file node lists.

  The **TransactionEntry** structures for all transactions are stored sequentially in **TransactionLogFragment.sizeTable**. The last entry for a transaction MUST be a special sentinel entry with the value of the **TransactionEntry.srcID** field set to 0x00000001.

- A **Header.cTransactionsInLog** field (section 2.3.1) that maintains the total number of transactions that have occurred. The transaction’s number is defined by the order of the transaction in the transaction log.

A transaction is committed only after the **Header.cTransactionsInLog** field is updated to match the number of the transaction. All **TransactionEntry** structures for transactions with higher numbers and **FileNode** structures added by them MUST be ignored.

The **Header.fcrTransactionLog** field (section 2.3.1) references the first **TransactionLogFragment** structure (section 2.3.3.1) in the log.

2.3.3.1 TransactionLogFragment

The **TransactionLogFragment** structure specifies an array of **TransactionEntry** structures (section 2.3.3.2) and a reference to the next **TransactionLogFragment** structure if it exists.

This structure has the following format.

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

...
nextFragment
...
...

sizeTable (variable): An array of **TransactionEntry** structures. A transaction MUST add all of its entries to the array sequentially and MUST terminate with a sentinel entry with **TransactionEntry.srcID** set to 0x00000001.

nextFragment (12 bytes): An optional **FileChunkReference64x32** structure (section 2.2.4.4) that specifies the location and size of the next **TransactionLogFragment** structure. If this fragment contains the last transaction as specified by the **Header.cTransactionsInLog** field, **nextFragment** is undefined and MUST be ignored.

2.3.3.2 TransactionEntry

The **TransactionEntry** structure specifies either the file node list (section 2.4) included in the current transaction and the number of **FileNode** structures (section 2.4.3) in the list, or the sentinel entry of the current transaction.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
srcID																															
TransactionEntrySwitch																															

srcID (4 bytes): An unsigned integer that specifies the identity of the file node list modified by this transaction, or the sentinel entry for the transaction. A value of 0x00000001 specifies the sentinel entry. Otherwise, MUST be equal to the value of the **FileNodeListFragment.header.FileNodeListID** field (section 2.4.2) of a file node list in the file.

TransactionEntrySwitch (4 bytes): An unsigned integer of 4 bytes in size. The meaning depends on the value of **srcID**.

Value of srcID	Value of TransactionEntrySwitch
0x00000001	SHOULD be equal to CRC (section 2.1.2) of all **TransactionEntry** structures in the current transaction.<8>
other	MUST be equal to the new number of **FileNode** structures in the file node list with a **FileNodeListFragment.header.FileNodeListID** field value equal to the value of the **srcID** field after the file node list is added or modified by the transaction.

A non-sentinel **TransactionEntry** structure MUST add one or more **FileNode** structures to the file node list it adds or modifies.

2.3.4 Hashed Chunk List

A hashed chunk list is an optional file node list (section 2.4) that specifies a collection of **FileNode** structures (section 2.4.3) with **FileNodeID** field values equal to "0x0C2" (**HashedChunkDescriptor2FND** structure, section 2.3.4.1). The **Header.fcrHashedChunkList** field (section 2.3.1) references the first **FileNodeListFragment** structure (section 2.4.1) in the hashed chunk list, if it exists.

2.3.4.1 HashedChunkDescriptor2FND

The data for a **FileNode** structure (section 2.4.3) that specifies a reference to an **ObjectSpaceObjectPropSet** structure (section 2.6.1).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BlobRef (variable)																															
...																															
guidHash (16 bytes)																															
...																															
...																															

BlobRef (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies the location and size of an **ObjectSpaceObjectPropSet** structure.

guidHash (16 bytes): An unsigned integer that specifies an MD5 checksum, as specified in [RFC1321], of data referenced by the **BlobRef** field.

2.4 File Node List

A file node list is the basic logical structure used to organize data in the file. Each list logically consists of a sequence of **FileNode** structures (section 2.4.3) that can contain data, references to data, or references to other file node lists.

For storage purposes a file node list can be divided into one or more **FileNodeListFragment** structures (section 2.4.1). Each fragment can specify whether there are more fragments in the list and the location of the next fragment. Each fragment specifies a sub-sequence of **FileNode** structures from the file node list.

When specifying the structure of a specific file node list in this document, the division of the list into fragments is ignored and **FileNode** structures with **FileNode.FileNodeID** field values equal to 0x0FF ("ChunkTerminatorFND") are not specified.

All file node list fragments in a file MUST form a tree. The **Header.fcrFileNodeListRoot** field (section 2.3.1) specifies the first fragment of the file node list that is the root of the tree.

2.4.1 FileNodeListFragment

The **FileNodeListFragment** structure specifies a sequence of file nodes from a file node list (section 2.4). The size of the **FileNodeListFragment** structure is specified by the structure that references it.

All fragments in the same file node list MUST have the same **FileNodeListFragment.header.FileNodeListID** field.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
header (16 bytes)																															
...																															
...																															
rgFileNodes (variable)																															
...																															
padding (variable)																															
...																															
nextFragment																															
...																															
...																															
footer																															
...																															

header (16 bytes): A **FileNodeListHeader** structure (section 2.4.2).

rgFileNodes (variable): A stream of bytes that contains a sequence of **FileNode** structures (section 2.4.3). The stream is terminated when any of the following conditions is met:

- The number of bytes between the end of the last read **FileNode** and the **nextFragment** field is less than 4 bytes.
- A **FileNode** structure with a **FileNodeID** field value equal to 0x0FF (**ChunkTerminatorFND** structure, section 2.4.3) is read. If a **ChunkTerminatorFND** structure is present, the value of the **nextFragment** field MUST be a valid **FileChunkReference64x32** structure (section 2.2.4.4) to the next **FileNodeListFragment** structure.
- The number of **FileNode** structures read for the containing file node list is equal to the number of nodes specified for the list by the transaction log (section 2.3.3) in the last **TransactionEntry** (section 2.3.3.2) that modified the list. In this case the **nextFragment** field MUST be ignored.

padding (variable): An optional array of bytes between the last **FileNode** structure in the **rgFileNodes** field and the **nextFragment** field. Undefined and MUST be ignored.

nextFragment (12 bytes): A **FileChunkReference64x32** structure (section 2.2.4.4) that specifies whether there are more fragments in this file node list, and if so, the location and size of the next fragment.

If this is the last fragment, the value of the **nextFragment** field MUST be "fcrNil" (see section 2.2.4). Otherwise the value of the **nextFragment.stp** field MUST specify the location of a valid **FileNodeListFragment** structure, and the value of the **nextFragment.cb** field MUST be equal to the size of the referenced fragment including the **FileNodeListFragment.header** field and the **FileNodeListFragment.footer** field.

The location of the **nextFragment** field is calculated by adding the size of this **FileNodeListFragment** structure minus the size of the **nextFragment** and **footer** fields to the location of this **FileNodeListFragment** structure.

footer (8 bytes): An unsigned integer; MUST be "0x8BC215C38233BA4B". Specifies the end of the **FileNodeListFragment** structure.

2.4.2 FileNodeListHeader

The **FileNodeListHeader** structure specifies the beginning of a **FileNodeListFragment** structure (section 2.4.1).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
uintMagic																															
...																															
FileNodeListID																															
nFragmentSequence																															

uintMagic (8 bytes): An unsigned integer; MUST be "0xA4567AB1F5F7F4C4".

FileNodeListID (4 bytes): An unsigned integer that specifies the identity of the file node list (section 2.4) this fragment belongs to. MUST be equal to or greater than 0x00000010. The pair of **FileNodeListID** and **nFragmentSequence** fields MUST be unique relative to other **FileNodeListFragment** structures in the file.

nFragmentSequence (4 bytes): An unsigned integer that specifies the index of the fragment in the file node list containing the fragment. The **nFragmentSequence** field of the first fragment in a given file node list MUST be 0 and the **nFragmentSequence** fields of all subsequent fragments in this list MUST be sequential.

2.4.3 FileNode

A **FileNode** structure is the basic unit for holding and referencing data in the file. **FileNode** structures are organized into file node lists (section 2.4).

A **FileNode** structure is divided into header fields and a data field, **fnd**. The header fields specify what type of **FileNode** structure it is, and what format the **fnd** field is in. The **fnd** field can be empty, or it can contain data directly, or it can contain a reference to another block of the file by byte position and byte count, or it can contain both data and a reference.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
FileNodeID										Size													A		B		C				D
fnd (variable)																															
...																															

FileNodeID (10 bits): An unsigned integer that specifies the type of this **FileNode** structure. The meaning of this value is specified by the **fnd** field.

Size (13 bits): An unsigned integer that specifies the size, in bytes, of this **FileNode** structure.

A - StpFormat (2 bits): An unsigned integer that specifies the size and format of the **FileNodeChunkReference.stp** field specified by the **fnd** field if this **FileNode** structure has a value of the **BaseType** field equal to 1 or 2. MUST be ignored if the value of the **BaseType** field of this **FileNode** structure is equal to 0. The meaning of the **StpFormat** field is given by the following table.

Value	Meaning
0	8 bytes, uncompressed.
1	4 bytes, uncompressed.
2	2 bytes, compressed.
3	4 bytes, compressed.

The value of an uncompressed file pointer specifies a location in the file. To uncompress a compressed file pointer, multiply the value by 8.

B - CbFormat (2 bits): An unsigned integer that specifies the size and format of the **FileNodeChunkReference.cb** field specified by the **fnd** field if this **FileNode** structure has a **BaseType** field value equal to 1 or 2. MUST be 0 and MUST be ignored if **BaseType** of this **FileNode** structure is equal to 0. The meaning of **CbFormat** is given by the following table.

Value	Meaning
0	4 bytes, uncompressed.
1	8 bytes, uncompressed.
2	1 byte, compressed.
3	2 bytes, compressed.

The value of an uncompressed byte count specifies the size, in bytes, of the data referenced by a **FileNodeChunkReference** structure (section 2.2.4.2). To uncompress a compressed byte count, multiply the value by 8.

C - BaseType (4 bits): An unsigned integer that specifies whether the structure specified by **fnd** contains a **FileNodeChunkReference** structure (section 2.2.4.2). MUST be one of the values described in the following table.

Value	Meaning
0	This **FileNode** structure does not reference other data. The data structure specified by **fnd** MUST NOT contain a **FileNodeChunkReference** structure. The **StpFormat** and **CbFormat** fields MUST be ignored.
1	This **FileNode** structure contains a reference to data. The first field in the data structure specified by an **fnd** field MUST be a **FileNodeChunkReference** structure that specifies the location and size of the referenced data. The type of the **FileNodeChunkReference** structure is specified by the **StpFormat** and **CbFormat** fields.
2	This **FileNode** structure contains a reference to a file node list. The first field in the data structure specified by the **fnd** field MUST be a **FileNodeChunkReference** structure that specifies the location and size of a file node list. The type of the **FileNodeChunkReference** is specified by the **StpFormat** and **CbFormat** fields.

D - Reserved (1 bit): MUST be 1, and MUST be ignored.

fnd (variable): A field that specifies additional data for this **FileNode** structure, if present. The type of structure is specified by the value of the **FileNodeID** field. Some **FileNodeID** field values and their corresponding structures are valid for .one or .onetoc2 only. The meaning of the **FileNodeID** field value is given by the following table.

Value of FileNodeID	Basetype	Fnd structure	Allowed file formats
0x004	0	ObjectSpaceManifestRootFND (section 2.5.1)	one onetoc2
0x008	2	ObjectSpaceManifestListReferenceFND (section 2.5.2)	one onetoc2
0x00C	0	ObjectSpaceManifestListStartFND (section 2.5.3)	one onetoc2
0x010	2	RevisionManifestListReferenceFND (section 2.5.4)	one onetoc2
0x014	0	RevisionManifestListStartFND (section 2.5.5)	one onetoc2
0x01B	0	RevisionManifestStart4FND (section 2.5.6)	onetoc2
0x01C	0	RevisionManifestEndFND Specifies the end of a revision manifest (section 2.1.9). MUST contain no data.	one onetoc2
0x01E	0	RevisionManifestStart6FND (section 2.5.7)	one
0x01F	0	RevisionManifestStart7FND (section 2.5.8)	one
0x021	0	GlobalIdTableStartFNDX (section 2.5.9)	onetoc2
0x022	0	GlobalIdTableStart2FND Specifies the beginning of the global identification table (section 2.1.3). MUST contain no data.	one
0x024	0	GlobalIdTableEntryFNDX (section 2.5.10)	one

Value of FileNodeID	Basetype	Fnd structure	Allowed file formats
			onetoc2
0x025	0	GlobalIdTableEntry2FNDX (section 2.5.11)	onetoc2
0x026	0	GlobalIdTableEntry3FNDX (section 2.5.12)	onetoc2
0x028	0	GlobalIdTableEndFNDX Specifies the end of the global identification table (section 2.1.3). MUST contain no data.	one onetoc2
0x02D	1	ObjectDeclarationWithRefCountFNDX (section 2.5.23)	onetoc2
0x02E	1	ObjectDeclarationWithRefCount2FNDX (section 2.5.24)	onetoc2
0x041	1	ObjectRevisionWithRefCountFNDX (section 2.5.13)	onetoc2
0x042	1	ObjectRevisionWithRefCount2FNDX (section 2.5.14)	onetoc2
0x059	0	RootObjectReference2FNDX (section 2.5.15)	onetoc2
0x05A	0	RootObjectReference3FND (section 2.5.16)	one
0x05C	0	RevisionRoleDeclarationFND (section 2.5.17)	one onetoc2
0x05D	0	RevisionRoleAndContextDeclarationFND (section 2.5.18)	one
0x072	0	ObjectDeclarationFileData3RefCountFND (section 2.5.27)	one
0x073	0	ObjectDeclarationFileData3LargeRefCountFND (section 2.5.28)	one
0x07C	1	ObjectDataEncryptionKeyV2FNDX (section 2.5.19)	one
0x084	1	ObjectInfoDependencyOverridesFND (section 2.5.20)	one onetoc2
0x08C	0	DataSignatureGroupDefinitionFND (section 2.5.33)	one onetoc2
0x090	2	FileDataStoreListReferenceFND (section 2.5.21)	one
0x094	1	FileDataStoreObjectReferenceFND (section 2.5.22)	one
0x0A4	1	ObjectDeclaration2RefCountFND (section 2.5.25)	one
0x0A5	1	ObjectDeclaration2LargeRefCountFND (section 2.5.26)	one
0x0B0	2	ObjectGroupListReferenceFND (section 2.5.31)	one
0x0B4	0	ObjectGroupStartFND (section 2.5.32)	one
0x0B8	0	ObjectGroupEndFND Specifies the end of an object group (section 2.1.13). MUST contain no data.	one
0x0C2	1	HashedChunkDescriptor2FND (section 2.3.4.1)	one
0x0C4	1	ReadOnlyObjectDeclaration2RefCountFND (section 2.5.29)	one
0x0C5	1	ReadOnlyObjectDeclaration2LargeRefCountFND (section	one

Value of FileNodeID	Basetype	Fnd structure	Allowed file formats
		2.5.30)	
0x0FF		ChunkTerminatorFND Specifies the end of the stream of **FileNode** structures in a **FileNodeListFragment** structure (section 2.4.1). MUST contain no data. MUST NOT be used in **FileNodeListFragment** structure that is the last fragment in the containing file node list.	one onetoc2

2.5 File Node Types

2.5.1 ObjectSpaceManifestRootFND

The data for a **FileNode** structure (section 2.4.3) that specifies the root object space (section 2.1.4) in a revision store file. There MUST be only one **ObjectSpaceManifestRootFND** structure (section 2.5.1) in the revision store file. This **FileNode** structure MUST be in the root file node list (section 2.1.14).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
gosidRoot (20 bytes)																															
...																															
...																															

gosidRoot (20 bytes): An **ExtendedGUID** structure (section 2.2.1) that specifies the identity of the root object space. This value MUST be equal to the **ObjectSpaceManifestListReferenceFND.gosid** field (section 2.5.2) of an object space within the object space manifest list (section 2.1.6).

2.5.2 ObjectSpaceManifestListReferenceFND

The data for a **FileNode** structure (section 2.4.3) that specifies the reference to an object space manifest list (section 2.1.6).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ref (variable)																															
...																															
gosid (20 bytes)																															

...
...

ref (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies the location and size of the first **FileNodeListFragment** structure (section 2.4.1) in the object space manifest list.

gosid (20 bytes): An **ExtendedGUID** structure (section 2.2.1) that specifies the identity of the object space (section 2.1.4) specified by the object space manifest list. MUST NOT be {{00000000-0000-0000-0000-000000000000},0} and MUST be unique relative to the other **ObjectSpaceManifestListReferenceFND.gosid** fields in this file.

2.5.3 ObjectSpaceManifestListStartFND

The data for a **FileNode** structure (section 2.4.3) that specifies the beginning of an object space manifest list (section 2.1.6).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
gosid (20 bytes)																															
...																															
...																															

gosid (20 bytes): An **ExtendedGUID** structure that specifies the identity of the object space (section 2.1.4) being specified by this object space manifest list. MUST match the **ObjectSpaceManifestListReferenceFND.gosid** field (section 2.5.2) of the **FileNode** structure that referenced this file node list (section 2.4).

2.5.4 RevisionManifestListReferenceFND

The data for a **FileNode** structure (section 2.4.3) that specifies the reference to a revision manifest list (section 2.1.10) for the current object space (section 2.1.4).

This structure has the following format.

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

ref (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies the location and size of the first **FileNodeListFragment** structure (section 2.4.1) in the revision manifest list.

2.5.5 RevisionManifestListStartFND

The data for a **FileNode** structure (section 2.4.3) that specifies the beginning of a revision manifest list (section 2.1.10).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
gosid (20 bytes)																															
...																															
...																															
nInstance																															

gosid (20 bytes): An **ExtendedGUID** structure (section 2.2.1) that specifies the identity of the object space (section 2.1.4) being revised by the revisions (section 2.1.8) in this list.

nInstance (4 bytes): MUST be ignored.

2.5.6 RevisionManifestStart4FND

The data for a **FileNode** structure (section 2.4.3) that specifies the beginning of a revision manifest (section 2.1.9). This revision manifest applies to the default context (section 2.1.11) of the containing object space (section 2.1.4).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
rid (20 bytes)																															
...																															
...																															
ridDependent (20 bytes)																															
...																															
...																															
timeCreation																															
...																															
RevisionRole																															
odcsDefault																															

rid (20 bytes): An **ExtendedGUID** structure (section 2.2.1) that specifies the identity of this revision (section 2.1.8). MUST NOT be "{{00000000-0000-0000-0000-000000000000}, 0}" and MUST be unique among **RevisionManifestStart4FND.rid** fields within the containing revision manifest list (section 2.1.10).

ridDependent (20 bytes): An **ExtendedGUID** structure that specifies the identity of a dependency revision.

If the value is "{{00000000-0000-0000-0000-000000000000}, 0}", then this revision manifest has no dependency revision. Otherwise, this value MUST be equal to the **RevisionManifestStart4FND.rid** field of a previous revision manifest within this (section 2.1.9) revision manifest list.

timeCreation (8 bytes): Undefined and MUST be ignored.

RevisionRole (4 bytes): An integer that specifies the revision role (section 2.1.12) that labels this revision (section 2.1.8).

odcsDefault (2 bytes): An unsigned integer that specifies whether the data contained by this revision manifest is encrypted. MUST be 0 and MUST be ignored.

2.5.7 RevisionManifestStart6FND

The data for a **FileNode** structure (section 2.4.3) that specifies the beginning of a revision manifest (section 2.1.9) for the default context (section 2.1.11) of an object space (section 2.1.4).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
rid (20 bytes)																															
...																															
...																															
ridDependent (20 bytes)																															
...																															
...																															
RevisionRole																															
odcsDefault																															

rid (20 bytes): An **ExtendedGUID** structure (section 2.2.1) that specifies the identity of this revision (section 2.1.8). MUST NOT be "{{00000000-0000-0000-0000-000000000000}, 0}" and MUST be unique among **RevisionManifestStart6FND.rid** and **RevisionManifestStart7FND.base.rid** fields within the containing revision manifest list (section 2.1.10).

ridDependent (20 bytes): An **ExtendedGUID** structure that specifies the identity of a dependency revision. If the value is "{{00000000-0000-0000-0000-000000000000}, 0}", then this revision manifest has no dependency revision. Otherwise, this value MUST be equal to the

RevisionManifestStart6FND.rid field or the **RevisionManifestStart7FND.base.rid** field of a previous revision manifest within this revision manifest list.

RevisionRole (4 bytes): An integer that specifies the revision role (section 2.1.12) that labels this revision (section 2.1.8).

odcsDefault (2 bytes): An unsigned integer that specifies whether the data contained by this revision manifest is encrypted. MUST be one of the values described in the following table.

Value	Meaning
0x0000	No encryption.
0x0002	Encrypted. Property sets within this revision manifest MUST be ignored and MUST NOT be altered.

MUST specify the same type of data encoding as used in the dependency revision (section 2.1.8), if one was specified in the **ridDependent** field.

2.5.8 RevisionManifestStart7FND

The data for a **FileNode** structure (section 2.4.3) that specifies the beginning of a revision manifest (section 2.1.9) for a context (section 2.1.11) of an object space (section 2.1.4).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
base (46 bytes)																															
...																															
...																															
...																gctxid (20 bytes)															
...																															
...																															
...																															

base (46 bytes): A **RevisionManifestStart6FND** structure (section 2.5.7) that specifies the identity and other attributes of this revision (section 2.1.8).

gctxid (20 bytes): An **ExtendedGUID** structure (section 2.2.1) that specifies the context that labels this revision (section 2.1.8).

2.5.9 GlobalIdTableStartFNDX

The data for a **FileNode** structure (section 2.4.3) that specifies the beginning of a global identification table (section 2.1.3). The value of the **FileNode.FileNodeID** field MUST be 0x021.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Reserved																															

Reserved (1 byte): MUST be 0, and MUST be ignored.

2.5.10 GlobalIdTableEntryFNDX

The data for a **FileNode** structure (section 2.4.3) that specifies an entry in the current global identification table (section 2.1.3). The value of the **FileNode.FileNodeID** field MUST be 0x024.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
index																															
guid (16 bytes)																															
...																															
...																															

index (4 bytes): An unsigned integer that specifies the index of the entry. MUST be less than 0xFFFFFF. MUST be unique relative to the other indexes in this global identification table specified by **FileNode** structures with the values of the **FileNode.FileNodeID** fields equal to 0x024 (**GlobalIdTableEntryFNDX** structure), 0x25 (**GlobalIdTableEntry2FNDX** structure), and 0x26 (**GlobalIdTableEntry3FNDX** structure).

guid (16 bytes): A **GUID**, as specified by [MS-DTYP]. MUST NOT be {00000000-0000-0000-0000-000000000000} and MUST be unique relative to the other **GlobalIDTableEntryFNDX.guid** fields in this global identification table (section 2.1.3).

2.5.11 GlobalIdTableEntry2FNDX

The data for a **FileNode** structure (section 2.4.3) that specifies an entry in the current global identification table (section 2.1.3) by referring to an entry in the global identification table of the dependency revision (section 2.1.8) of the revision manifest (section 2.1.9) that contains this global identification table. The value of the **FileNode.FileNodeID** field MUST be 0x025.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
iIndexMapFrom																															
iIndexMapTo																															

iIndexMapFrom (4 bytes): An unsigned integer that specifies the index of the entry in the dependency revision's global identification table that is used to define this entry. The index MUST be present in the global identification table of the dependency revision (section 2.1.8).

iIndexMapTo (4 bytes): An unsigned integer that specifies the index of the entry in the current global identification table. MUST be less than 0xFFFFFF. MUST be unique relative to the other indices in this global identification table specified by **FileNode** structures with the value of the **FileNode.FileNodeID** field equal to 0x024 (**GlobalIdTableEntryFNDX** structure), 0x25 (**GlobalIdTableEntry2FNDX** structure), and 0x26 (**GlobalIdTableEntry3FNDX** structure).

2.5.12 GlobalIdTableEntry3FNDX

The data for a **FileNode** structure (section 2.4.3) that specifies a range of entries in the current global identification table (section 2.1.3) by referring to a range of entries in the global identification table of the dependency revision (section 2.1.8) of the revision manifest (section 2.1.9) that contains this global identification table. The value of the **FileNode.FileNodeID** field MUST be 0x026.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
iIndexCopyFromStart																															
cEntriesToCopy																															
iIndexCopyToStart																															

iIndexCopyFromStart (4 bytes): An unsigned integer that specifies the index of the first entry in the range of entries in global identification table of the dependency revision (section 2.1.8). The index MUST be present in the global identification table of the dependency revision.

cEntriesToCopy (4 bytes): An unsigned integer that specifies the number of entries in the range. All indices from the value of the **iIndexCopyFromStart** field to the value of (`iIndexCopyFromStart + cEntriesToCopy - 1`) inclusive MUST be present in the global identification table of the dependency revision.

iIndexCopyToStart (4 bytes): An unsigned integer that specifies the index of the first entry in the range in the current global identification table. Other entries assume the consecutive indices.

All indices from the value of **iIndexCopyToStart** to the value of (`iIndexCopyToStart + cEntriesToCopy - 1`) inclusive MUST be less than 0xFFFFFF and MUST be unique relative to the other indices in this global identification table specified by **FileNode** structures with the values of the **FileNode.FileNodeID** field equal to 0x024 (**GlobalIdTableEntryFNDX** structure), 0x025 (**GlobalIdTableEntry2FNDX** structure), and 0x026 (**GlobalIdTableEntry3FNDX** structure).

2.5.13 ObjectRevisionWithRefCountFNDX

The data for a **FileNode** structure (section 2.4.3) that specifies an object (section 2.1.5) that has been revised. The value of the **FileNode.FileNodeID** field MUST be 0x041. The revised object is identified by the **oid** field, and the revised data is at the location specified by the **ref** field.

This structure has the following format.

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

...																															
oid																															
A	B	cRef																													

ref (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies a reference to an **ObjectSpaceObjectPropSet** structure (section 2.6.1) containing the revised data for the object referenced by the **oid** field.

oid (4 bytes): A **CompactID** structure (section 2.2.2) that specifies the object that has been revised.

A - fHasOidReferences (1 bit): A bit that specifies whether the **ObjectSpaceObjectPropSet** structure referenced by the **ref** field contains references to other objects.

B - fHasOsidReferences (1 bit): A bit that specifies whether the **ObjectSpaceObjectPropSet** structure referenced by the **ref** field contains references to object spaces (section 2.1.4).

cRef (6 bits): An unsigned integer that specifies the reference count for this object.

2.5.14 ObjectRevisionWithRefCount2FNDX

The data for a **FileNode** structure (section 2.4.3) that specifies an object (section 2.1.5) that has been revised. The value of the **FileNode.FileNodeID** field MUST be 0x042. The revised object is identified by the **oid** field, and the revised data is at the location specified by the **ref** field.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ref (variable)																															
...																															
oid																															
A	B	Reserved																													
cRef																															

ref (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies a reference to an **ObjectSpaceObjectPropSet** structure (section 2.6.1) containing the revised data for the object referenced by the **oid** field.

oid (4 bytes): A **CompactID** structure (section 2.2.2) that specifies the object that has been revised.

A - fHasOidReferences (1 bit): A bit that specifies whether the **ObjectSpaceObjectPropSet** structure referenced by the **ref** field contains references to other objects.

B - fHasOsidReferences (1 bit): A bit that specifies whether the **ObjectSpaceObjectPropSet** structure referenced by the **ref** field contains references to object spaces (section 2.1.4).

Reserved (30 bits): MUST be zero, and MUST be ignored.

cRef (4 bytes): An unsigned integer that specifies the reference count for this object.

2.5.15 RootObjectReference2FNDX

The data for a **FileNode** structure (section 2.4.3) that specifies the root object (section 2.1.7) of a revision (section 2.1.8) for a particular root role. The value of the **FileNode.FileNodeID** field MUST be 0x059.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
oidRoot																															
RootRole																															

oidRoot (4 bytes): A **CompactID** structure (section 2.2.2) that specifies the identity of the root object of the containing revision for the role specified by the **RootRole** field.

RootRole (4 bytes): An unsigned integer that specifies the role of the root object.

2.5.16 RootObjectReference3FND

The data for a **FileNode** structure (section 2.4.3) that specifies the root object (section 2.1.7) of a revision (section 2.1.8) for a particular root role. The value of the **FileNode.FileNodeID** field MUST be 0x05A.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
oidRoot (20 bytes)																															
...																															
...																															
RootRole																															

oidRoot (20 bytes): An **ExtendedGUID** (section 2.2.1) that specifies the identity of the root object of the containing revision for the role specified by the **RootRole** field.

RootRole (4 bytes): An unsigned integer that specifies the role of the root object.

2.5.17 RevisionRoleDeclarationFND

The data for a **FileNode** structure (section 2.4.3) that specifies a new additional revision role (section 2.1.12) value to associate with a revision (section 2.1.8). The revision role label is in the default context (section 2.1.11).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
rid (20 bytes)																															
...																															
...																															
RevisionRole																															

rid (20 bytes): An **ExtendedGUID** structure (section 2.2.1) that specifies the identity of the revision to add the revision role to. MUST match the value of the **RevisionManifestStart4FND.rid** field, **RevisionManifestStart6FND.rid** field or **RevisionManifestStart7FND.base.rid** field of one of preceding revision manifests (section 2.1.9) in the current revision manifest list (section 2.1.10).

RevisionRole (4 bytes): Specifies a revision role for the default context.

2.5.18 RevisionRoleAndContextDeclarationFND

The data for a **FileNode** structure (section 2.4.3) that specifies a new additional revision role (section 2.1.12) and context (section 2.1.11) pair to associate with a revision (section 2.1.8).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
base (24 bytes)																															
...																															
...																															
gctxid (20 bytes)																															
...																															
...																															

base (24 bytes): A **RevisionRoleDeclarationFND** structure (section 2.5.17) that specifies the revision and revision role.

gctxid (20 bytes): An **ExtendedGUID** structure (section 2.2.1) that specifies the context.

2.5.19 ObjectDataEncryptionKeyV2FNDX

The data for a **FileNode** structure (section 2.4.3) that specifies that the object space (section 2.1.4) is encrypted. If any revision manifest (section 2.1.9) for an object space contains this **FileNode** structure, all other revision manifests for this object space MUST contain this **FileNode** structure, and these **FileNode** structures MUST point to structures with identical encryption data.

This structure has the following format.

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

ref (variable): Specifies a **FileNodeChunkReference** structure (section 2.2.4.2) that refers to the following structure.

Field	Meaning
Header	A 64-bit unsigned integer that MUST be 0xFB6BA385DAD1A067.
Encryption Data	A variable sized array of bytes. MUST be ignored.
Footer	A 64-bit unsigned integer that MUST be 0x2649294F8E198B3C.

2.5.20 ObjectInfoDependencyOverridesFND

The data for a **FileNode** structure (section 2.4.3) that specifies updated reference counts for objects (section 2.1.5). The override data is specified by the **ref** field if the value of the **ref** field is not "fcrNil" (section 2.2.4); otherwise, the override data is specified by the **data** field. The total size of the **data** field, in bytes, MUST be less than 1024; otherwise, the override data MUST be in the location referenced by the **ref** field.

This structure has the following format.

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

ref (variable): A **FileNodeChunkReference** structure that specifies the location of an **ObjectInfoDependencyOverrideData** structure (section 2.6.10) if the value of the **ref** field is not "fcrNil".

data (variable): An optional **ObjectInfoDependencyOverrideData** structure (section 2.6.10) that specifies the updated reference counts for objects (section 2.1.5). MUST exist if the value of the **ref** field is "fcrNil".

2.5.21 FileDataStoreListReferenceFND

The data for a **FileNode** structure (section 2.4.3) that specifies a file node list (section 2.4) containing references to file data objects. The referenced file node list MUST contain only **FileNode** structures with a **FileNodeID** field value equal to 00x094 (**FileDataStoreObjectReferenceFND** structure). The value of the **FileNode.FileNodeID** field MUST be 0x090.

This structure has the following format.

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

ref (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies a reference to a **FileNodeListFragment** structure (section 2.4.1).

2.5.22 FileDataStoreObjectReferenceFND

The data for a **FileNode** structure (section 2.4.3) that specifies a reference to a **file data object**. All such **FileNode** structures MUST be contained in the file node list (section 2.4) specified by a **FileDataStoreListReferenceFND** structure (section 2.5.21).

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ref (variable)																															
...																															
guidReference (16 bytes)																															
...																															
...																															

ref (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies a reference to a **FileDataStoreObject** structure (section 2.6.13).

guidReference (16 bytes): A **GUID**, as specified by [MS-DTYP], that specifies the identity of this file data object. MUST be unique with respect to all **FileDataStoreObjectReferenceFND** structures.

2.5.23 ObjectDeclarationWithRefCountFNDX

The data for a **FileNode** structure (section 2.4.3) that specifies an object (section 2.1.5) with a reference count. The value of the **FileNode.FileNodeID** field MUST be 0x02D.

This structure has the following format.

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

body																															
...																															
...																cRef															

ObjectRef (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies a reference to an **ObjectSpaceObjectPropSet** structure (section 2.6.1).

body (10 bytes): An **ObjectDeclarationWithRefCountBody** structure (section 2.6.15) that specifies the identity and other attributes of this object.

cRef (1 byte): An unsigned integer that specifies the number of objects that reference this object.

2.5.24 ObjectDeclarationWithRefCount2FNDX

The data for a **FileNode** structure (section 2.4.3) that specifies an object with a reference count. The value of the **FileNode.FileNodeID** field MUST be 0x02E.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ObjectRef (variable)																															
...																															
body																															
...																															
...																cRef															
...																															

ObjectRef (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies a reference to an **ObjectSpaceObjectPropSet** structure (section 2.6.1).

body (10 bytes): An **ObjectDeclarationWithRefCountBody** structure (section 2.6.15) that specifies the identity and other attributes of this object (section 2.1.5).

cRef (4 bytes): An unsigned integer that specifies the reference count for this object.

2.5.25 ObjectDeclaration2RefCountFND

The data for a **FileNode** structure (section 2.4.3) that specifies an object (section 2.1.5) with a reference count. The value of the **FileNode.FileNodeID** field MUST be 0x0A4.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BlobRef (variable)																															
...																															
body																															
...																															
...								cRef																							

BlobRef (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies a reference to an **ObjectSpaceObjectPropSet** structure (section 2.6.1).

body (9 bytes): An **ObjectDeclaration2Body** structure (section 2.6.16) that specifies the identity and other attributes of this object.

cRef (1 byte): An unsigned integer that specifies the reference count for this object (section 2.1.5).

2.5.26 ObjectDeclaration2LargeRefCountFND

The data for a **FileNode** structure (section 2.4.3) that specifies an object (section 2.1.5) with a reference count. The value of the **FileNode.FileNodeID** field MUST be 0x0A5.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
BlobRef (variable)																															
...																															
body																															
...																															
...								cRef																							
...																															

BlobRef (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies a reference to an **ObjectSpaceObjectPropSet** structure (section 2.6.1).

body (9 bytes): An **ObjectDeclaration2Body** structure (section 2.6.16) that specifies the identity and other attributes of this object.

cRef (4 bytes): An unsigned integer that specifies the reference count for this object (section 2.1.5).

2.5.27 ObjectDeclarationFileData3RefCountFND

The data for a **FileNode** structure (section 2.4.3) that specifies an object that references data for a **file data object**. If this object is revised, all declarations of this object MUST specify identical data. The value of the **FileNode.FileNodeID** field MUST be 0x072. This structure has the following format.

This structure has the following format.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
oid																															
jcid																															
cRef								FileDataReference (variable)																							
...																															
Extension (variable)																															
...																															

oid (4 bytes): A **CompactID** structure (section 2.2.2) that specifies the identity of this object.

jcid (4 bytes): A **JCID** structure (section 2.6.14) that specifies the type of this object and the type of the data the object contains.

cRef (1 byte): An unsigned integer that specifies the reference count for this object (section 2.1.5).

FileDataReference (variable): A **StringInStorageBuffer** structure (section 2.2.3) that specifies the type and the target of the reference. The value of the **FileDataReference.StringData** field MUST begin with one of the following strings: "<file>"; "<ifndf>"; "<invfdo>". The prefix specifies the type of the reference and the remaining part of the string specifies the target of the reference:

Prefix	Type of the reference	Remaining part of the string
<file>	A file in the **onefiles folder**	SHOULD specify a file name, including the file extension, of an existing file in the onefiles folder. The name portion of the file name MUST be a string form of a **UUID**, as specified in [RFC4122] section 3. The extension MUST be "onebin".
<ifndf>	**FileDataStoreObject** (section 2.6.13)	Specifies a **curly braced GUID string** that MUST represent one of the **FileDataStoreObjectReferenceFND**.**guidReference** fields.
<invfdo>	Invalid	Specifies that the reference is not valid. MUST be followed by an empty string.

Extension (variable): A **StringInStorageBuffer** (section 2.2.3) that specifies the file extension including period.

2.5.28 ObjectDeclarationFileData3LargeRefCountFND

The data for a **FileNode** structure (section 2.4.3) that specifies an object that references data for a **file data object**. If this object is revised, all declarations of this object MUST specify identical data. The value of the **FileNode.FileNodeID** field MUST be 0x073.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
oid																															
jcid																															
cRef																															
FileDataReference (variable)																															
...																															
Extension (variable)																															
...																															

oid (4 bytes): A **CompactID** structure (section 2.2.2) that specifies the identity of this object.

jcid (4 bytes): A **JCID** structure (section 2.6.14) that specifies the type of this object and the type of the data the object contains.

cRef (4 bytes): An unsigned integer that specifies the reference count for this objects (section 2.1.5).

FileDataReference (variable): A **StringInStorageBuffer** structure (section 2.2.3) that specifies the type and the target of the reference. The value of the **FileDataReference.StringData** field MUST begin with one of the following strings: "<file>"; "<ifndf>"; "<invfdo>". The prefix specifies the type of the reference and the remaining part of the string specifies the target of the reference:

Prefix	Type of the reference	Remaining part of the string
<file>	A file in the **onefiles folder**	SHOULD specify a file name, including the file extension, of an existing file in the onefiles folder. The name portion of the file name MUST be a string form of a **UUID**, as specified in [RFC4122] section 3. The extension MUST be "onebin".
<ifndf>	**FileDataStoreObject** (section 2.6.13)	Specifies a **curly braced GUID string** that MUST represent one of the **FileDataStoreObjectReferenceFND**.**guidReference** fields.
<invfdo>	Invalid	Specifies that the reference is not valid. MUST be followed by an empty string.

Extension (variable): A **StringInStorageBuffer** (section 2.2.3) that specifies the file extension including period.

2.5.29 ReadOnlyObjectDeclaration2RefCountFND

The data for a **FileNode** structure (section 2.4.3) that specifies an object with a reference count. If this object is revised, all declarations of this object MUST specify identical data. The value of the **FileNode.FileNodeID** field MUST be 0x0C4.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
base (variable)																															
...																															
md5Hash (16 bytes)																															
...																															
...																															

base (variable): An **ObjectDeclaration2RefCountFND** structure (section 2.5.25) that specifies the identity and other attributes of this object (section 2.1.5). The values of the **base.body.jcid.IsPropertySet** and **base.body.jcid.IsReadOnly** fields MUST be **true**.

md5Hash (16 bytes): An unsigned integer that specifies an MD5 checksum, as specified in [RFC1321], of the data referenced by the **base.BlobRef** field. If the referenced data is encrypted, the data MUST be decrypted and padded with zeros to an 8-byte boundary before calculating the checksum of the data.

2.5.30 ReadOnlyObjectDeclaration2LargeRefCountFND

The data for a **FileNode** structure (section 2.4.3) that specifies an object with a reference count. If this object is revised, all declarations of this object MUST specify identical data. The value of the **FileNode.FileNodeID** field MUST be 0x0C5.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
base (variable)																															
...																															
md5Hash (16 bytes)																															
...																															
...																															

base (variable): An **ObjectDeclaration2LargeRefCountFND** structure (section 2.5.26) that specifies the identity and other attributes of this object (section 2.1.5). The values of the **base.body.jcid.IsPropertySet** and **base.body.jcid.IsReadOnly** fields MUST be **true**.

md5Hash (16 bytes): An unsigned integer that specifies an MD5 checksum, as specified in [RFC1321], of the data referenced by the **base.BlobRef** field. If the referenced data is encrypted, the data MUST be decrypted and padded with zeros to an 8-byte boundary before calculating the checksum of the data.

2.5.31 ObjectGroupListReferenceFND

The data for a **FileNode** structure (section 2.4.3) that specifies a reference to an object group (section 2.1.13). The value of the **FileNode.FileNodeID** field MUST be set to 0x0B0.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ref (variable)																															
...																															
ObjectGroupID (20 bytes)																															
...																															
...																															

ref (variable): A **FileNodeChunkReference** structure (section 2.2.4.2) that specifies the location and size of the first **FileNodeListFragment** structure (section 2.4.1) in the file node list (section 2.4) of the object group.

ObjectGroupID (20 bytes): An **ExtendedGUID** structure (section 2.2.1) that specifies the identity of the object group that the **ref** field value points to. MUST be the same value as the **ObjectGroupStartFND.oid** field value of the object group that the **ref** field points to.

2.5.32 ObjectGroupStartFND

The data for a **FileNode** structure (section 2.4.3) that specifies the beginning of an object group (section 2.1.13).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
oid (20 bytes)																															
...																															
...																															

oid (20 bytes): An **ExtendedGUID** (section 2.2.1) that specifies the identity of the object group.

2.5.33 DataSignatureGroupDefinitionFND

The data for a **FileNode** structure (section 2.4.3) that specifies a signature for data of objects declared by **FileNode** structures that follow this **FileNode** structure. The signature's effect terminates when a **FileNode** structure with **FileNodeID** field values equal to one of the following is encountered:

1. 0x0B8 (**ObjectGroupEndFND** structure, section 2.4.3)

2. 0x08C (**DataSignatureGroupDefinitionFND** structure, section 2.5.33)

3. 0x01C (**RevisionManifestEndFND** structure, section 2.4.3)

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
DataSignatureGroup (20 bytes)																															
...																															
...																															

DataSignatureGroup (20 bytes): An **ExtendedGUID** structure (section 2.2.1) that specifies the signature. All declarations of an object (section 2.1.5) with the same identity and the same **DataSignatureGroup** field not equal to {{00000000-0000-0000-0000-000000000000}, 0} MUST have the same data.

2.6 Other Structures

2.6.1 ObjectSpaceObjectPropSet

The **ObjectSpaceObjectPropSet** structure specifies the data for an object, including a property set (section 2.1.1) and references to other objects, object spaces (section 2.1.4), and contexts (section 2.1.11).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
OIDs (variable)																															
...																															
OSIDs (variable)																															
...																															
ContextIDs (variable)																															
...																															
body (variable)																															
...																															
padding (variable)																															
...																															

OIDs (variable): An **ObjectSpaceObjectStreamOfOIDs** (section 2.6.2) that specifies the count and list of objects that are referenced by this **ObjectSpaceObjectPropSet**. The count of referenced objects is calculated as the number of properties specified by the **body** field, with **PropertyID** equal to 0x8 plus the number of referenced objects specified by properties with **PropertyID** equal to 0x9, 0x10, and 0x11. This count MUST be equal to the value of **OIDs.header.Count** field. Properties that reference other objects MUST be matched with the **CompactID** structures (section 2.2.2) from **OIDs.body** field in the same order as the properties are listed in the **body.rgPrids** field.

OSIDs (variable): An optional **ObjectSpaceObjectStreamOfOSIDs** structure (section 2.6.3) that specifies the count and list of object spaces referenced by this **ObjectSpaceObjectPropSet** structure. MUST be present if the value of the **OIDs.header.OsidStreamNotPresent** field is **false**; otherwise, the **OSIDs** field MUST NOT be present. The count of referenced object spaces is calculated as the number of properties specified by the **body** field with **PropertyID** equal to 0xA plus the number of referenced object spaces specified by properties with **PropertyID** equal to 0xB, 0x10, and 0x11. This count MUST be equal to the value of **OSIDs.header.Count** field. Properties that reference other object spaces MUST be matched with the **CompactID** structures from **OSIDs.body** field in the same order as the properties are listed in the **body.rgPrids** field.

ContextIDs (variable): An optional **ObjectSpaceObjectStreamOfContextIDs** (section 2.6.4) that specifies the count and list of contexts referenced by this **ObjectSpaceObjectPropSet** structure. MUST be present if **OSIDs** is present and the value of the **OSIDs.header.ExtendedStreamsPresent** field is **true**; otherwise, the **ContextIDs** field MUST NOT be present. The count of referenced contexts is calculated as the number of properties specified by the **body** field with **PropertyID** equal to 0xC plus the number of referenced contexts specified by properties with **PropertyID** equal to 0xD, 0x10, and 0x11. This count MUST be equal to the value of **ContextIDs.header.Count** field. Properties that reference other contexts MUST be matched with the **CompactID** structures from **ContextIDs.body** field in the same order as the properties are listed in the **body.rgPrids** field.

body (variable): A **PropertySet** structure (section 2.6.7) that specifies properties that modify this object, and how other objects relate to this object.

padding (variable): An optional array of bytes that, if present, MUST be zero and MUST be ignored. The total size, in bytes, of an **ObjectSpaceObjectPropSet** structure MUST be a multiple of 8 if the **padding** field is present; the size of the **padding** field is the number of bytes necessary to ensure the total size of **ObjectSpaceObjectPropSet** structure is a multiple of 8. The size of the **padding** field MUST NOT exceed 7 bytes. If the sum of the sizes of the **OIDs**, **OSIDs**, **ContextIDs**, and **body** fields is a multiple of 8, then the **padding** field is not present.

2.6.2 ObjectSpaceObjectStreamOfOIDs

The **ObjectSpaceObjectStreamOfOIDs** structure specifies the count and list of objects referenced by an **ObjectSpaceObjectPropSet** structure (section 2.6.1).

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

header (4 bytes): An **ObjectSpaceObjectStreamHeader** structure (section 2.6.5) that specifies the number of elements in the **body** field and whether the **ObjectSpaceObjectPropSet** structure contains an **OSIDs** field and **ContextIDs** field. If the **OSIDs** field is present, the value of the

header.OsidStreamNotPresent field MUST be **false**; otherwise, it MUST be **true**. If the **ContextIDs** field is present the value of the **header**.**ExtendedStreamsPresent** field MUST be **true**; otherwise, it MUST be **false**.

body (variable): An array of **CompactID** structures (section 2.2.2) where each element in the array specifies the identity of an object. The number of elements is equal to the value of the **header.Count** field.

2.6.3 ObjectSpaceObjectStreamOfOSIDs

The **ObjectSpaceObjectStreamOfOSIDs** structure specifies the count and list of object spaces (section 2.1.4) referenced by an **ObjectSpaceObjectPropSet** structure (section 2.6.1).

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

header (4 bytes): An **ObjectSpaceObjectStreamHeader** structure (section 2.6.5) that specifies the number of elements in the **body** field and whether the **ObjectSpaceObjectPropSet** structure contains **ContextIDs** field. The value of the **header.OsidStreamNotPresent** field MUST be "false". If the **ContextIDs** field is present, the value of the **header**.**ExtendedStreamsPresent** field MUST be **true**; otherwise, it MUST be **false**.

body (variable): An array of **CompactID** structures (section 2.2.2) where each element in the array specifies the identity of an object space. The number of elements is equal to the value of the **header.Count** field.

2.6.4 ObjectSpaceObjectStreamOfContextIDs

The **ObjectSpaceObjectStreamOfContextIDs** structure specifies the count and list of contexts (section 2.1.11) referenced by an **ObjectSpaceObjectPropSet** structure (section 2.6.1).

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

header (4 bytes): An **ObjectSpaceObjectStreamHeader** structure (section 2.6.5) that specifies the number of elements in the **body** field. The value of **header.OsidStreamNotPresent** field and **header**.**ExtendedStreamsPresent** field MUST be **false**.

body (variable): An array of **CompactID** structures (section 2.2.2) where each element in the array specifies a context (section 2.1.11). The number of elements is equal to the value of the **header.Count** field.

2.6.5 ObjectSpaceObjectStreamHeader

The **ObjectSpaceObjectStreamHeader** structure specifies the number of objects (see section 2.1.5) in a stream and whether there are more streams following the stream that contains this **ObjectSpaceObjectStreamHeader** structure.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Count																								Reserved						A	B

Count (24 bits): An unsigned integer that specifies the number of **CompactID** structures (section 2.2.2) in the stream that contains this **ObjectSpaceObjectStreamHeader** structure.

Reserved (6 bits): MUST be zero, and MUST be ignored.

A - ExtendedStreamsPresent (1 bit): A bit that specifies whether the **ObjectSpaceObjectPropSet** structure (section 2.6.1) contains any additional streams of data following this stream of data.

B - OsidStreamNotPresent (1 bit): A bit that specifies whether the **ObjectSpaceObjectPropSet** structure does not contain **OSIDs** or **ContextIDs** fields.

2.6.6 PropertyID

The **PropertyID** structure specifies the identity of a property and the size and location of the data for the property. The meaning of the data contained by a property is specified in [MS-ONE] section 2.1.12.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
id																										type					A

id (26 bits): An unsigned integer that specifies the identity of this property. The meanings of the **id** field values are specified in [MS-ONE] section 2.1.12.

type (5 bits): An unsigned integer that specifies the property type and the size and location of the data for this property. MUST be one of the following values:

Value	Name	Meaning
0x1	NoData	The property contains no data.
0x2	Bool	The property is a Boolean value specified by **boolValue**.
0x3	OneByteOfData	The property contains 1 byte of data in the **PropertySet.rgData** stream field.
0x4	TwoBytesOfData	The property contains 2 bytes of data in the **PropertySet.rgData** stream field.
0x5	FourBytesOfData	The property contains 4 bytes of data in the **PropertySet.rgData** stream field.
0x6	EightBytesOfData	The property contains 8 bytes of data in the **PropertySet.rgData** stream field.

Value	Name	Meaning
0x7	FourBytesOfLengthFollowedByData	The property contains a **prtFourBytesOfLengthFollowedByData** (section 2.6.8) in the **PropertySet.rgData** stream field.
0x8	ObjectID	The property contains one **CompactID** (section 2.2.2) in the **ObjectSpaceObjectPropSet.OIDs.body** stream field.
0x9	ArrayOfObjectIDs	The property contains an array of **CompactID** structures in the **ObjectSpaceObjectPropSet.OIDs.body** stream field. The property contains an unsigned integer of size 4 bytes in the **PropertySet.rgData** stream field that specifies the number of **CompactID** structures this property contains.
0xA	ObjectSpaceID	The property contains one **CompactID** structure in the **ObjectSpaceObjectPropSet.OSIDs.body** stream field.
0xB	ArrayOfObjectSpaceIDs	The property contains an array of **CompactID** structures in the **ObjectSpaceObjectPropSet.OSIDs.body** stream field. The property contains an unsigned integer of size 4 bytes in the **PropertySet.rgData** stream field that specifies the number of **CompactID** structures this property contains.
0xC	ContextID	The property contains one **CompactID** in the **ObjectSpaceObjectPropSet.ContextIDs.body** stream field.
0xD	ArrayOfContextIDs	The property contains an array of **CompactID** structures in the **ObjectSpaceObjectPropSet.ContextIDs.body** stream field. The property contains an unsigned integer of size 4 bytes in the **PropertySet.rgData** stream field that specifies the number of **CompactID** structures this property contains.
0x10	ArrayOfPropertyValues	The property contains a **prtArrayOfPropertyValues** (section 2.6.9) structure in the **PropertySet.rgData** stream field.
0x11	PropertySet	The property contains a child **PropertySet** (section 2.6.7) structure in the **PropertySet**.**rgData** stream field of the parent **PropertySet**.

A - boolValue (1 bit): A bit that specifies the value of a Boolean property. MUST be **false** if the value of the **type** field is not equal to 0x2.

2.6.7 PropertySet

The **PropertySet** structure specifies the format of a property set (section 2.1.1).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
cProperties																rgPrids (variable)															
...																															
rgData (variable)																															
...																															

cProperties (2 bytes): An unsigned integer that specifies the number of properties in this **PropertySet** structure.

rgPrids (variable): An array of **PropertyID** structures (section 2.6.6). The number of elements in the array is equal to the value of the **cProperties** field.

rgData (variable): A stream of bytes that specifies the data for each property specified by a **rgPrids** array. The total size, in bytes, of the **rgData** field is the sum of the sizes specified by the **PropertyID.type** field for each property in a **rgPrids** array. The total size of **rgData** MUST be zero if no property in a **rgPrids** array specifies that it contains data in the **rgData** field.

2.6.8 prtFourBytesOfLengthFollowedByData

The **prtFourBytesOfLengthFollowedByData** structure specifies a container of variable-sized data used by properties in a **PropertySet** structure (section 2.6.7). The total size, in bytes, of **prtFourBytesOfLengthFollowedByData** is equal to **cb** + 4.

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

cb (4 bytes): An unsigned integer that specifies the size, in bytes, of the **Data** field. MUST be less than 0x40000000.

Data (variable): A stream of bytes that specifies the data for the property.

2.6.9 prtArrayOfPropertyValues

The **prtArrayOfPropertyValues** structure specifies a container of variable-sized data that stores an array of properties with the same **PropertyID.type** value.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
cProperties																															
prid																															

Data (variable)
...

cProperties (4 bytes): An unsigned integer that specifies the number of properties in **Data**.

prid (4 bytes): An optional **PropertyID** structure (section 2.6.6) that specifies the type of each property in the array. **PropertyID.type** MUST be 0x11 ("PropertySet"). Values of **PropertyID.id** and **PropertyID.boolValue** are undefined and MUST be ignored. MUST NOT be present if **cProperties** is zero; otherwise, MUST be present.

Data (variable): A stream of bytes that specifies the data for each property in the array. The total size, in bytes, of the **Data** field is the sum of the sizes specified by the **prid.type** field for each property in the array, if **prid** is present. Otherwise, the total size of **Data** is zero if **cProperties** is zero.

2.6.10 ObjectInfoDependencyOverrideData

The **ObjectInfoDependencyOverrideData** structure specifies updated reference counts for objects (section 2.1.5).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
c8BitOverrides																															
c32BitOverrides																															
crc																															
Overrides1 (variable)																															
...																															
Overrides2 (variable)																															
...																															

c8BitOverrides (4 bytes): An unsigned integer that specifies the number of elements in **Overrides1**.

c32BitOverrides (4 bytes): An unsigned integer that specifies the number of elements in **Overrides2**.

crc (4 bytes): An unsigned integer that specifies a **CRC** (section 2.1.2) of the reference counts. The **crc** field is computed as follows:

1. **crc** is initialized to zero.
2. If this **FileNode** structure (section 2.4.3) follows an object group (section 2.1.13) in the revision manifest, the **crc** field is calculated cumulatively on every reference count of every object declared in the group in the order they appear in the group. The following reference counts are used:
   - The **ObjectDeclarationFileData3RefCountFND.cRef** field, extended to an unsigned integer of size 4 bytes.
   - The **ObjectDeclarationFileData3LargeRefCountFND.cRef** field.

- The **ObjectDeclaration2RefCountFND.cRef** field, extended to an unsigned integer of size 4 bytes.
- The **ObjectDeclaration2LargeRefCountFND.cRef** field.
- The **ReadOnlyObjectDeclaration2RefCountFND.cRef** field, extended to an unsigned integer of size 4 bytes.
- The **ReadOnlyObjectDeclaration2LargeRefCountFND.cRef** field.

3. The **crc** field is calculated cumulatively on each element in the **Overrides1** field, where each element is treated as an array of 5 bytes.
4. The **crc** field is calculated cumulatively on each element in the **Overrides2** field, where each element is treated as an array of 8 bytes.

Overrides1 (variable): An array of **ObjectInfoDependencyOverride8** structures (section 2.6.11) that specifies the updated reference counts for objects (section 2.1.5) if the updated reference count is less than or equal to 255.

Overrides2 (variable): An array of **ObjectInfoDependencyOverride32** structures (section 2.6.12) that specifies the updated reference counts for objects if the updated reference count is greater than 255.

2.6.11 ObjectInfoDependencyOverride8

The **ObjectInfoDependencyOverride8** structure specifies the updated reference count for an object (section 2.1.5).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
oid																															
cRef																															

oid (4 bytes): A **CompactID** structure (section 2.2.2) that specifies the identity of the object with the updated reference count. The object MUST already be defined in the current revision (section 2.1.8).

cRef (1 byte): An unsigned integer that specifies the updated reference count for the **oid** field.

2.6.12 ObjectInfoDependencyOverride32

The **ObjectInfoDependencyOverride32** structure specifies the updated reference count for an object (section 2.1.5).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
oid																															
cRef																															

oid (4 bytes): A **CompactID** structure (section 2.2.2) that specifies the identity of the object with the updated reference count. The object MUST already be defined in the current revision (section 2.1.8).

cRef (4 bytes): An unsigned integer that specifies the updated reference count for an **oid** field.

2.6.13 FileDataStoreObject

The **FileDataStoreObject** structure specifies the data for a **file data object**.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
guidHeader (16 bytes)																															
...																															
...																															
cbLength																															
...																															
unused																															
reserved																															
...																															
FileData (variable)																															
...																															
guidFooter (16 bytes)																															
...																															
...																															

guidHeader (16 bytes): A **GUID**, as specified by [MS-DTYP], that specifies the beginning of a **FileDataStoreObject**. MUST be {BDE316E7-2665-4511-A4C4-8D4D0B7A9EAC}.

cbLength (8 bytes): An unsigned integer that specifies the size, in bytes, of the **FileData** field without padding.

unused (4 bytes): MUST be zero, and MUST be ignored.

reserved (8 bytes): MUST be zero, and MUST be ignored.

FileData (variable): A stream of bytes that specifies the data for the file data object. Padding is added to the end of the **FileData** stream to ensure that the FileDataStoreObject structure ends on an 8-byte boundary.

guidFooter (16 bytes): A GUID, as specified by [MS-DTYP], that specifies the end of a **FileDataStoreObject** structure. MUST be {71FBA722-0F79-4A0B-BB13-899256426B24}.

2.6.14 JCID

The **JCID** structure specifies the type of object (section 2.1.5) and the type of data the object contains. A **JCID** structure can be considered to be an unsigned integer of size four bytes as specified by property set (as specified in [MS-ONE] section 2.1.13) and file data object (as specified in [MS-ONE] section 2.1.5).

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
index																A	B	C	D	E	Reserved										

index (2 bytes): An unsigned integer that specifies the type of object.

A - IsBinary (1 bit): Specifies whether the object contains encryption data transmitted over the File Synchronization via SOAP over HTTP Protocol, as specified in [MS-FSSHTTP].

B - IsPropertySet (1 bit): Specifies whether the object contains a property set (section 2.1.1).

C - IsGraphNode (1 bit): Undefined and MUST be ignored.

D - IsFileData (1 bit): Specifies whether the object is a file data object. If the value of **IsFileData** is "true", then the values of the **IsBinary**, **IsPropertySet**, **IsGraphNode**, and **IsReadOnly** fields MUST all be **false**.

E - IsReadOnly (1 bit): Specifies whether the object's data MUST NOT be changed when the object is revised.

Reserved (11 bits): MUST be zero, and MUST be ignored.

2.6.15 ObjectDeclarationWithRefCountBody

The **ObjectDeclarationWithRefCountBody** structure specifies the identity of an object (section 2.1.5) and the type of data the object contains.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
oid																															
jci										odcs				A		B	C	fReserved2													
...																															

oid (4 bytes): A **CompactID** structure (section 2.2.2) that specifies the identity of this object.

jci (10 bits): An unsigned integer that specifies the value of the **JCID.index** field of the object. MUST be 0x01.

odcs (4 bits): An unsigned integer that specifies whether the data contained by this object is encrypted. MUST be zero.

A - fReserved1 (2 bits): An unsigned integer that MUST be zero, and MUST be ignored.

B - fHasOidReferences (1 bit): Specifies whether this object contains references to other objects.

C - fHasOsidReferences (1 bit): Specifies whether this object contains references to object spaces (section 2.1.4). MUST be zero.

fReserved2 (30 bits): An unsigned integer that MUST be zero, and MUST be ignored.

2.6.16 ObjectDeclaration2Body

The **ObjectDeclaration2Body** structure specifies the identity of an object (section 2.1.5) and the type of data the object contains.

0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
oid																															
jcid																															
A	B	fReserved2																													

oid (4 bytes): A **CompactID** (section 2.2.2) that specifies the identity of this object.

jcid (4 bytes): A **JCID** (section 2.6.14) that specifies the type of data this object contains.

A - fHasOidReferences (1 bit): A bit that specifies whether this object contains references to other objects.

B - fHasOsidReferences (1 bit): A bit that specifies whether this object contains references to object spaces (section 2.1.4) or contexts (section 2.1.11).

fReserved2 (6 bits): MUST be zero, and MUST be ignored.

2.7 Transmission by Using the File Synchronization via SOAP Over HTTP Protocol

Revision store files can be transmitted using the File Synchronization via SOAP over HTTP Protocol, as specified in [MS-FSSHTTP]. The following sections specify the elements specific to revision store files when transmitted over this protocol.

2.7.1 Storage Manifest

When a revision store file is transmitted over the File Synchronization via SOAP over HTTP Protocol, as specified in [MS-FSSHTTP], the following **Storage Manifest Data Element** structure (as specified in [MS-FSSHTTPB] section 2.2.1.12.3) MUST be used:

- **GUID** (storage manifest schema GUID)**:** A **GUID**, as specified in [MS-DTYP], that specifies the file type. MUST be one of the following values, depending on the file type.

File type	Value
.one	{1F937CB4-B26F-445F-B9F8-17E20160E461}
.onetoc2	{E4DBFD38-E5C7-408B-A8A1-0E7B421E1F5F}

The following root context and cell pairs are used:

Header cell (section 2.7.2) root specified by the following **Root Extended GUID** and **Cell ID:**

- **Root Extended GUID:** An **ExtendedGUID** structure used to contain the root identifier of the header cell (section 2.7.2). MUST be "{{1A5A319C-C26b-41AA-B9C5-9BD8C44E07D4} , 1}".
- **Cell ID:** A **Cell ID** structure (as specified in [MS-FSSHTTPB] section 2.2.1.10) where the **EXGUID1** field MUST be equal to "{{84DEFAB9-AAA3-4A0D-A3A8-520C77AC7073} , 1}" and the **EXGUID2** field MUST be equal to "{{111E4CF3-7FEF-4087-AF6A-B9544ACD334D} , 1}".

  Data root specified by the following **Root Extended GUID** and **Cell ID:**

- **Root Extended GUID:** An **ExtendedGUID** structure used to contain the root identifier of the root object space (section 2.1.4). MUST be the default root "{{84DEFAB9-AAA3-4A0D-A3A8-520C77AC7073} , 2}".
- **Cell ID:** A **Cell ID** structure (as specified in [MS-FSSHTTPB] section 2.2.1.10) where **EXGUID1** MUST be equal to "{{84DEFAB9-AAA3-4A0D-A3A8-520C77AC7073} , 1}", and **EXGUID2** MUST be equal to the revision store root object space identifier.

2.7.2 Header Cell

When a revision store file is transmitted using the File Synchronization via SOAP over HTTP Protocol, as specified in [MS-FSSHTTP], the revision store header (section 2.3.1) is transmitted as a single cell which MUST be referenced from the storage manifest of this revision store file (section 2.7.1).

The **Cell Manifest Current Revision Extended GUID** field of the cell's **Cell Manifest Data Element** structure, as specified in[MS-FSSHTTPB] section 2.2.1.12.4, SHOULD<9> be derived from the header (section 2.3.1) using the following method:

1. An **ExtendedGUID** structure (eguid1) is created with the **ExtendedGUID.guid** field value equal to the **Header.guidFile** field value and the **ExtendedGUID.n** field value equal to **Header.crcName** field value.
2. An **ExtendedGUID** structure (eguid2) is created with the **ExtendedGUID.guid** field value equal to the **Header.guidAncestor** field value and the **ExtendedGUID.n** field value equal to the **Header.ffvLastCodeThatWroteToThisFile** field value.
3. The value of the **Cell Manifest Current Revision Extended GUID** is "eguid1 XOR eguid2 XOR {{F5367D2F-F167-4830-A9C3-68F8336A2A09} , 0 }".

The **Revision Manifest Data Element** structure, as specified in [MS-FSSHTTPB] section 2.2.1.12.5, for this cell MUST be specified as follows:

- **Root Extended GUID**: MUST be "{{4A3717F8-1C14-49E7-9526-81D942DE1741}, 1}".
- **Object Extended GUID**: MUST be "{{B4760B1A-FBDF-4AE3-9D08-53219D8A8D21}, 1}".
- **Object Group Extended GUID**: MUST be calculated as **Cell Manifest Current Revision Extended GUID** specified previously XOR "{{26402DF1-68C7-43C0-A37C-04B9100893CE}, 0x80000000 }".

The root object specified by the **Object Extended GUID** field (described earlier) MUST be transmitted as a pair of **Object Declaration** and **Object Data** structures:

Object Declaration.PartitionID	Object Data.Data	Object Data.Object extended GUID array	Object Data.Cell ID array
1 (Object Data)	MUST be an **ObjectSpaceObjectPropSet** structure (section 2.6.1) with properties specified later in this section.	None	None

The **ObjectSpaceObjectPropSet** structure contained by the **Object Data.Data** field for the root object MUST contain the properties described in the following table.

Name	PropertyID	Value
FileIdentityGuid	0x1C001D94	A **GUID**, as specified by [MS-DTYP]. MUST be the value specified by the **Header.guidFile** field.
FileAncestorIdentityGuid	0x1C001D95	A GUID, as specified by [MS-DTYP]. MUST be the value specified by the **Header.guidAncestor** field.
FileLastCodeVersionThatWroteToIt	0x14001D99	An unsigned integer. MUST be the value specified by the **Header.ffvLastCodeThatWroteToThisFile** field.
FileNameCRC	0x14001D93	MUST be the **Header.crcName** field.

2.7.3 Cells

When a revision store file is transmitted using the File Synchronization via SOAP over HTTP Protocol, object spaces (section 2.1.4) are transmitted as cells as specified in [MS-FSSHTTP] section 1.3.

The **Cell ID** structure, as specified in [MS-FSSHTTPB] section 2.2.1.10, for this cell MUST be specified as follows:

- **EXGUID1**: MUST be equal to the identity of the context (section 2.1.11) of the active revision (section 2.1.8) of this object space except for the default context. The value of the default context data element of a revision store file is {{00000000-0000-0000-0000-000000000000}, 0}". This value is converted to "{{84DEFAB9-AAA3-4A0D-A3A8-520C77AC7073} , 1 }" when transmitted using the File Synchronization via SOAP over HTTP Protocol.
- **EXGUID2**: MUST be equal to the object space identifier (section 2.5.2).

2.7.4 Revisions

When a revision store file is transmitted using the File Synchronization via SOAP over HTTP Protocol, revision manifests (section 2.1.9) are transmitted as **Revision Manifest Data Element** structures, as specified in [MS-FSSHTTPB] section 2.2.1.12.5.

- **Revision ID:** MUST be equal to the revision store file revision identifier (section 2.1.8).
- **Base Revision ID:** MUST be equal to the revision store file dependency revision identifier (section 2.1.9).

The value of **RootObjectReference3FND.rootRole** field of a root object (section 2.1.7) from a revision in a revision store specifies **Root Extended GUID** field in the **Revision Manifest Data Element** structure:

Value of RootObjectReference3FND.rootRole	Meaning	Root extended GUID
0x00000001	Default content root	{ {4A3717F8-1C14-49E7-9526-81D942DE1741}, 1}
0x00000002	Metadata root	{ {4A3717F8-1C14-49E7-9526-

Value of RootObjectReference3FND.rootRole	Meaning	Root extended GUID
		81D942DE1741}, 2}
0x00000003	Encryption Key root	{4A3717F8-1C14-49E7-9526-81D942DE1741}, 3}
0x00000004	Version metadata root	{ {4A3717F8-1C14-49E7-9526-81D942DE1741}, 4}

The root object with **RootObjectReference3FND.rootRole** value set to 0x00000003 MUST be present only when the file is encrypted. (see section 2.7.7).

The **Object Extended GUID** field in the **Revision Manifest Data Element** structure MUST be equal to the identity of the corresponding root object in the revision in the revision store.

2.7.5 Object Groups

When a revision store file is transmitted using the File Synchronization via SOAP over HTTP Protocol, revision store object groups (section 2.1.13) are transmitted as **Object Group Data Element** structures, as specified in [MS-FSSHTTPB] section 2.2.1.12.6.

The **Revision Manifest Data Element** structure, as specified in [MS-FSSHTTPB] section 2.2.1.12.5, that references an object group MUST specify the object group extended GUID to be equal to the revision store object group identifier.

2.7.6 Objects

When a revision store file is transmitted using the File Synchronization via SOAP over HTTP Protocol, revision store objects (section 2.1.5) are transmitted as the following structures:

- The **Object Declaration** structure, as specified in [MS-FSSHTTPB] section 2.2.1.12.6.1.
- The **Object Data BLOB Declaration** structure, as specified in [MS-FSSHTTPB] section 2.2.1.12.6.2.
- The **Object Data** structure, as specified in [MS-FSSHTTPB] section 2.2.1.12.6.4
- The **Object Data BLOB Reference** structure, as specified in [MS-FSSHTTPB] section 2.2.1.12.6.5.

The **Object Extended GUID** field in the object declaration structures MUST be equal to the object identifier of the corresponding revision store object.

Objects that have the **JCID.IsFileData** field value equal to **false** are transmitted as two pairs of **Object Declaration** and **Object Data** structures in the specified order:

Object Declaration.PartitionID	Object Data.Data	Object Data.Object Extended GUID array	Object Data.Cell ID array
4 (Static Object MetaData)	An unsigned integer that specifies the **JCID** (section 2.6.14) of the object.	None.	None.
1 (Object Data)	MUST be an **ObjectSpaceObjectPropSet** structure (section 2.6.1).	Identifiers of the referenced objects in the revision store.	Identifiers of the referenced object spaces in the revision store.

Objects that have the **JCID.IsFileData** field value equal to **true** are transmitted as two pairs of **Object Declaration** and **Object Data** structures and one pair of **Object Data BLOB Declaration** and **Object Data BLOB Reference** structures in the specified order:

Object Declaration.PartitionID	**Object Data.Data**	**Object Data.Object Extended GUID array**	**Object Data.Cell ID array**
4 (Static Object MetaData)	An unsigned integer that specifies the **JCID** (section 2.6.14) of the object.	None	None
1 (Object Data)	MUST be an **ObjectSpaceObjectPropSet** structure (section 2.6.1) with properties specified later in this section.	None	None

Object Data BLOB Declaration.PartitionID	**Object Data BLOB Reference. BLOB Extended GUID**	**Object Data BLOB Reference.Object Extended GUID array**	**Object Data BLOB Reference.Cell ID array**
2 (File Data)	MUST have a reference to an **Object Data BLOB Data Element** structure, as specified in [MS-FSSHTTPB] section 2.2.1.12.8, used to transmit the data of the **file data object**.	None	None

The **ObjectSpaceObjectPropSet** structure contained by the **Object Data.Data** field for an object with **JCID.IsFileData** field equal to "true" MUST contain the following properties:

Name	**PropertyID**	**Value**
FileDataObject_GUID	0x1C00343E	A **GUID**, as specified by [MS-DTYP]. This property MUST be set only if the prefix specified by the **ObjectDeclarationFileData3RefCountFND.FileDataReference** field (section 2.5.27) or **ObjectDeclarationFileData3LargeRefCountFND.FileDataReference** field (section 2.5.28) is not <invfdo>. This property MUST be the GUID value of the remaining part of the **curly braced GUID string** in the **FileDataReference** field.
FileDataObject_InvalidData	0x0800343D	A Boolean value that specifies whether this file data object is not valid. MUST be **true** if the prefix specified by the **ObjectDeclarationFileData3RefCountFND.FileDataReference** field or the **ObjectDeclarationFileData3LargeRefCountFND.FileDataReference** field is <invfdo>.
FileDataObject_Extension	0x1C003424	A **Unicode** string that specifies the extension of the file data object. MUST be the value specified by the **ObjectDeclarationFileData3RefCountFND.Extension** field or the **ObjectDeclarationFileData3LargeRefCountFND.Extension** field.

2.7.7 Encryption

When a revision store file is transmitted by using the File Synchronization via SOAP over HTTP Protocol, a revision store file object space encryption data (see section 2.5.19) is transmitted as an object specified by a pair of **Object Declaration** ([MS-FSSHTTPB] section 2.2.1.12.6.1) and **Object Data** ([MS-FSSHTTPB] section 2.2.1.12.6.4) structures:

Object Declaration.PartitionID	Object Data.Data	Object Data.Object extended GUID array	Object Data.Cell ID array
1 (Object Data)	MUST be encryption data	None	None

Every transmitted base revision of a revision store MUST contain an object group (see section 2.7.5) with this object. The object MUST be the only one in the object group, and the value of the object group identifier MUST be set to the revision identifier XOR "{{0FE4F06F-BB18-4F79-B3D6-BC157D533D23}, 0x00000000 }".

The value of the object identifier MUST be the set to the value of the revision identifier XOR "{{06357FE8-CBC7-4DD1-AA81-09BC1695B19B}, 0x00000000 }".

2.7.8 Mapping Table

If you are reading a revision store file encoded using the File Synchronization via SOAP Over HTTP Protocol, then follow these steps to create a mapping table in memory.

1. Read **Object** field from **Object Data** ([MS-FSSHTTPB] section 2.2.1.12.6.4) **Extended GUID Array** and the **CompactID** array in order to populate Mapping Table, ignore nulls when populating.
2. Read **Cell ID Array** ([MS-FSSHTTPB] section 2.2.1.12.6.4) and the **ContextID (**section 2.6.6**)** where **Cell ID** is different than current **Cell ID** in Order and append them to the Mapping Table, ignore nulls when populating.
3. Read **Cell ID Array** ([MS-FSSHTTPB] section 2.2.1.12.6.4) and **OSID (**section 2.1.4**)** where **Cell ID** is same as current **Cell ID** and append them to the Mapping Table, ignore nulls when populating.

Each mapping table is a two-column table with the **CompactID** in first column and **ExtendedGuid**/**CellID** in the second column, as shown in the empty table, below:

CompactID	ExtendedGuid/CellID

2.8 Alternative Encoding Using the File Synchronization via SOAP Over HTTP Protocol

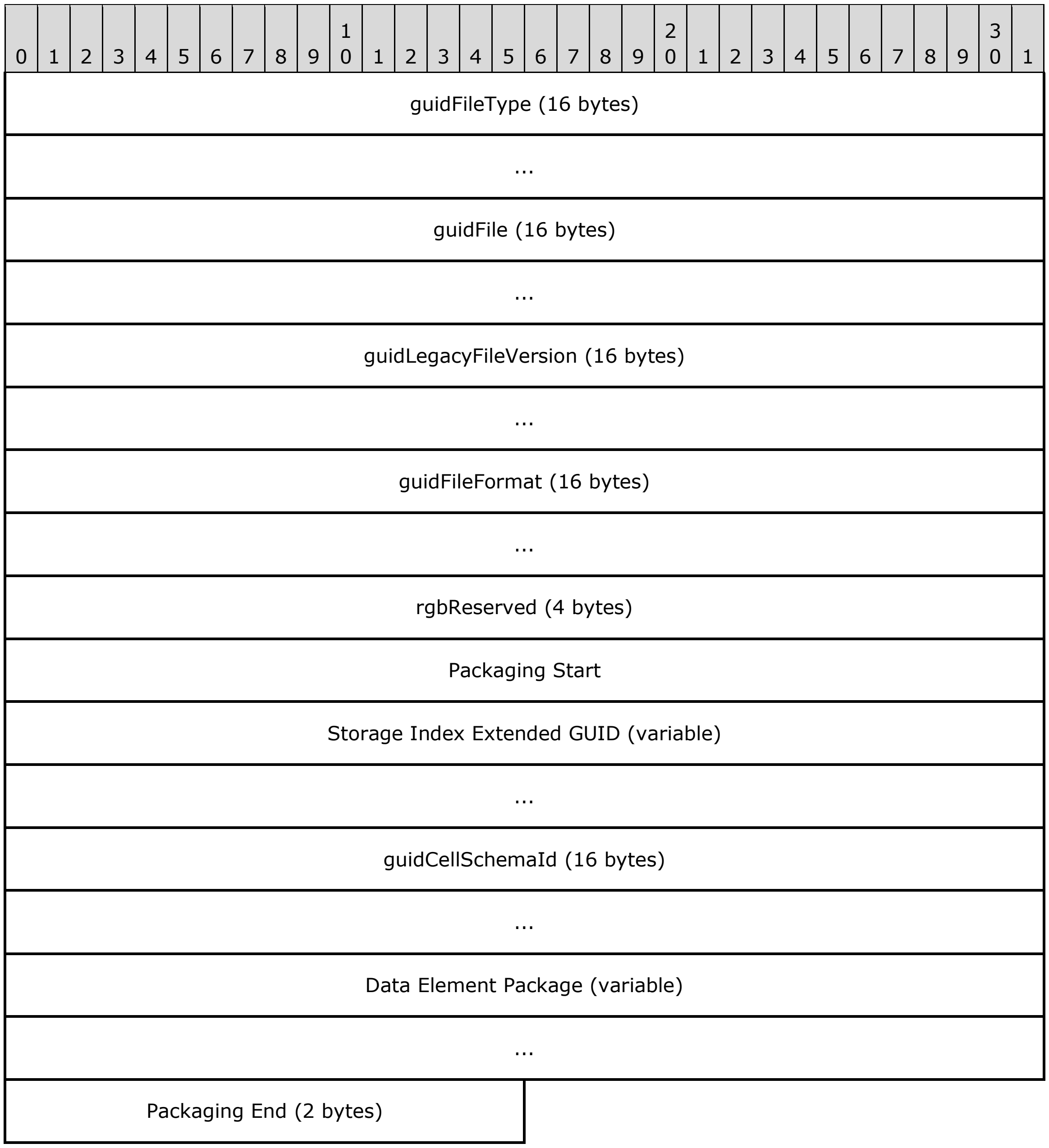
The alternative packaging format is used as the file format for .one and .onetoc2 files stored on a server. The alternative packaging format is effectively an envelope format that contains the equivalent of the results of an **Query Changes** sub-request, as specified in [MS-FSSHTTPB] section 2.2.2.1.3, for the full contents of the file.

When a revision store file is encoded using the File Synchronization via SOAP Over HTTP Protocol, specified in [MS-FSSHTTP], the encoded file contains a packaging structure that contains a **data element package**, as specified in [MS-FSSHTTPB] section 2.2.1.12, that specifies the serialized data elements that comprise the revision store file as specified in section 2.7.

2.8.1 Packaging Structure

The **Packaging** structure MUST be at the beginning of the file.

This structure has the following format.



guidFileType (16 bytes): A **GUID**, as specified by [MS-DTYP], MUST be "{7B5C52E4-D88C-4DA7-AEB1-5378D02996D3}".

guidFile (16 bytes): A GUID, as specified by [MS-DTYP], that specifies the identity of this package store file. SHOULD be globally unique.

guidLegacyFileVersion (16 bytes): A GUID, as specified by [MS-DTYP], that specifies the version of this package store file. SHOULD be globally unique.<10>

guidFileFormat (16 bytes): A GUID, as specified by [MS-DTYP], that specifies that the file is a package store file. MUST be "{638DE92F-A6D4-4bc1-9A36-B3FC2511A5B7}".

rgbReserved (4 bytes): MUST be zero. MUST be ignored.

Packaging Start (4 bytes): A 32-bit stream object header, as specified in [MS-FSSHTTPB] section 2.2.1.5.2, with a **Header Type** of 0x00, **Compound** of 0x1, **Type** of 0x7A, and **Length** set to the length of the **Storage Index Extended GUID** plus the length of the **guidCellSchemaId**.

Storage Index Extended GUID (variable): An **extended GUID**, as specified in [MS-FSSHTTPB] section 2.2.1.7) that specifies the storage index.

guidCellSchemaId (16 bytes): A GUID, as specified in [MS-DTYP], that specifies the file type. MUST be one of the following values, depending on the file type.

File type	Value
.one	{1F937CB4-B26F-445F-B9F8-17E20160E461}
.onetoc2	{E4DBFD38-E5C7-408B-A8A1-0E7B421E1F5F}

Data Element Package (variable): A **data element package**, as specified in [MS-FSSHTTPB] section 2.2.1.12, that specifies the serialized data elements.

Packaging End (2 bytes): A 16-bit stream object header end, as specified in [MS-FSSHTTPB] section 2.2.1.5.4, that specifies a stream object of type 0x7A. The value of this field MUST be 0x1EB.

3 Structure Examples

The following examples illustrate a typical of a OneNote Revision Store. All data is taken from a single .one file containing the data for a section, with a single page with two lines of text and an embedded file.

To navigate through the file, and to see how the pieces fit together, read it in the following order:

1. Read the file header (section 2.3.1) to determine if it is a .one section file or a .onetoc2 table of contents file. Inside the file header, find the locations in the file of the root file node list and of the transaction log (section 2.3.3).
2. Read the root file node list to find the identity and location of the section and page Object Spaces (section 2.1.4), as well as the file data store in which embedded file data objects (section 2.6.13) are stored, if any.
3. Starting with the root object space, read the object space manifest list (section 2.1.6) to get the location of the object space revision manifest list (section 2.1.10).
4. Read the revision manifest list to find the current revision (section 2.1.8) of the object space, including the choices of root objects (section 2.1.7), and either the set of objects (section 2.1.5) directly, or the locations of object group lists (section 2.1.13).
5. Read the object group lists to enumerate each of the objects, and to get the locations of the objects property sets (section 2.6.7).
6. Read each of the objects' properties. Discover the relationships between objects, and between objects and child object spaces and contexts (section 2.1.11).
7. Return to the root file node list to find the locations of those other Object Spaces, and repeat the process.
8. Finally, steps 2 through 7 require consulting the transaction log, to make sure that only the active portions of each file node list are read.

For brevity, several simplifications have been made when describing these examples. Some structures are expanded and described in fine detail in the first or second example, and then are merely summarized when they appear again in later examples.<11>

3.1 File Header

The following is an example of a **Header** structure (section 2.3.1) for a .one file.

Offset	Size	Structure	Value
000001F8	0400	FileNodeHeader - **FileNodeHeader**	
00000000	0010	GUID - **guidFileType**	{7B5C52E4-D88C-4DA7-AEB1-5378D02996D3}
00000010	0010	GUID - **guidFile**	{8DFAB807-A83A-4AC0-9393-E1DD0691CFB8}
00000020	0010	GUID - **guidLegacyFileVersion**	{00000000-0000-0000-0000-000000000000}
00000030	0010	GUID - **guidFileFormat**	{109ADD3F-911B-49F5-A5D0-1791EDC8AED8}
00000040	0004	FileFormatVersion - **ffvLastCodeThatWroteToThisFile**	42
00000044	0004	FileFormatVersion - **ffvOldestCodeThatHasWrittenToThisFile**	42

Offset	Size	Structure	Value
00000048	0004	FileFormatVersion - **ffvNewestCodeThatHasWrittenToThisFile**	42
0000004C	0004	FileFormatVersion - **ffvOldestCodeThatMayReadThisFile**	42
00000050	0008	FileChunkReference32 - **fcrLegacyFreeChunkList**	
00000050	0004	ULONG - **stp**	0x00000000
00000054	0004	ULONG - **cb**	0x00000000
00000058	0008	FileChunkReference32 - **fcrLegacyTransactionLog**	
00000058	0004	ULONG - **stp**	0xFFFFFFFF
0000005C	0004	ULONG - **cb**	0x00000000
00000060	0004	ULONG - **cTransactionsInLog**	15
00000064	0004	ULONG - **cbLegacyExpectedFileLength**	0
00000068	0008	UINT64 - **rgbPlaceholder**	0
00000070	0008	FileChunkReference32 - **fcrLegacyFileNodeListRoot**	
00000070	0004	ULONG - **stp**	0xFFFFFFFF
00000074	0004	ULONG - **cb**	0x00000000
00000078	0004	ULONG - **cbLegacyFreeSpaceInFreeChunkList**	0
0000007C	0001	BYTE - **fNeedsDefrag**	0
0000007D	0001	BYTE - **fRepairedFile**	0
0000007E	0001	BYTE - **fNeedsGarbageCollect**	0
0000007F	0001	BYTE - **fHasNoEmbeddedFileObjects**	0
00000080	0010	GUID - **guidAncestor**	{E2FA6D4F-08F1-4E6B-B156-0EDFB16FB083}
00000090	0004	ULONG - **crcName**	0xCEBE8422
00000094	000C	FileNodeListFragmentReference64x32 - **fcrHashedChunkList**	
00000094	0008	UINT64 - **stp**	0x00000000000012A8
00000098	0004	ULONG - **cb**	0x00000400
000000A0	000C	TransactionLogFragmentReference64x32 - **fcrTransactionLog**	
000000A0	0008	UINT64 - **stp**	0x0000000000000800
000000A8	0004	ULONG - **cb**	0x00000400
000000AC	000C	FileNodeListFragmentReference64x32 - **fcrFileNodeListRoot**	
000000AC	0008	UINT64 - **stp**	0x0000000000000400
000000B4	0004	ULONG - **cb**	0x00000400

Offset	Size	Structure	Value
000000B8	000C	FreeChunkListChunkReference - **fcrFreeChunkList**	
000000B8	0008	UINT64 - **stp**	18446744073709551615
000000C0	0004	ULONG - **cb**	0x00000000
000000C4	0008	UINT64 - **cbExpectedFileLength**	0x0000000000003190
000000CC	0008	UINT64 - **cbFreeSpaceInFreeChunkList**	0x0000000000000000
000000D4	0010	GUID - **guidFileVersion**	{A9E957F9-A065-4E7D-AFB3-96B7311504DF}
000000E4	0008	UINT64 - **nFileVersionGeneration**	15
000000EC	0010	GUID - **guidDenyReadFileVersion**	{A8EE13E0-7E37-46D4-BA83-AB3248782D16}
000000FC	0004	ULONG - **grfDebugLogFlags**	0
00000100	000C	FileChunkReference64x32 - **fcrDebugLog**	
00000100	0008	UINT64 - **stp**	0x0000000000000000
00000108	0004	ULONG - **cb**	0x00000000
0000010C	000C	FileChunkReference64x32 - **fcrAllocVerificationFreeChunkList**	
0000010C	0008	UINT64 - **stp**	0x0000000000000000
00000114	0004	ULONG - **cb**	0x00000000
00000118	0004	BuildNumber - **bnCreated**	0x0FA129B4
0000011C	0004	BuildNumber - **bnLastWroteToThisFile**	0x0FA129B4
00000120	0004	BuildNumber - **bnOldestWritten**	0x0FA129B4
00000124	0004	BuildNumber - **bnNewestWritten**	0x0FA129B4
00000128	02D8	FixedBlob - **rgbReserved**	0

guidFileType: "{7B5C52E4-D88C-4DA7-AEB1-5378D02996D3}" specifies that the file format is .one, rather than .onetoc2.

guidFile: This GUID provides a unique identity for each file.

guidFileFormat: {109ADD3F-911B-49F5-A5D0-1791EDC8AED8} identifies this file as a OneNote Revision Store.

ffvLastCodeThatWroteToThisFile: 42 specifies that the file format is from Microsoft OneNote 2010.

ffvOldestCodeThatHasWrittenToThisFile: 42 specifies that the file format is from OneNote 2010.

ffvNewestCodeThatHasWrittenToThisFile: 42 specifies that the file format is from OneNote 2010.

ffvOldestCodeThatMayReadThisFile: 42 specifies that versions of OneNote older than OneNote 2010 cannot read this file.

cTransactionsInLog: 15 specifies that the first 15 entries in the transaction log are complete. Any additional entries are not yet ready and are ignored.

fHasNoEmbeddedFileObjects: Set to 0, even when the file indeed contains no embedded file objects.

guidAncestor: This GUID matches the **guidFile** field of the .onetoc2 file in the same directory as this .one file.

crcName: 0xCEBE8422 is the CRC that is derived from the section name "Example.one".

fcrHashedChunkList: Specifies that the hashed chunk list is located at offset 0x12a8 in the file, and is 0x400 bytes in length.

fcrTransactionLog: Specifies that the first fragment of the transaction log is located at offset 0x800 in the file, and is 0x400 bytes in length.

fcrFileNodeListRoot: Specifies that the first fragment of the root file node list is located at offset 0x400 in the file, and is 0x400 bytes in length.

fcrFreeChunkList: "fcrNil" specifies that there is no free chunk list in this file.

cbExpectedFileLength: 0x3190 -- matches the byte size of this file.

cbFreeSpaceInFreeChunkList: This field is zero, because the file contains no free chunk list.

guidFileVersion: This is a unique GUID, created when this file was last modified.

nFileVersionGeneration: 15 indicates the number of times this file has been modified. It coincides with **cTransactionsInLog**, so each modification adds a new log entry, but the two numbers do not need to be in sync.

guidDenyReadFileVersion: This is a unique GUID, created during a recent modification to the file.

bnCreated: 0x0FA129B4 indicates OneNote 2010 build number 14.0.4763.1000.

bnLastWroteToThisFile: 0x0FA129B4 indicates OneNote 2010 build number 14.0.4763.1000.

bnOldestWritten: 0x0FA129B4 indicates OneNote 2010 build number 14.0.4763.1000.

bnNewestWritten: 0x0FA129B4 indicates OneNote 2010 build number 14.0.4763.1000.

3.2 Root File Node List

Using the **fcrFileNodeListRoot** field from the preceding file header example, 0x400 bytes can be read from the file at offset 0x400. Those bytes can be read as a file node list. In this example, the list includes only four **FileNodes**:

- Two **ObjectSpaceManifestListReferences**, one for the section object space, and one for the page object space.
- One **ObjectSpaceManifestRoot**, which identifies the section object space as the root object space.
- One **FileDataStoreListReference**, which specifies where the data for embedded files is stored.

Offset	**Size**	**Structure**	**Value**
00000400	0400	FileNodeListFragment **- FileNodeListFragment**	
00000400	0010	FileNodeListHeader **- header**	
00000400	0008	UINT64 **- uintMagic**	11841787174971765956

Offset	Size	Structure	Value
00000408	0004	ULONG - **FileNodeListID**	0x00000010
0000040C	0004	ULONG - **nFragmentSequence**	0
00000410	001B	FileNode - **fnd[0]**	
00000410	0004	FileNodeHeader - **header**	
00000410	10 bits	ULONG - **FileNodeID**	0x008
00000410	13 bits	ULONG - **Size**	27
00000410	2 bits	ULONG - **StpFormat**	2
00000410	2 bits	ULONG - **CbFormat**	2
00000410	4 bits	ULONG - **BaseType**	2
00000410	1 bit	ULONG - **Reserved**	1
00000414	0017	ObjectSpaceManifestListReferenceFND - **fnd**	
00000414	0003	FileNodeListReferenceWithIndirection - **ref**	
00000414	0002	USHORT - **stp**	0x0180
00000416	0001	BYTE - **cb**	0x24
00000417	0014	ExtendedGUID - **gosid**	{{CFDC64CA-6425-4223-903F-0B7D8A6690F9},1}
0000042B	0018	FileNode - **fnd[1]**	
0000042B	0004	FileNodeHeader - **header**	
0000042B	10 bits	ULONG - **FileNodeID**	0x004
0000042B	13 bits	ULONG - **Size**	24
0000042B	2 bits	ULONG - **StpFormat**	1
0000042B	2 bits	ULONG - **CbFormat**	0
0000042B	4 bits	ULONG - **BaseType**	0
0000042B	1 bit	ULONG - **Reserved**	1
0000042F	0014	ObjectSpaceManifestRootFND - **fnd**	
0000042F	0014	ExtendedGUID - **gosid**	{{CFDC64CA-6425-4223-903F-0B7D8A6690F9},1}
00000443	001B	FileNode - **fnd[2]**	
00000443	0004	FileNodeHeader - **header**	
00000443	10 bits	ULONG - **FileNodeID**	0x008
00000443	13 bits	ULONG - **Size**	27
00000443	2 bits	ULONG - **StpFormat**	2

Offset	Size	Structure	Value
00000443	2 bits	ULONG - **CbFormat**	2
00000443	4 bits	ULONG - **BaseType**	2
00000443	1 bit	ULONG - **Reserved**	1
00000447	0017	ObjectSpaceManifestListReferenceFND **- fnd**	
00000447	0003	FileNodeListReferenceWithIndirection **- ref**	
00000447	0002	USHORT **- stp**	0x020D
00000449	0001	BYTE **- cb**	0x24
0000044A	0014	ExtendedGUID **- gosid**	{{418BCA1D-C31E-4342-966B-B7949DA56A9F},1}
0000045E	0007	FileNode **- fnd[3]**	
0000045E	0004	FileNodeHeader **- header**	
0000045E	10 bits	ULONG **- FileNodeID**	0x090
0000045E	13 bits	ULONG **- Size**	7
0000045E	2 bits	ULONG **- StpFormat**	2
0000045E	2 bits	ULONG **- CbFormat**	2
0000045E	4 bits	ULONG **- BaseType**	2
0000045E	1 bit	ULONG **- Reserved**	1
00000462	0003	FileDataStoreListReferenceFND **- fnd**	
00000462	0003	FileNodeListReferenceWithIndirection **- ref**	
00000462	0002	USHORT **- stp**	0x0509
00000464	0001	BYTE **- cb**	0x24
00000465	0387	Blob **- padding**	0
000007EC	000C	FileNodeListReferenceWithIndirection **- nextFragment**	fcrNil
000007F8	0008	UINT64 **- footer**	10070635646201084491

header.FileNodeListID: 0x00000010 specifies that an identifier for this file node list. All **FileNodeListFragments** that are part of this list need to share this identifier.

header.nFragmentSequence: Zero specifies that this is the first fragment in this file node list.

fnd[0]: This **FileNode** establishes that there is an object space and gives the location in the file in which the object space’s revision manifest list resides.

fnd[0].header.FileNodeID: 0x008 specifies that this **FileNode** is of type "ObjectSpaceManifestListReferenceFND".

fnd[0].header.Size: 27 specifies that in addition to the 4 bytes of **FileNodeHeader**, there are 23 bytes of data. The next **FileNode**, if any, occurs immediately afterwards.

fnd[0].header.StpFormat: 2 specifies that if this FileNode type contains a file offset reference to another part of the file, the reference is compressed into 2 bytes, and the actual byte offset is 8 times that value.

fnd[0].header.CbFormat: 2 specifies that if this **FileNode** type contains a file offset reference to another part of the file, the size is compressed into 1 byte, and the actual byte length is 8 times that value.

fnd[0].header.BaseType: 2 specifies that this **FileNode** does indeed contain a file offset reference to another part of the file, and that a file node list is found at that offset.

fnd[0].fnd.ref: Specifies that the first fragment of the object space manifest file node list is located at offset 0xC00 in the file (0x180 ×8), and is 0x120 bytes in length (0x24 ×8).

fnd[0].fnd.gosid: {CFDC64CA-6425-4223-903F-0B7D8A6690F9},1} specifies the identity of the object space.

fnd[1]: This **FileNode** identifies the object space specified in **fnd**[0] as the root object space for this file.

fnd[1].header: Specifies that the **FileNode** is of type **ObjectSpaceManifestRootFND**, that it contains 20 bytes of data beyond the header, and that it does not contain a reference to another part of the file.

fnd[1].fnd.gosid: {{CFDC64CA-6425-4223-903F-0B7D8A6690F9},1} specifies that the object space identified by this gosid is the root object space for the entire file. This identity matches the gosid in the previous **ObjectSpaceManifestListReferenceFND** FileNode.

fnd[2]: This **FileNode** establishes that there is another object space, and gives the location in the file where the object space's revision manifest list resides.

fnd[2].header: Specifies that the **FileNode** is of type "ObjectSpaceManifestListReferenceFND", that it contains 23 bytes of data beyond the header, that the data begins with a compressed reference to another part of the file, and that a file node list can be found at that other file location.

fnd[2].fnd.ref: Specifies that the first fragment of the object space manifest file node list is located at offset 0x1068 in the file, and is 0x120 bytes in length.

fnd[2].fnd.gosid: {{418BCA1D-C31E-4342-966B-B7949DA56A9F},1} specifies the identity of this second object space.

fnd[3]: This **FileNode** establishes that there is file data store in the file, and gives its location in the file.

fnd[3].header: Specifies that the **FileNode** is of type "FileDataStoreListReferenceFND", that it contains 3 bytes of data beyond the header, that the data begins with a compressed reference to another part of the file, and that a file node list can be found at that other file location.

fnd[3].fnd.ref: Specifies that the first fragment of the file data store list is located at offset 0x2848 in the file, and is 0x120 bytes in length.

nextFragment: "fcrNil" specifies that there are no additional fragments in this file node list.

3.3 Root Object Space

In the preceding Root File Node List example, the root object space was identified as **gosid** set to "{{CFDC64CA-6425-4223-903F-0B7D8A6690F9},1}". Because this is a .one file, the root object space is the data for a OneNote 2010 section.

To gather the data for this section, read the object space manifest list for that object space. That indicates the location of the active revision manifest list. The revision manifest list, in turn, indicates a collection of revision manifests, including references to object group lists that contain all the object nodes in the section. Finally, the objects have references to their properties.

The object space manifest list starts with an **ObjectSpaceManifestListStartFND** file node, which confirms the **gosid** of the object space. It then contains one or more **RevisionManifestListReferenceFND** file nodes. If there is more than one, ignore all but the last one. In this example, there is only one.

Offset	Size	Structure	Value
00000C00	0120	FileNodeListFragment - **FileNodeListFragment**	
00000C00	0010	FileNodeListHeader - **header**	
00000C00	0008	UINT64 - **uintMagic**	11841787174971765956
00000C08	0004	ULONG - **FileNodeListID**	0x00000011
00000C0C	0004	ULONG - **nFragmentSequence**	0
00000C10	0018	FileNode - **fnd[0]**	
00000C10	0004	FileNodeHeader - **header**	0x0880600C
00000C14	0014	ObjectSpaceManifestListStartFND - **fnd**	
00000C14	0014	ExtendedGUID - **gosid**	{{CFDC64CA-6425-4223-903F-0B7D8A6690F9},1}
00000C28	0007	FileNode - **fnd[1]**	
00000C28	0004	FileNodeHeader - **header**	0x95001C10
00000C2C	0003	RevisionManifestListReferenceFND - **fnd**	
00000C2C	0003	FileNodeListReferenceWithIndirection - **ref**	
00000C2C	0002	USHORT - **stp**	0x01A4
00000C2E	0001	BYTE - **cb**	0x24
00000C2F	00F1	Blob - **padding and footer**	...

fnd[0].header: 0x0880600C specifies that the **FileNode** is of type "ObjectSpaceListStartFND", that it contains 20 bytes of data beyond the header, and that it does not contain a reference to another part of the file.

fnd[0].fnd.gosid: {{CFDC64CA-6425-4223-903F-0B7D8A6690F9},1} specifies the identity of the object space. It matches the **gosid** specified by the **ObjectSpaceManifestListReferenceFND** that directed it here.

fnd[1].header: 0x95001C10 specifies that the **FileNode** is of type "RevisionManifestListReferenceFND", that it contains 3 bytes of data beyond the header, that the data begins with a compressed reference to another part of the file, and that a file node list can be found at that other file location.

fnd[1].fnd.ref: Specifies that the first fragment of the revision manifest list is located at offset 0xD20 in the file, and is 0x120 bytes in length.

Following the file offset to the object space’s revision manifest list, there is potentially a long list of **FileNodes** that create and manage multiple revisions of the object space. Some of the **FileNodes** might be grouped into sequences that together specify a revision. Other **FileNodes** might modify or extend an existing revision. In this example, there is just one sequence, which specifies a single revision for the section object space.

Offset	Size	Structure	Value
00000D20	0120	FileNodeListFragment - **FileNodeListFragment**	
00000D20	0010	FileNodeListHeader - **header**	
00000D20	0008	UINT64 - **uintMagic**	11841787174971765956
00000D28	0004	ULONG - **FileNodeListID**	0x00000012
00000D2C	0004	ULONG - **nFragmentSequence**	0
00000D30	001C	FileNode - **fnd[0]**	
00000D30	0004	FileNodeHeader - **header**	0x80807014
00000D34	0018	RevisionManifestListStartFND - **fnd**	
00000D34	0014	ExtendedGUID - **gosid**	{{CFDC64CA-6425-4223-903F-0B7D8A6690F9},1}
00000D48	0004	ULONG - **nInstance**	
00000D4C	0032	FileNode - **fnd[1]**	
00000D4C	0004	FileNodeHeader - **header**	0x8080C81E
00000D50	002E	RevisionManifestStart6FND - **fnd**	
00000D50	0014	ExtendedGUID - **rid**	{{082ED24F-9AA5-4DB8-875D-3E1FEE0AD4F4},1}
00000D64	0014	ExtendedGUID - **ridDependent**	{{00000000-0000-0000-0000-000000000000},0}
00000D78	0004	ULONG - **RevisionRole**	1
00000D7C	0002	USHORT - **odcsDefault**	0x0000
00000D7E	001B	FileNode - **fnd[2]**	
00000D7E	0004	FileNodeHeader - **header**	0x95006CB0
00000D82	0017	ObjectGroupListReferenceFND - **fnd**	
00000D82	0003	FileNodeListReferenceWithIndirection - **ref**	
00000D82	0002	USHORT - **stp**	0x01E9
00000D84	0001	BYTE - **cb**	0x24
00000D85	0014	ExtendedGUID - **ObjectGroupID**	{{2A83AE62-6754-4383-8E2B-4033FF3CFBA1},1}
00000D99	001C	FileNode - **fnd[3]**	
00000D99	0004	FileNodeHeader - **header**	0x8080705A
00000D9D	0018	RootObjectReference3FND - **fnd**	

Offset	**Size**	**Structure**	**Value**
00000D9D	0014	ExtendedGUID **- oidRoot**	{{9A416C35-236A-49A1-91FE-BD73076400A4},11}
00000DB1	0004	ULONG **- RootRole**	2
00000DB5	001C	FileNode **- fnd[4]**	
00000DB5	0004	FileNodeHeader **- header**	0x8080705A
00000DB9	0018	RootObjectReference3FND **- fnd**	
00000DB9	0014	ExtendedGUID **- oidRoot**	{{9A416C35-236A-49A1-91FE-BD73076400A4},10}
00000DCD	0004	ULONG **- RootRole**	1
00000DD1	0004	FileNode **- fnd[5]**	
00000DD1	0004	FileNodeHeader **- header**	0x8080101C
00000DD5	006B	Blob **- padding and footer**	...

fnd[0].header: 0x80807014 specifies that the **FileNode** is of type "RevisionManifestListStartFND", that it contains 24 bytes of data beyond the header, and that it does not contain a reference to another part of the file.

fnd[0].fnd.gosid: {{CFDC64CA-6425-4223-903F-0B7D8A6690F9},1} specifies yet again the identity of the object space. It matches the **gosid** specified by the **ObjectSpaceManifestListReferenceFND** and **ObjectSpaceManifestListStartFND** that directed it here.

fnd[1].header: 0x8080C81E specifies that the **FileNode** is of type "RevisionManifestStart6FND", that it contains 46 bytes of data beyond the header, and that it does not contain a reference to another part of the file.

fnd[1].fnd.rid: {{082ED24F-9AA5-4DB8-875D-3E1FEE0AD4F4},1} specifies the identity of the revision formed by this revision manifest.

fnd[1].fnd.ridDependent: "{{00000000-0000-0000-0000-000000000000},0" specifies that there is no dependent revision. All of the objects needed to make a valid object space revision are declared in this **FileNode** sequence.

fnd[1].fnd.RevisionRole: 1 specifies that the revision created from this manifests labels itself the active content for this object space. It is all right if other revision manifests also specify **RevisionRole** 1. In that case, the actual active content is the last revision in the revision manifest list which receives that label.

fnd[1].fnd.odcsDefault: 0x0000 specifies that the object properties for objects declared by this revision manifest are not encrypted.

fnd[2].header: 0x95006CB0 specifies that the **FileNode** is of type "ObjectGroupListReferenceFND", that it contains 23 bytes of data beyond the header, that the data begins with a compressed reference to another part of the file, and that a file node list can be found at that other file location.

fnd[2].fnd.ref: Specifies that the first fragment of the object group list is located at offset 0xF48 in the file, and is 0x120 bytes in length

fnd[2].fnd.ObjectGroupID: {{2A83AE62-6754-4383-8E2B-4033FF3CFBA1},1} specifies the identity of the object group.

fnd[3].header: 0x8080705A specifies that the **FileNode** is of type "RootObjectReference3FND", that it contains 24 bytes of data beyond the header, and that the data does not contain a reference to another part of the file.

fnd[3].fnd.oidRoot: {{9A416C35-236A-49A1-91FE-BD73076400A4},11} specifies the identity of one of the root objects of this revision. The object might or might not be declared in this revision, because it could be declared in a dependent revision. In this example, the object is declared in the object group list referenced previously.

fnd[3].fnd.RootRole: 2 specifies that the object is the metadata root.

fnd[4].header: 0x8080705A specifies that the **FileNode** is of type "RootObjectReference3FND", that it contains 24 bytes of data beyond the header, and that the data does not contain a reference to another part of the file.

fnd[4].fnd.oidRoot: {{9A416C35-236A-49A1-91FE-BD73076400A4},10} specifies the identity of another root object for this revision.

fnd[4].fnd.RootRole: 1 specifies that the object is the default content root.

fnd[5].header: 0x8080101C specifies that the **FileNode** is of type "RevisionManifestEndFND", and that it contains zero bytes of data beyond the header. If there were more **FileNodes** in the list after this one, they would not be considered part of the revision manifest declared by **fnd[1]**.

3.4 Section Object Space Objects

An object group list contains two types of **FileNodes**: those that define a Global ID Table, and those that declare Object Space Objects.

The section (root) object space in this example contains four objects. They are not declared directly within the revision manifest list, although they could be, and are in .onetoc2 files. Instead, they are declared in a separate object group list.

The four objects are:

- The Section node, which is the content root. It is responsible for tracking the set of page series and their relative ordering.
- A Page Series node, which corresponds to one top-level page. A page series can also track additional lower-level pages, but this example does not contain any.
- A Section metadata, which is the metadata root. The metadata tracks the schema version that was used to serialize this object space, as well section properties such as color that are commonly frequently by higher levels such as notebooks.
- A Page metadata, which ought to be a clone of the page metadata found in the actual page object space. OneNote 2010 caches copies of the metadata of child object spaces in parent object spaces.

Offset	Size	Structure	Value
00000F48	0120	FileNodeListFragment - **FileNodeListFragment**	
00000F48	0010	FileNodeListHeader - **header**	
00000F48	0008	UINT64 - **uintMagic**	11841787174971765956
00000F50	0004	ULONG - **FileNodeListID**	0x00000013

Offset	Size	Structure	Value
00000F54	0004	ULONG - **nFragmentSequence**	0
00000F58	0018	FileNode - **fnd[0]**	
00000F58	0004	FileNodeHeader - **header**	0x808060B4
00000F5C	0014	ObjectGroupStartFND - **fnd**	
00000F5C	0014	ExtendedGUID - **oid**	{{2A83AE62-6754-4383-8E2B-4033FF3CFBA1},1}
00000F70	0004	FileNode - **fnd[1]**	
00000F70	0004	FileNodeHeader - **header**	0x80801022
00000F74	0018	FileNode - **fnd[2]**	
00000F74	0004	FileNodeHeader - **header**	0x80806024
00000F78	0014	GlobalIdTableEntryFNDX - **fnd**	
00000F78	0004	ULONG - **index**	0
00000F7C	0010	GUID - **guid**	{63230A2C-F51E-01AC-217F-603847815F77}
00000F8C	0018	FileNode - **fnd[3]**	
00000F8C	0004	FileNodeHeader - **header**	0x80806024
00000F90	0014	GlobalIdTableEntryFNDX - **fnd**	
00000F90	0004	ULONG - **index**	1
00000F94	0010	GUID - **guid**	{418BCA1D-C31E-4342-966B-B7949DA56A9F}
00000FA4	0018	FileNode - **fnd[4]**	
00000FA4	0004	FileNodeHeader - **header**	0x80806024
00000FA8	0014	GlobalIdTableEntryFNDX - **fnd**	
00000FA8	0004	ULONG - **index**	2
00000FAC	0010	GUID - **guid**	{9A416C35-236A-49A1-91FE-BD73076400A4}
00000FBC	0004	FileNode - **fnd[5]**	
00000FBC	0004	FileNodeHeader - **header**	0x80801028
00000FC0	0018	FileNode - **fnd[6]**	
00000FC0	0004	FileNodeHeader - **header**	0x8080608C
00000FC4	0014	DataSignatureGroupDefinitionFND - **fnd**	
00000FC4	0014	ExtendedGUID - **DataSignatureGroup**	{{7257E098-D9C8-4F9F-9EE1-E41D6AEF9D46},1}
00000FD8	0011	FileNode - **fnd[7]**	
00000FD8	0004	FileNodeHeader - **header**	0x8D0044A4

Offset	Size	Structure	Value
00000FDC	000D	ObjectDeclaration2RefCountFND - **fnd**	
00000FDC	0003	BlobReferenceWithIndirection - **ref**	
00000FDC	0002	USHORT - **stp**	0x01E5
00000FDE	0001	BYTE - **cb**	0x04
00000FDF	0009	ObjectDeclaration2Body - **body**	
00000FDF	0004	CompactID - **oid**	
00000FDF	0001	BYTE - **n**	11
00000FE0	0003	ULONG - **guidIndex**	2
00000FE3	0004	JCID - **jcid**	
00000FE3	0002	USHORT - **index**	jcidSectionMetaData
00000FE5	0002	USHORT - **flags**	0x0002
00000FE7	1 bit	BYTE - **fHasOidReferences**	0
00000FE7	1 bit	BYTE - **fHasOsidReferences**	0
00000FE7	6 bits	BYTE - **fReserved2**	0
00000FE8	0001	BYTE - **cRef**	1
00000FE9	0011	FileNode - **fnd[8]**	
00000FE9	0004	FileNodeHeader - **header**	0x8D0044A4
00000FED	000D	ObjectDeclaration2RefCountFND - **fnd**	
00000FED	0003	BlobReferenceWithIndirection - **ref**	
00000FED	0002	USHORT - **stp**	0x01DE
00000FEF	0001	BYTE - **cb**	0x07
00000FF0	0009	ObjectDeclaration2Body - **body**	
00000FF0	0004	CompactID - **oid**	
00000FF0	0001	BYTE - **n**	10
00000FF1	0003	ULONG - **guidIndex**	2
00000FF4	0004	JCID - **jcid**	
00000FF4	0002	USHORT - **index**	jcidSectionNode
00000FF6	0002	USHORT - **flags**	0x0006
00000FF8	1 bit	BYTE - **fHasOidReferences**	1
00000FF8	1 bit	BYTE - **fHasOsidReferences**	0

Offset	Size	Structure	Value
00000FF8	6 bits	BYTE - **fReserved2**	0
00000FF9	0001	BYTE - **cRef**	1
00000FFA	0011	FileNode - **fnd[9]**	
00000FFA	0004	FileNodeHeader - **header**	0x8D0044A4
00000FFE	000D	ObjectDeclaration2RefCountFND - **fnd**	
00000FFE	0003	BlobReferenceWithIndirection - **ref**	
00000FFE	0002	USHORT - **stp**	0x01D1
00001000	0001	BYTE - **cb**	0x0D
00001001	0009	ObjectDeclaration2Body - **body**	
00001001	0004	CompactID - **oid**	
00001001	0001	BYTE - **n**	1
00001002	0003	ULONG - **guidIndex**	0
00001005	0004	JCID - **jcid**	
00001005	0002	USHORT - **index**	jcidPageMetaData
00001007	0002	USHORT - **flags**	0x0002
00001009	1 bit	BYTE - **fHasOidReferences**	0
00001009	1 bit	BYTE - **fHasOsidReferences**	0
00001009	6 bits	BYTE - **fReserved2**	0
0000100A	0001	BYTE - **cRef**	1
0000100B	0011	FileNode - **fnd[10]**	
0000100B	0004	FileNodeHeader - **header**	0x8D0044A4
0000100F	000D	ObjectDeclaration2RefCountFND - **fnd**	
0000100F	0003	BlobReferenceWithIndirection - **ref**	
0000100F	0002	USHORT - **stp**	0x01C8
00001011	0001	BYTE - **cb**	0x09
00001012	0009	ObjectDeclaration2Body - **body**	
00001012	0004	CompactID - **oid**	
00001012	0001	BYTE - **n**	12
00001013	0003	ULONG - **guidIndex**	2
00001016	0004	JCID - **jcid**	

Offset	Size	Structure	Value
00001016	0002	USHORT - **index**	jcidPageSeries
00001018	0002	USHORT - **flags**	0x0006
0000101A	1 bit	BYTE - **fHasOidReferences**	1
0000101A	1 bit	BYTE - **fHasOsidReferences**	1
0000101A	6 bits	BYTE - **fReserved2**	0
0000101B	0001	BYTE - **cRef**	1
0000101C	0004	FileNode - **fnd[11]**	
0000101C	0004	FileNodeHeader - **header**	0x808010B8
00001020	0048	Blob - **padding and footer**	...

fnd[0].header: 0x808060B4 specifies that the **FileNode** is of type "ObjectGroupStartFND", that it contains 20 bytes of data beyond the header, and that the data does not contain a reference to another part of the file.

fnd[0].fnd.oid: {{2A83AE62-6754-4383-8E2B-4033FF3CFBA1},1} specifies the identity of the object group. It matches the **ObjectGroupID** field in the **ObjectGroupListReferenceFND FileNode** in the previous example.

fnd[1].header: 0x80801022 specifies that the **FileNode** is of type "GlobalIdTableStart2FND", and that it contains zero bytes of data beyond the header.

fnd[2].header: 0x80806024 specifies that the **FileNode** is of type "GlobalIdTableEntryFNDX", that it contains 20 bytes of data beyond the header, and that the data does not contain a reference to another part of the file.

fnd[2].fnd.index: 0 specifies that **CompactIDs** in this object group that specify 0x0 for their index are to be matched with this.

fnd[2].fnd.guid: {63230A2C-F51E-01AC-217F-603847815F77} specifies that **CompactIDs** that specify 0x0 as their **guidIndex** can be converted to **ExtendedGUIDs** using this GUID.

fnd[3].header: 0x80806024 specifies that the **FileNode** is of type "GlobalIdTableEntryFNDX", that it contains 20 bytes of data beyond the header, and that the data does not contain a reference to another part of the file.

fnd[3].fnd.index: 1 specifies that **CompactIDs** in this object group that specify 0x1 for their index are to be matched with this.

fnd[3].fnd.guid: {418BCA1D-C31E-4342-966B-B7949DA56A9F} specifies that **CompactIDs** that specify 0x1 as their **guidIndex** can be converted to **ExtendedGUIDs** using this GUID.

fnd[4].header: 0x80806024 specifies that the **FileNode** is of type "GlobalIdTableEntryFNDX", that it contains 20 bytes of data beyond the header, and that the data does not contain a reference to another part of the file.

fnd[4].fnd.index: 2 specifies that **CompactIDs** in this object group that specify 0x2 for their index are to be matched with this.

fnd[4].fnd.guid: {9A416C35-236A-49A1-91FE-BD73076400A4} specifies that **CompactIDs** that specify 0x2 as their **guidIndex** can be converted to **ExtendedGUIDs** using this GUID.

fnd[5].header: 0x80801028 specifies that the **FileNode** is of type "GlobalIdTableEndFNDX", and that it contains zero bytes of data beyond the header.

fnd[6].header: 0x8080608C specifies that the **FileNode** is of type "DataSignatureGroupDefinitionFND", that it contains 20 bytes of data beyond the header, and that the data does not contain a reference to another part of the file.

fnd[6].fnd.DataSignatureGroup: {{7257E098-D9C8-4F9F-9EE1-E41D6AEF9D46},1} specifies a signature for each object in this group. Subsequent occurrences of the same objects in other object groups will have a different signature if their data changes in any way. In this way, a client that already has a local copy of the object can tell if it needs to fetch a new copy or not, based on the signature of the copy it has and the signature in the newer object group.

fnd[7].header: 0x8D0044A4 specifies that the **FileNode** is of type "ObjectDeclaration2RefCountFND", that it contains 13 bytes of data beyond the header, and that the data does not contain a reference to another part of the file.

fnd[7].fnd.ref: Specifies that the property set for this object is located at offset 0xF28 in the file, and is 0x20 bytes in length.

fnd[7].fnd.body.oid: By using the guidIndex to look up the GUID in the **GlobalIdTable** declared previously, the **ExtendedGUID** identity for this object is "{{9A416C35-236A-49A1-91FE-BD73076400A4}, 11}". Note that this is the ExtendedGUID that was identified as the metadata root of this object space by one of the **RootObjectReference3FND FileNodes** in the previous example.

fnd[7].fnd.body.jcid: The flags field specifies that this object is a basic property set. The index field specifies that the properties are for the section metadata.

fnd[7].fnd.body.fHasOidReferences: 0 specifies that this object does not have references to other objects in this object space.

fnd[7].fnd.body.fHasOsidReferences: 0 specifies that this object does not have references to other object spaces.

fnd[7].fnd.cRef: 1 specifies that one other item references this object. That item is the object space itself, because this is a root.

fnd[8].header: 0x8D0044A4 specifies that the **FileNode** is of type "ObjectDeclaration2RefCountFND", that it contains 13 bytes of data beyond the header, and that the data does not contain a reference to another part of the file.

fnd[8].fnd.ref: Specifies that the property set for this object is located at offset 0xEF0 in the file, and is 0x38 bytes in length.

fnd[8].fnd.body.oid: By using the guidIndex to look up the GUID in the **GlobalIdTable** declared previously, the **ExtendedGUID** identity for this object is "{{9A416C35-236A-49A1-91FE-BD73076400A4}, 10}". Note that this is the ExtendedGUID that was identified as the default content root of this object space by one of the **RootObjectReference3FND** FileNodes in the previous example.

fnd[8].fnd.body.jcid: The flags field specifies that this object is a property set as well as a graph node. The index field specifies that it is a section node.

fnd[8].fnd.body.fHasOidReferences: 1 specifies that the properties of this object contain references to other objects in this object space. Specifically, the **ElementChildNodes** property refers to the **PageSeries** object.

fnd[8].fnd.body.fHasOsidReferences: Zero specifies that this object does not have references to other object spaces.

fnd[8].fnd.cRef: 1 specifies that 1 other item references this object. That item is the object space itself, because this is a root.

fnd[9].header: 0x8D0044A4 specifies that the **FileNode** is of type "ObjectDeclaration2RefCountFND", that it contains 13 bytes of data beyond the header, and that the data does not contain a reference to another part of the file.

fnd[9].fnd.ref: Specifies that the property set for this object is located at offset 0xE88 in the file, and is 0x68 bytes in length.

fnd[9].fnd.body.oid: By using the guidIndex to look up the GUID in the **GlobalIdTable** declared previously, the **ExtendedGUID** identity for this object is "{{63230A2C-F51E-01AC-217F-603847815F77}, 1}".

fnd[9].fnd.body.jcid: The flags field specifies that this object is a basic property set. The index field specifies that it is a page metadata.

fnd[9].fnd.body.fHasOidReferences: Zero specifies that this object does not have references to other objects in this object space.

fnd[9].fnd.body.fHasOsidReferences: Zero specifies that this object does not have references to other object spaces.

fnd[9].fnd.cRef: 1 specifies that 1 other item references this object. That item is the page series node.

fnd[10].header: 0x8D0044A4 specifies that the **FileNode** is of type "ObjectDeclaration2RefCountFND", that it contains 13 bytes of data beyond the header, and that the data does not contain a reference to another part of the file.

fnd[10].fnd.ref: Specifies that the property set for this object is located at offset 0xE40 in the file, and is 0x48 bytes in length.

fnd[10].fnd.body.oid: By using the guidIndex to look up the GUID in the **GlobalIdTable** declared previously, the **ExtendedGUID** identity for this object is "{{9A416C35-236A-49A1-91FE-BD73076400A4}, 12}".

fnd[10].fnd.body.jcid: The flags field specifies that this object is a property set as well as a graph node. The index field specifies that it is a page series node.

fnd[10].fnd.body.fHasOidReferences: 1 specifies that the properties of this object contain references to other objects in this object space. Specifically, the **MetaDataObjectsAboveGraphSpace** property refers to the **PageMetaData** object.

fnd[10].fnd.body.fHasOsidReferences: 1 specifies that this object does have references to other object spaces. Specifically, the **ChildGraphSpaceElementNodes** property refers to the object space of the page.

fnd[10].fnd.cRef: 1 specifies that 1 other item references this object. That item is the section node.

fnd[11].header: 0x808010B8 specifies that the **FileNode** is of type "ObjectGroupEndFND", and that it contains zero bytes of data beyond the header.

3.5 Reading a File Encoded Using the File Synchronization via SOAP Over HTTP Protocol

The following is an example of reading a revision store file encoded using the File Synchronization via SOAP Over HTTP Protocol, specified in [MS-FSSHTTP].

1. Read the **Packaging** Structure (section 2.8.1) to determine if it is a .one section file or a .onetoc2 table of contents file.
2. Read the **data element package** ([MS-FSSHTTPB] section 2.2.1.12), inside the **Packaging** Structure.
3. Read the **Storage Index Extended GUID**.
4. Read the data element of type **Storage Index** ([MS-FSSHTTPB] section 2.2.1.12.2) from **data element packages**. The types of data elements are listed in [MS-FSSHTTPB] section 2.2.1.12.1.
5. Read the data element of type **Storage Manifest** ([MS-FSSHTTPB] section 2.2.1.12.3) from **data element packages.**
6. Read the **Storage Manifest** (section 2.7.1), to locate and read the Data Root element.
7. Read the **Cell ID** from the Data Root.
8. Read the **Storage Index** data element to find the **Storage Index Cell Mapping** that matches the **Cell ID** with the **Cell ID** read in Step 7. Read the corresponding **Cell Mapping Extended GUID**.
9. Find the **Cell Manifest** data element ([MS-FSSHTTPB] section 2.2.1.12.4) from **Cell Mapping Extended GUID** and read the **Cell Manifest Current Revision** and **Cell Manifest Current Revision Extended GUID**.
10. Read the **Storage Index** data element to find the **Storage Index Revision Mapping** that matches the **Revision Mapping Extended GUID** with the **Cell Manifest Current Revision Extended GUID** read in step 9. Read the corresponding **Revision Mapping Extended GUID**.
11. Find the **Revision Manifest** data element ([MS-FSSHTTPB] section 2.2.1.12.5) from **Revision Mapping Extended GUID** and read the **Revision Manifest Object Group References** (0 or more) and corresponding **Object Group Extended GUID**.
12. Loop through all **Object Group Extended GUIDs** and find and read **Object Group** data element ([MS-FSSHTTPB] section 2.2.1.12.6) where **Data Element Extended GUID** matches **Object Group Extended GUID**.
13. Read **Object Data** (section 2.7.6) or **Object Data BLOB Reference** in each **Object Group** data element.

4 Security

None.

4.1 Security Considerations for Implementers

4.2 Index of Security Fields

5 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include updates to those products.

- Microsoft OneNote 2010
- Microsoft OneNote 2013
- Microsoft SharePoint Foundation 2010
- Microsoft SharePoint Foundation 2013
- Microsoft SharePoint Server 2010
- Microsoft SharePoint Server 2013
- Windows 8.1 Update
- Microsoft SharePoint Server 2016
- Microsoft OneNote 2016
- Microsoft SharePoint Server 2019
- Windows 10 operating system
- Microsoft OneNote 2021
- Microsoft SharePoint Server Subscription Edition
- Microsoft OneNote LTSC 2024

Exceptions, if any, are noted in this section. If an update version, service pack or Knowledge Base (KB) number appears with a product name, the behavior changed in that update. The new behavior also applies to subsequent updates unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms "SHOULD" or "SHOULD NOT" implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term "MAY" implies that the product does not follow the prescription.

<1> Section 1.5: This persistence format provides interoperability with applications that create or read documents conforming to this structure, including Microsoft OneNote 2010.

<2> Section 2.1.12: OneNote 2010 also specifies Revision Role 0x00000004 in some revision manifests. Until the revision is also associated with Revision Role 0x00000001, it is not the active content of the object space.

<3> Section 2.3.1: Sometimes OneNote 2010 writes an incorrect sum of the total size of the free space in the Free Chunk List.

<4> Section 2.3.1: OneNote 2010 writes its own application build number.

<5> Section 2.3.1: OneNote 2010 writes its own application build number.

<6> Section 2.3.1: OneNote 2010 writes its own application build number.

<7> Section 2.3.1: OneNote 2010 writes its own application build number.

<8> Section 2.3.3.2: For .onetoc2 file, the running CRC algorithm will include all bytes except any that is flushed with the end of the TransactionLogFragment.

<9> Section 2.7.2: Microsoft OneNote Online generates an Extended GUID not following this algorithm.

<10> Section 2.8.1: This field is ignored by OneNote 2013.

<11> Section 3: In cases, where this document specifies that a field can only have one value, or that its value can be ignored, the examples might show the actual value written by OneNote 2010, but will not further explain them.

6 Change Tracking

This section identifies changes that were made to this document since the last release. Changes are classified as Major, Minor, or None.

The revision class **Major** means that the technical content in the document was significantly revised. Major changes affect protocol interoperability or implementation. Examples of major changes are:

- A document revision that incorporates changes to interoperability requirements.
- A document revision that captures changes to protocol functionality.

The revision class **Minor** means that the meaning of the technical content was clarified. Minor changes do not affect protocol interoperability or implementation. Examples of minor changes are updates to clarify ambiguity at the sentence, paragraph, or table level.

The revision class **None** means that no new technical changes were introduced. Minor editorial and formatting changes may have been made, but the relevant technical content is identical to the last released version.

The changes made to this document are listed in the following table. For more information, please contact dochelp@microsoft.com.

Section	Description	Revision class
2.3.3.2 TransactionEntry	Updated "MUST" to "SHOULD".	Minor

7 Index