[MS-FSSHTTPB]:
Binary Requests for File Synchronization via SOAP Protocol

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
07/13/2009	0.1	Major	Initial Availability
08/28/2009	0.2	Editorial	Revised and edited the technical content
11/06/2009	0.3	Editorial	Revised and edited the technical content
02/19/2010	1.0	Major	Updated and revised the technical content
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.05	Minor	Clarified the meaning of the technical content.
09/27/2010	1.06	Editorial	Changed language and formatting in the technical content.
11/15/2010	1.06	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.07	Editorial	Changed language and formatting in the technical content.
03/18/2011	1.07	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	1.07	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	2.0	Major	Significantly changed the technical content.
04/11/2012	2.0	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	2.1	Minor	Clarified the meaning of the technical content.
09/12/2012	2.1	No change	No changes to the meaning, language, or formatting of the technical content.
10/08/2012	3.0	Major	Significantly changed the technical content.
02/11/2013	4.0	Major	Significantly changed the technical content.
07/30/2013	4.1	Minor	Clarified the meaning of the technical content.

Table of Contents

1 Introduction

The Binary Requests for File Synchronization via SOAP Protocol enables protocol clients to synchronize the state of a structured file hosted by a protocol server. A typical scenario for using this protocol is to allow multiple users to edit separate parts of the file at the same time.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are defined in [MS-GLOS]:

GUID
little-endian
UTF-16

The following terms are defined in [MS-OFCGLOS]:

binary large object (BLOB)

The following terms are specific to this document:

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

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

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. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

[MS-ERREF] Microsoft Corporation, "Windows Error Codes".

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

[MS-PCCRC] Microsoft Corporation, "Peer Content Caching and Retrieval: Content Identification".

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

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary".

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary".

1.3 Overview

This protocol enables a protocol client to synchronize the state of a structured file hosted by a protocol server. It allows the protocol client to pass a set of requests to the protocol server, and to receive from the protocol server a set of responses that can be used to synchronize the contents of the file.

A typical scenario for using this protocol is to allow the file to be edited by one or more users at the same time. This protocol allows incremental updates to parts of the file structure to be made without the need to resend unchanged parts of the file structure.

1.4 Relationship to Other Protocols

This protocol is embedded within the File Synchronization via SOAP over HTTP Protocol, as described in [MS-FSSHTTP].

1.5 Prerequisites/Preconditions

None.

1.6 Applicability Statement

This protocol is intended for use where multi-user editing or incremental updates are desired features of client server file synchronization. This protocol is designed for use with file formats that can be represented by a structure of objects that have well-defined interdependencies.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

This protocol uses and the following vendor-extensible fields:

- The **GUID** value of the storage manifest, as described in section 2.2.1.12.3, is used by a protocol client to uniquely identify the schema of the file data elements, and the usage of the user data of objects. It is the responsibility of the protocol client to specify the mapping of the client's file format to the elements of the protocol, and to associate a GUID with that mapping.
- The user data contained by an object, as described in section 2.2.1.12.6 is used by protocol clients and servers in a manner specific to the schema of the storage manifest GUID, but opaque to this protocol.
- The custom filter, as described in section 2.2.2.1.3.1.5, can be used by a protocol client and a protocol server to uniquely identify a customized filtering criterion to apply to **Query Changes sub-requests** (section 2.2.2.1.3).

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol uses File Synchronization via SOAP over HTTP protocol as specified in [MS-FSSHTTP].

2.2 Common Data Types

Unless noted otherwise, the following statements apply to this specification:

- Fields that consist of more than a single byte are specified in **little-endian** byte order.
- Fields are not aligned because data are of variable length.

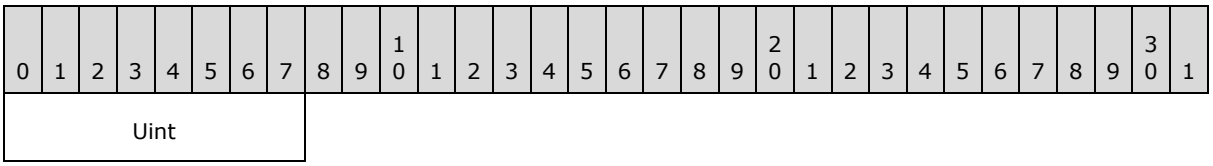
2.2.1 Basic Types

2.2.1.1 Compact Unsigned 64-bit Integer

A variable-width encoding of unsigned integers less than 18446744073709551616. The following formats are used for non-overlapping ranges of unsigned integers.

2.2.1.1.1 Compact Uint Zero

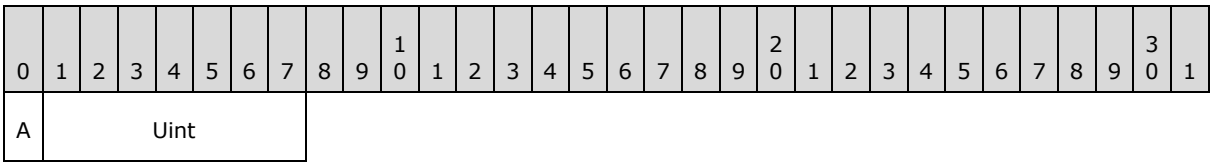
A 1-byte encoding of the value zero.



Uint (8 bits): An unsigned integer that specifies the value, and MUST be zero.

2.2.1.1.2 Compact Uint 7 bit values

A 1-byte encoding of values in the range 0x01 through 0x7F.

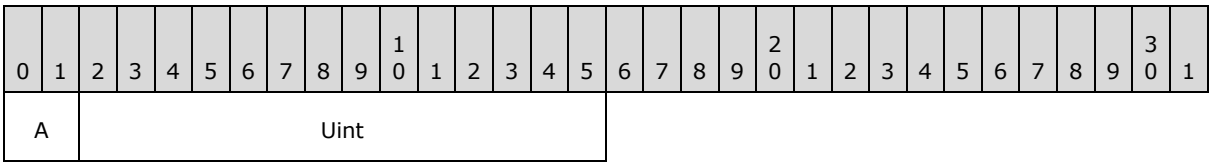


A – Type (1 bit): A flag that specifies this format from all other formats of a **compact unsigned 64-bit integer** (section 2.2.1.1), and MUST be one.

Uint (7 bits): An unsigned integer that specifies the value.

2.2.1.1.3 Compact Uint 14 bit values

A 2-byte encoding of values in the range 0x0080 through 0x3FFF.

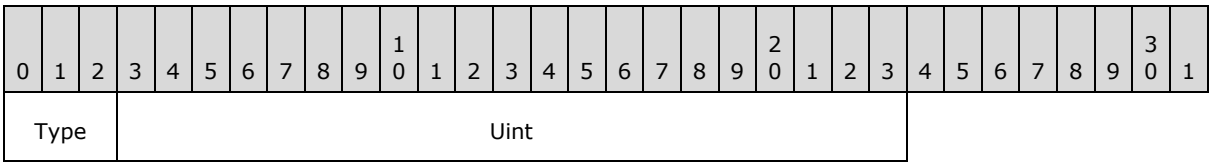


A – Type (2 bits): A flag that specifies this format from all other formats of a **compact unsigned 64-bit integer** (section 2.2.1.1), and MUST be two.

Uint (14 bits): An unsigned integer that specifies the value.

2.2.1.1.4 Compact Uint 21 bit values

A 3-byte encoding of values in the range 0x004000 through 0x1FFFFF.

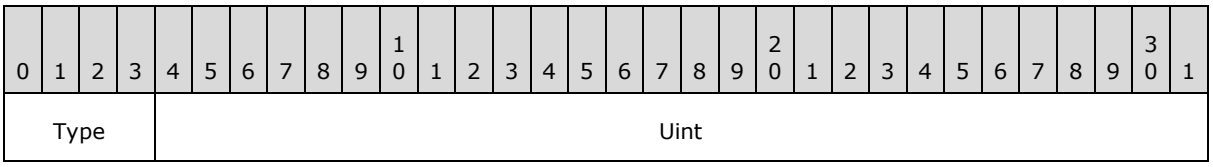


Type (3 bits): A flag that specifies this format from all other formats of a **compact unsigned 64-bit integer** (section 2.2.1.1), and MUST be four.

Uint (21 bits): An unsigned integer that specifies the value.

2.2.1.1.5 Compact Uint 28 bit values

A 4-byte encoding of values in the range 0x0200000 through 0xFFFFFFF.

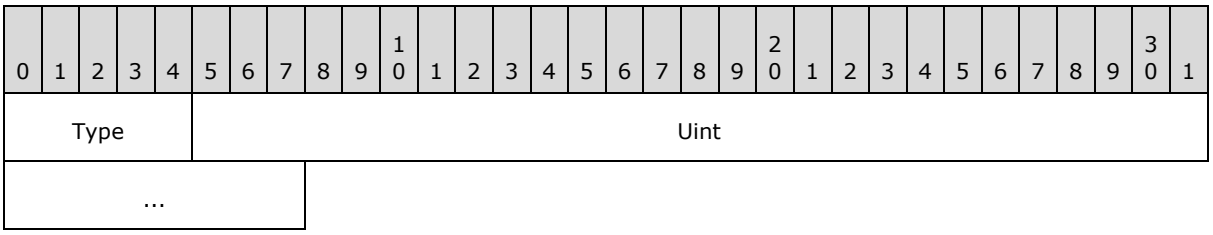


Type (4 bits): A flag that specifies this format from all other formats of a **compact unsigned 64-bit integer** (section 2.2.1.1), and MUST be eight.

Uint (28 bits): An unsigned integer that specifies the value.

2.2.1.1.6 Compact Uint 35 bit values

A 5-byte encoding of values in the range 0x010000000 through 0x7FFFFFFFF.

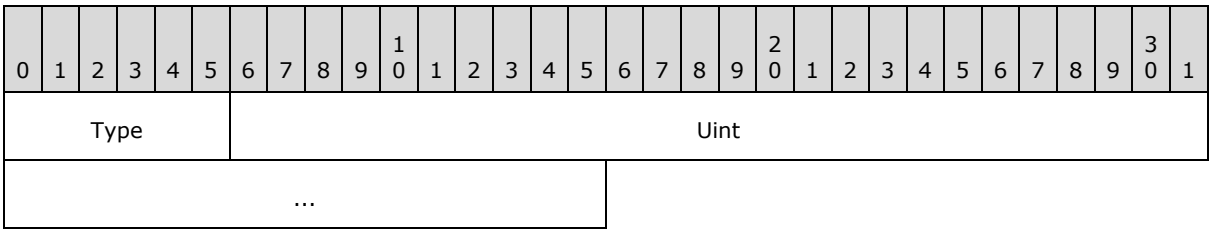


Type (5 bits): A flag that specifies this format from all other formats of a **compact unsigned 64-bit integer** (section 2.2.1.1), and MUST be 16.

Uint (35 bits): An unsigned integer that specifies the value.

2.2.1.1.7 Compact Uint 42 bit values

A 6-byte encoding of values in the range 0x00800000000 through 0x3FFFFFFFFF.

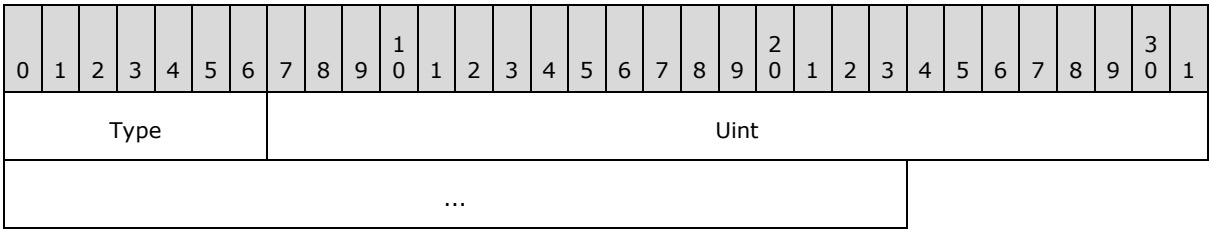


Type (6 bits): A flag that specifies this format from all other formats of a **compact unsigned 64-bit integer** (section 2.2.1.1), and MUST be 32.

Uint (42 bits): An unsigned integer that specifies the value.

2.2.1.1.8 Compact Uint 49 bit values

A 7-byte encoding of values in the range 0x0040000000000 through 0x1FFFFFFFFFFF.

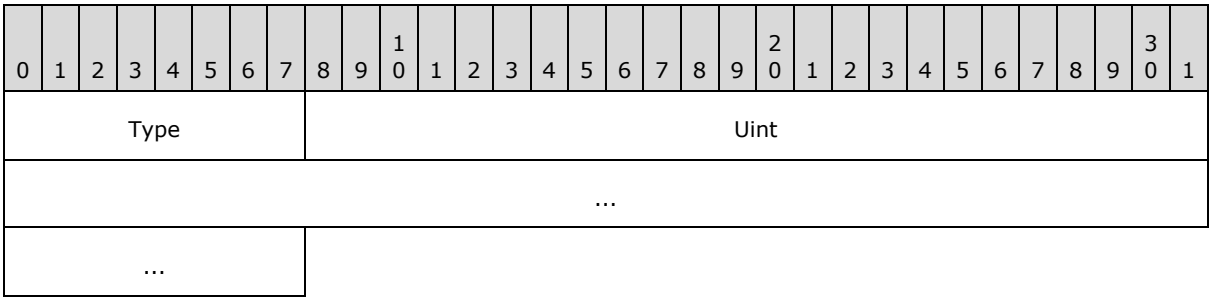


Type (7 bits): A flag that specifies this format from all other formats of a **compact unsigned 64-bit integer** (section 2.2.1.1), and MUST be 64.

Uint (49 bits): An unsigned integer that specifies the value.

2.2.1.1.9 Compact Uint 64 bit values

A 9-byte encoding of values in the range 0x0002000000000000 through 0xFFFFFFFFFFFFFFFF.

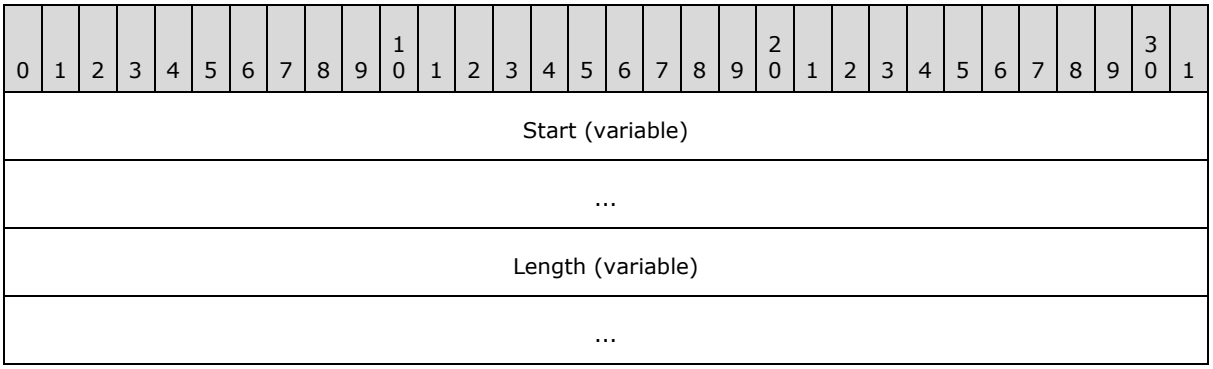


Type (8 bits): A flag that specifies this format from all other formats of a **compact unsigned 64-bit integer** (section 2.2.1.1), and MUST be 128.

Uint (64 bits): An unsigned integer that specifies the value.

2.2.1.2 File Chunk Reference

The starting offset and length of a contiguous portion of a file.

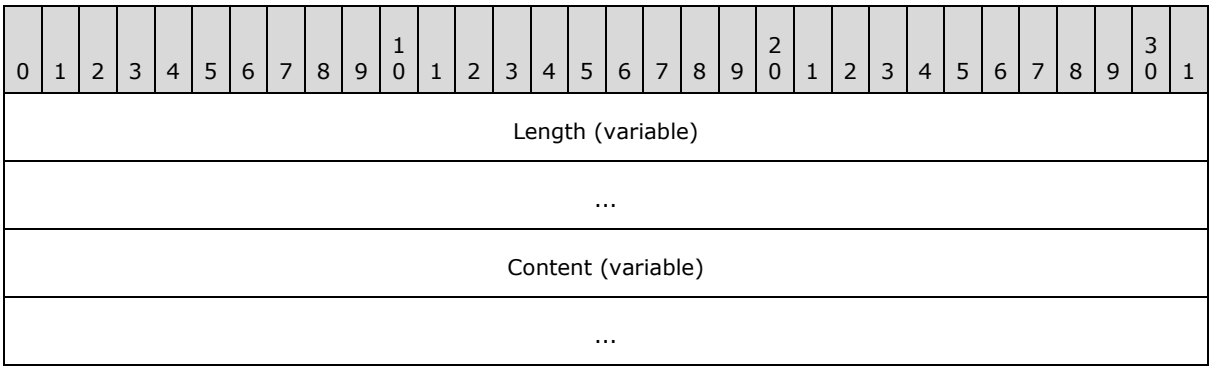


Start (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the byte-offset within the file of the beginning of the file chunk.

Length (variable): A **compact unsigned 64-bit integer** that specifies the count of bytes included in the file chunk.

2.2.1.3 Binary Item

The length and bytes of an arbitrary binary stream of data.

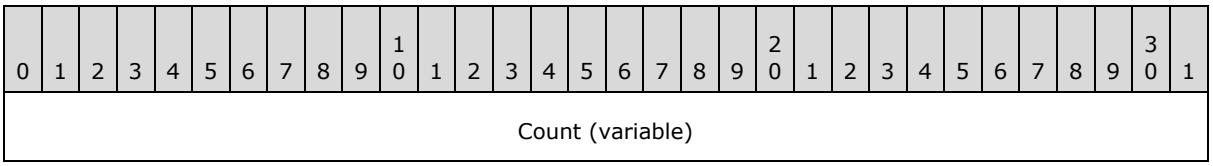


Length (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the count of bytes of **Content** of the item.

Content (variable): A byte stream that specifies the data for the item.

2.2.1.4 String Item

The count and content of an arbitrary wide character string.



...
Content (variable)
...

Count (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the count of characters in the string.

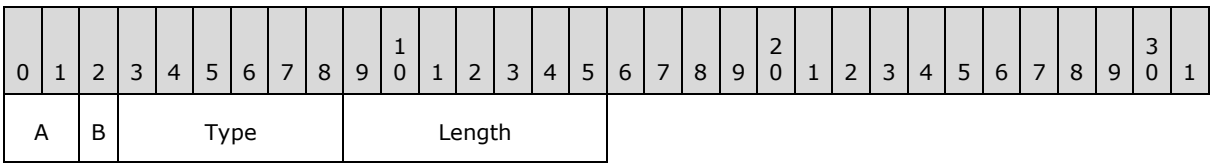
Content (variable): An array of **UTF-16** characters that specify the string. It MUST NOT be null-terminated.

2.2.1.5 Stream Object Header

The **stream object header** is either 8-bit, 16-bit or 32-bit indicating whether it is a single or compound stream object, a start or end type for compound stream objects, and the length of data after the header (if any). The length value does not include the size of the **stream object headers**.

2.2.1.5.1 16-bit Stream Object Header Start

A 16-bit header for either a single or a start of a compound object has the following format:



A - Header Type (2-bit): A flag that specifies a 16-bit stream object start. This MUST be set to 0x0.

B - Compound (1-bit): If set, a bit that specifies a compound parse type is needed, and MUST end with either an **8-bit stream object header end** (section 2.2.1.5.3) or a **16-bit stream object header end** (section 2.2.1.5.4). If the bit is not set, it specifies a single object.

Type (6-bits): A 6-bit unsigned integer that specifies the stream object type (see the following table for possible values).

Length (7-bits): A 7-bit unsigned integer that specifies the length in bytes for additional data (if any) before the next **Stream Object Header Start** or **Stream Object Header End**. If the length is more than 127 bytes, a **32-bit stream object header start** (section 2.2.1.5.2) MUST be used.

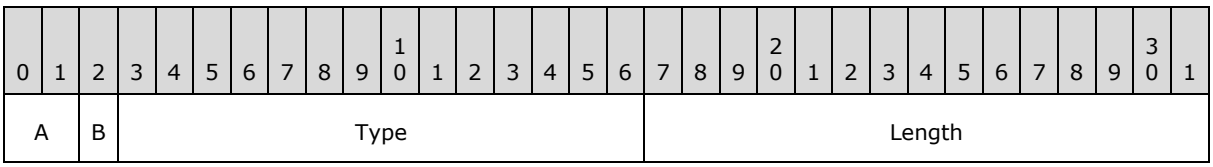
The following table lists the possible stream object types, and the corresponding **Compound** value:

Stream object type	Value	Compound
Data element.	0x01	1
Object data **BLOB**.	0x02	0
Object group object excluded data.	0x03	0

Stream object type	Value	Compound
Waterline knowledge entry (section 2.2.1.13.4.1).	0x04	0
Object group object BLOB data declaration.	0x05	0
Storage manifest root declare.	0x07	0
Revision manifest root declare.	0x0A	0
Cell manifest current revision.	0x0B	0
Storage manifest schema GUID.	0x0C	0
Storage index revision mapping.	0x0D	0
Storage index cell mapping.	0x0E	0
Cell knowledge range (section 2.2.1.13.2.1).	0x0F	0
Knowledge (section 2.2.1.13).	0x10	1
Storage index manifest mapping.	0x11	0
Cell Knowledge (section 2.2.1.13.2).	0x14	1
Data element package.	0x15	1
Object group object data.	0x16	0
Cell knowledge entry (section 2.2.1.13.2.2).	0x17	0
Object group object declare.	0x18	0
Revision manifest object group references.	0x19	0
Revision manifest.	0x1A	0
Object group object data BLOB reference.	0x1C	0
Object group declarations.	0x1D	1
Object group data.	0x1E	1
Waterline knowledge (section 2.2.1.13.4).	0x29	1
Content tag knowledge (section 2.2.1.13.5).	0x2D	1
Content tag knowledge entry.	0x2E	0

2.2.1.5.2 32-bit Stream Object Header Start

A 32-bit header for either a single or a start of a compound object has the following format.



Large Length (variable)
…

A – Header Type (2-bit): A flag that specifies a 32-bit stream object start. This MUST be set to 0x2.

B - Compound (1-bit): If set, a bit that specifies a compound parse type is needed, and MUST end with either an **8-bit stream object header end** (section 2.2.1.5.3) or a **16-bit stream object header end** (section 2.2.1.5.4). If the bit is not set, it specifies a single object.

Type (14-bits): A 14-bit unsigned integer that specifies the stream object type (see the following table for possible values).

Length (15-bits): A 15-bit unsigned integer that specifies the length in bytes for additional data (if any) before next **Stream Object Header Start** or **Stream Object Header End**. If the length is more than 32766, this field MUST specify 32767, and a **Large Length** field MUST be specified.

Large Length (variable): An optional **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the length in bytes for additional data (if any). This field MUST be specified if the **Length** field contains 32767, and MUST NOT be specified if the **Length** field contains any other value than 32767.

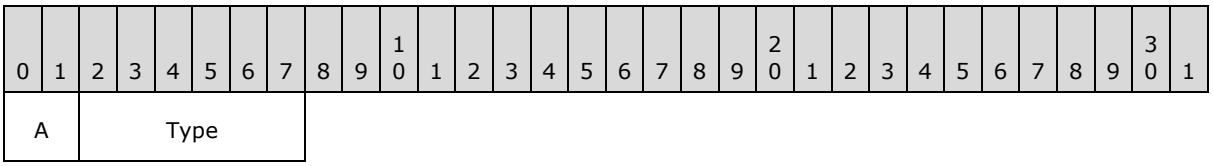
The following table lists the possible stream object types, and the corresponding Compound value:

Stream object type	Value	Compound
Request.	0x040	1
Sub-response.	0x041	1
Sub-request.	0x042	1
Read access response.	0x043	1
Specialized knowledge.	0x044	1
Put changes response serial number reassign all.	0x045	0
Write access response.	0x046	1
Query changes filter.	0x047	1
Error Win32.	0x049	0
Error Protocol.	0x04B	0
Error.	0x04D	1
Error String Supplemental Info.	0x04E	0
User agent version.	0x04F	0
Query changes filter schema specific.	0x050	0
Query changes request.	0x051	0

Stream object type	Value	Compound
Error HRESULT.	0x052	0
Put changes response serial number reassign.	0x053	0
Query changes filter data element IDs.	0x054	0
User agent GUID.	0x055	0
Query changes filter data element type.	0x057	0
Query changes data constraint.	0x059	0
Put changes request.	0x05A	0
Query changes request arguments.	0x05B	0
Query changes filter cell ID.	0x05C	0
User agent.	0x05D	1
Query changes response.	0x05F	0
Query changes filter hierarchy.	0x060	0
Response.	0x062	1
Error cell.	0x066	0
Query changes filter flags.	0x068	0
Data element fragment.	0x06A	0
Fragment knowledge.	0x06B	1
Fragment knowledge entry.	0x06C	0
Object group metadata declarations.	0x79	1
Object group metadata.	0x78	0
Allocate ExtendedGUID range request (section 2.2.2.1.5).	0x080	0
Allocate ExtendedGUID range response (section 2.2.3.1.4).	0x081	0
Target Partition Id. (section 2.2.2.1.1).	0x83	1
Put Changes Lock Id. (section 2.2.2.1.4.2)	0x85	
Additional Flags. (section 2.2.2.1.4.1)	0x86	0
Put Changes Response.	0x87	0
Request hashing options.	0x88	0

2.2.1.5.3 8-bit Stream Object Header End

An 8-bit header for a compound object that indicates the end of a stream object has the following format.



A – Header Type (2-bit): A flag that specifies an 8-bit stream object end. This MUST be set to 0x1.

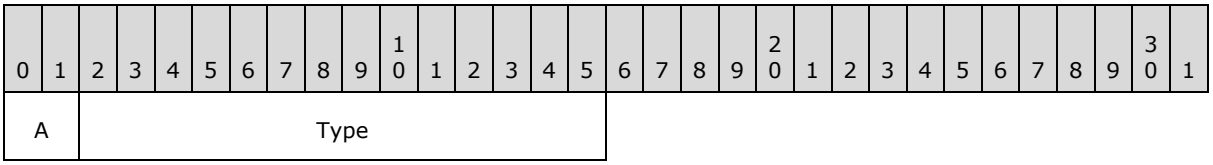
Type (6-bits): A 6-bit unsigned integer that specifies the stream object type.

The following table lists the possible stream object types.

Stream object type	Value
Data element.	0x01
Knowledge.	0x10
Cell Knowledge.	0x14
Data element package.	0x15
Object group declarations.	0x1D
Object group data.	0x1E
Waterline knowledge.	0x29
Content tag knowledge.	0x2D

2.2.1.5.4 16-bit Stream Object Header End

A 16-bit header for a compound object that indicates the end of a stream object has the following format.



A – Header Type (2 bits): A flag that specifies a 16-bit stream object end. This MUST be set to 0x3.

Type (14 bits): A 14-bit unsigned integer that specifies the stream object type.

The following table lists the possible stream object types:

Stream Object Type	Value
Request.	0x040
Sub-response.	0x041
Sub-request.	0x042

Stream Object Type	Value
Read access response.	0x043
Specialized knowledge.	0x044
Write access response.	0x046
Query changes filter.	0x047
Error.	0x04D
Query changes request.	0x051
User agent.	0x05D
Response.	0x062
Fragment knowledge.	0x06B
Object group metadata declarations.	0x079
Target Partition Id.	0x083

2.2.1.6 Request Type Enumeration

Specifies the **sub-request** type (section 2.2.2.1) to indicate the operation being requested, or the **sub-response** type (section 2.2.3.1) to indicate the response per the request.

The following table enumerates the values for each operation:

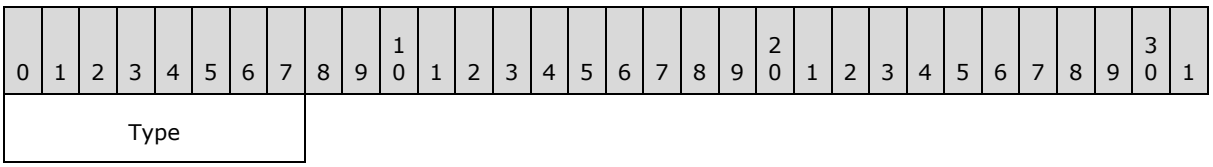
Value	Meaning
1	Query access.
2	Query changes.
5	Put changes.
11	Allocate ExtendedGuid range

2.2.1.7 Extended GUID

A variable-width encoding that specifies a combination of a GUID and **Value,** an unsigned 32-bit integer. The following formats are used to encode non-overlapping ranges of **extended GUIDs**.

2.2.1.7.1 Extended GUID Null Value

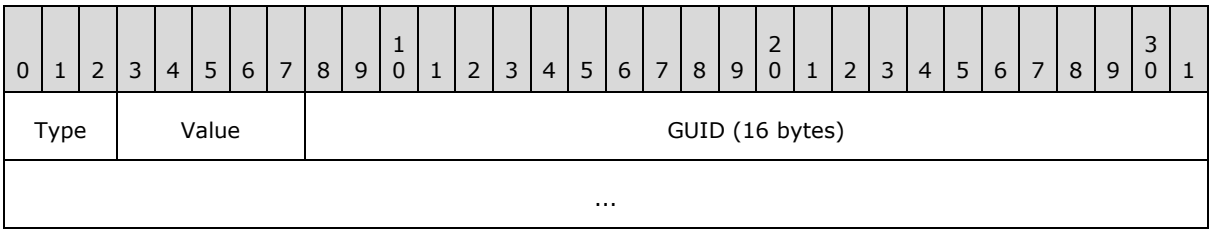
A 1-byte encoding of the **extended GUID** when the GUID part is {00000000-0000-0000-0000-000000000000} and the unsigned integer part is zero.



Type (1 byte): An unsigned integer that specifies a null GUID and MUST be zero.

2.2.1.7.2 Extended GUID 5 Bit Uint Value

A 17-byte encoding of the **extended GUID** when the integer part ranges from 0x0 through 0x1F.



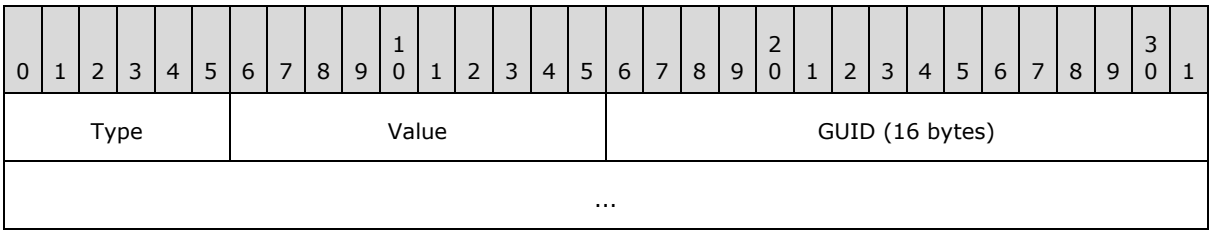
Type (3 bits): An unsigned integer that specifies the type that MUST be 4.

Value (5 bits): An unsigned integer that specifies the value.

GUID (16 bytes): A GUID that specifies the item. This MUST NOT be "{00000000-0000-0000-0000-000000000000}".

2.2.1.7.3 Extended GUID 10 Bit Uint Value

An 18-byte encoding of the **extended GUID** when the integer part ranges from 0x20 through 0x3FF.



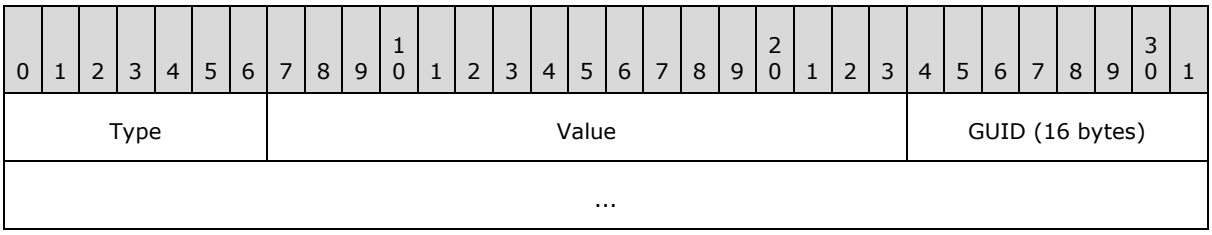
Type (6 bits): An unsigned integer that specifies the type that MUST be 32.

Value (10 bits): An unsigned integer that specifies the value.

GUID (16 bytes): A GUID that specifies the item. This MUST NOT be "{00000000-0000-0000-0000-000000000000}".

2.2.1.7.4 Extended GUID 17 Bit Uint Value

A 19-byte encoding of the **extended GUID** when the integer part ranges from 0x400 through 0x1FFFF.



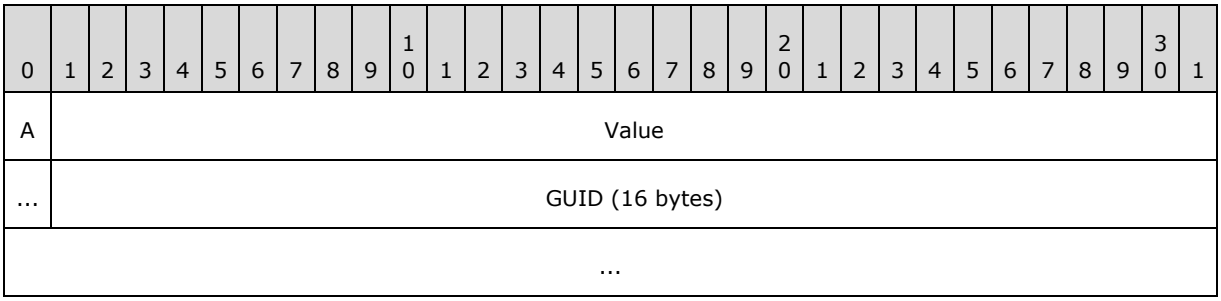
Type (7 bits): An unsigned integer that specifies the type that MUST be 64.

Value (17 bits): An unsigned integer that specifies the value.

GUID (16 bytes): A GUID that specifies the item. This MUST NOT be {00000000-0000-0000-0000-000000000000}.

2.2.1.7.5 Extended GUID 32 Bit Uint Value

A 21-byte encoding of the **extended GUID** when the integer part ranges from 0x20000 through 0xFFFFFFFF.



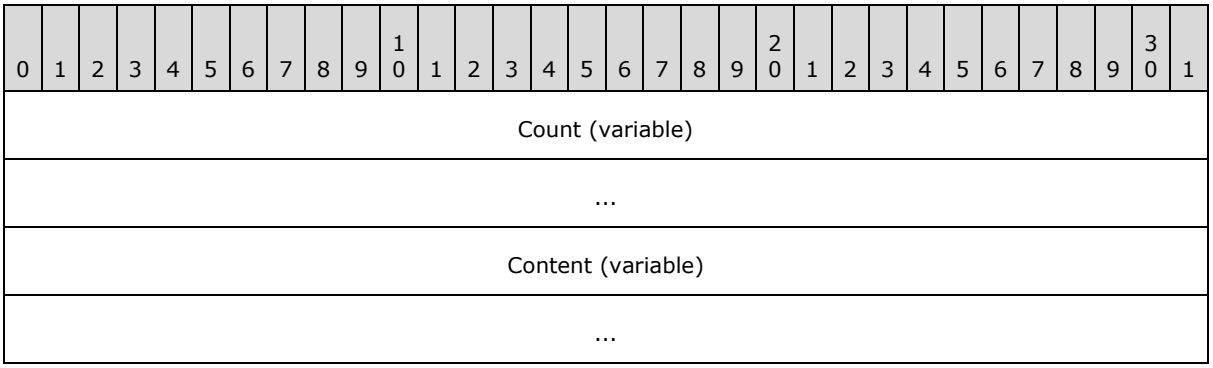
Type (1 byte): An unsigned integer that specifies the type that MUST be 128.

Value (4 bytes): An unsigned integer that specifies the value.

GUID (16 bytes): A GUID that specifies the item. This MUST NOT be {00000000-0000-0000-0000-000000000000}.

2.2.1.8 Extended GUID Array

The length and content of an array of **extended GUIDs**, as specified in section 2.2.1.7.



Count (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the count of **extended GUIDs** in the array.

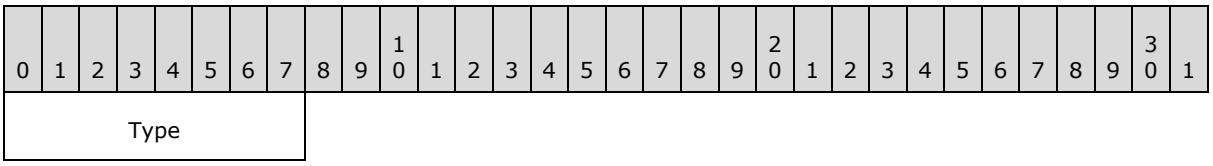
Content (variable): An **extended GUID array** that specifies an array of items.

2.2.1.9 Serial Number

A variable-width encoding that specifies a combination of a GUID and an unsigned 64-bit integer. The following formats are used to encode non-overlapping ranges of **serial numbers**.

2.2.1.9.1 Serial Number Null Value

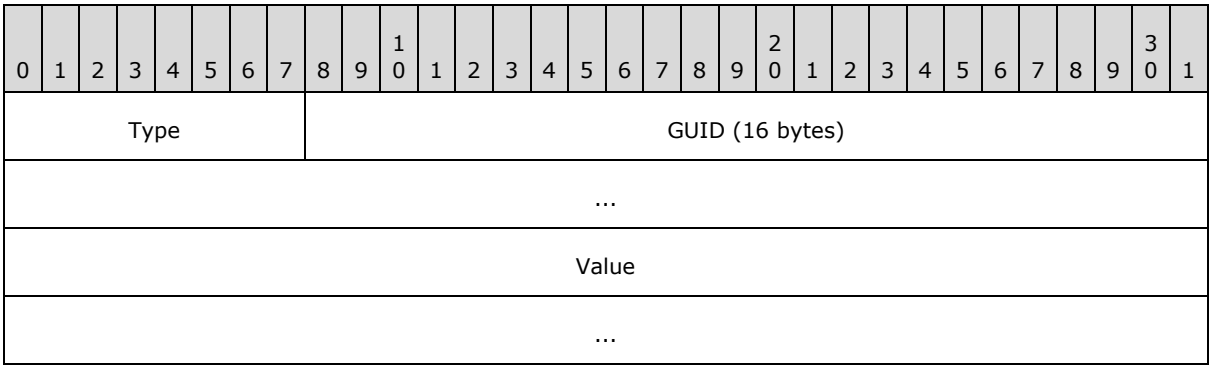
A 1-byte encoding of the **serial number** when the GUID part is {00000000-0000-0000-0000-000000000000} and the unsigned integer part is zero.



Type (8 bits): An unsigned integer that specifies a null **serial number**, and MUST be zero.

2.2.1.9.2 Serial Number 64 Bit Uint Value

A 25-byte encoding of the **serial number**.



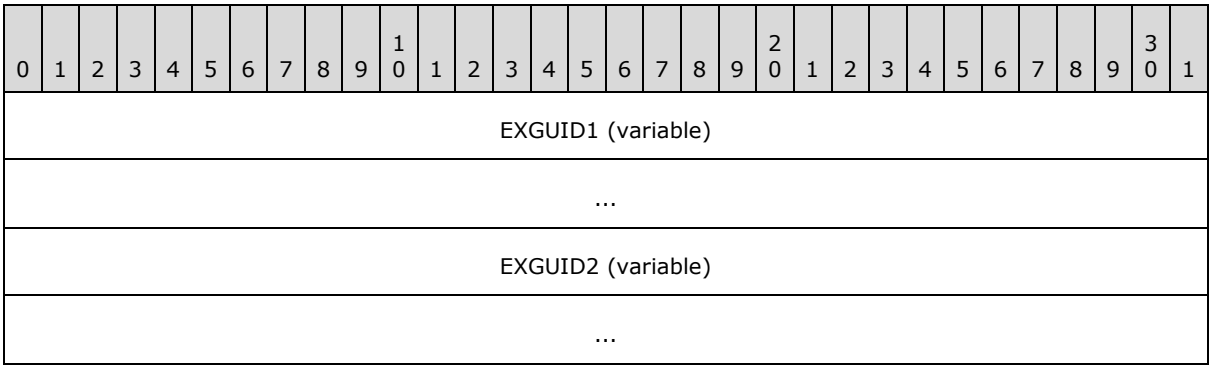
Type (1 byte): An unsigned integer that specifies the type, and MUST be 128.

GUID (16 bytes): A GUID that specifies the item and it MUST NOT be {00000000-0000-0000-0000-000000000000}.

Value (8 bytes): An unsigned integer that specifies the value of the **serial number**.

2.2.1.10 Cell ID

Specifies the cell identifier consisting of two **extended GUIDs** (section 2.2.1.7) specified in the following format.

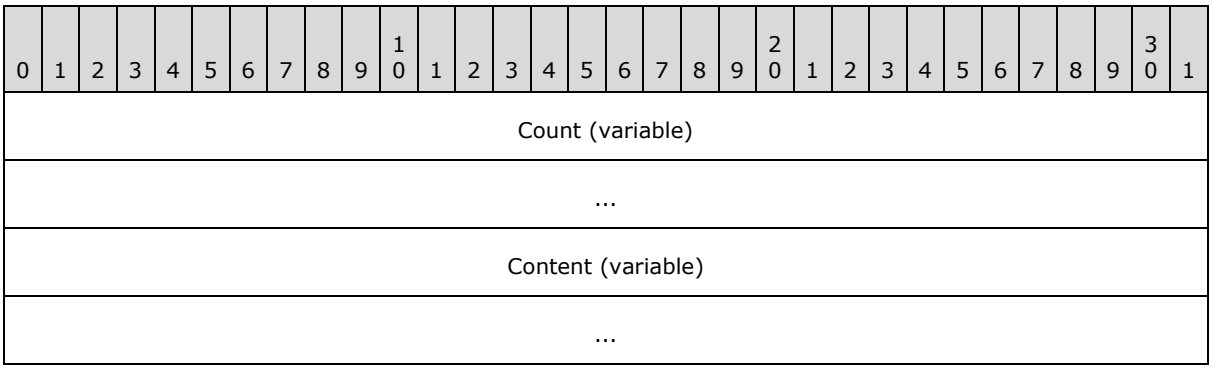


EXGUID1 (variable): An **extended GUID** that specifies the first cell identifier.

EXGUID2 (variable): An **extended GUID** that specifies the second cell identifier.

2.2.1.11 Cell ID Array

The count and content of an array of **Cell IDs**, as specified in section 2.2.1.10.

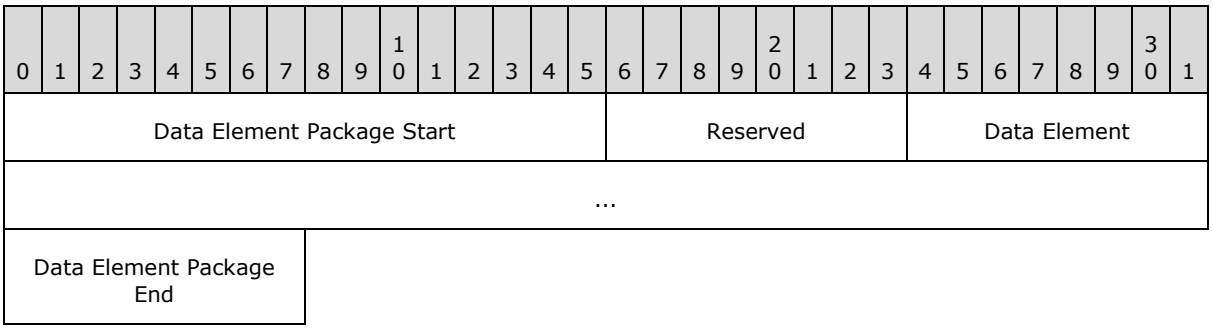


Count (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the count of **cell IDs** in the array.

Content (variable): A **cell ID array** that specifies an array of cells.

2.2.1.12 Data Element Package

A **data element package** contains the serialized file data elements made up of storage index (section 2.2.1.12.2), storage manifest (section 2.2.1.12.3), cell manifest (section 2.2.1.12.4), revision manifest (section 2.2.1.12.5), and object group (section 2.2.1.12.6) or object data (section 2.2.1.12.6.4), or both.



Data Element Package Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a **data element package** start.

Reserved (1 byte): A reserved field that MUST be set to zero, and MUST be ignored.

Data Element (variable): An optional array of data elements (section 2.2.1.12.1) that specifies the serialized file data elements. If the client doesn't have any data elements, this MUST NOT be present.

Data Element Package End (1 byte): An **8-bit stream object header** (section 2.2.1.5.3) that specifies a **data element package** end.

2.2.1.12.1 Data Element Types

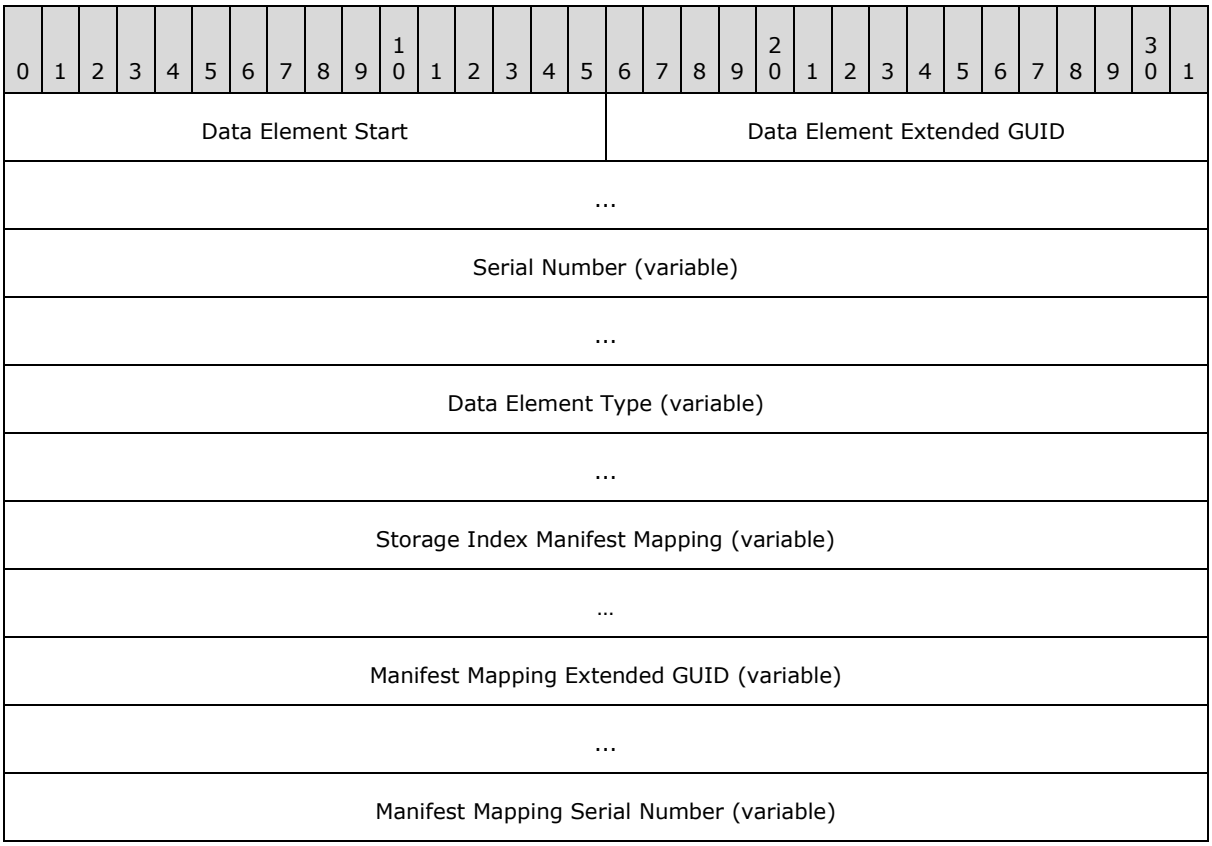
The following table lists the possible data element types:

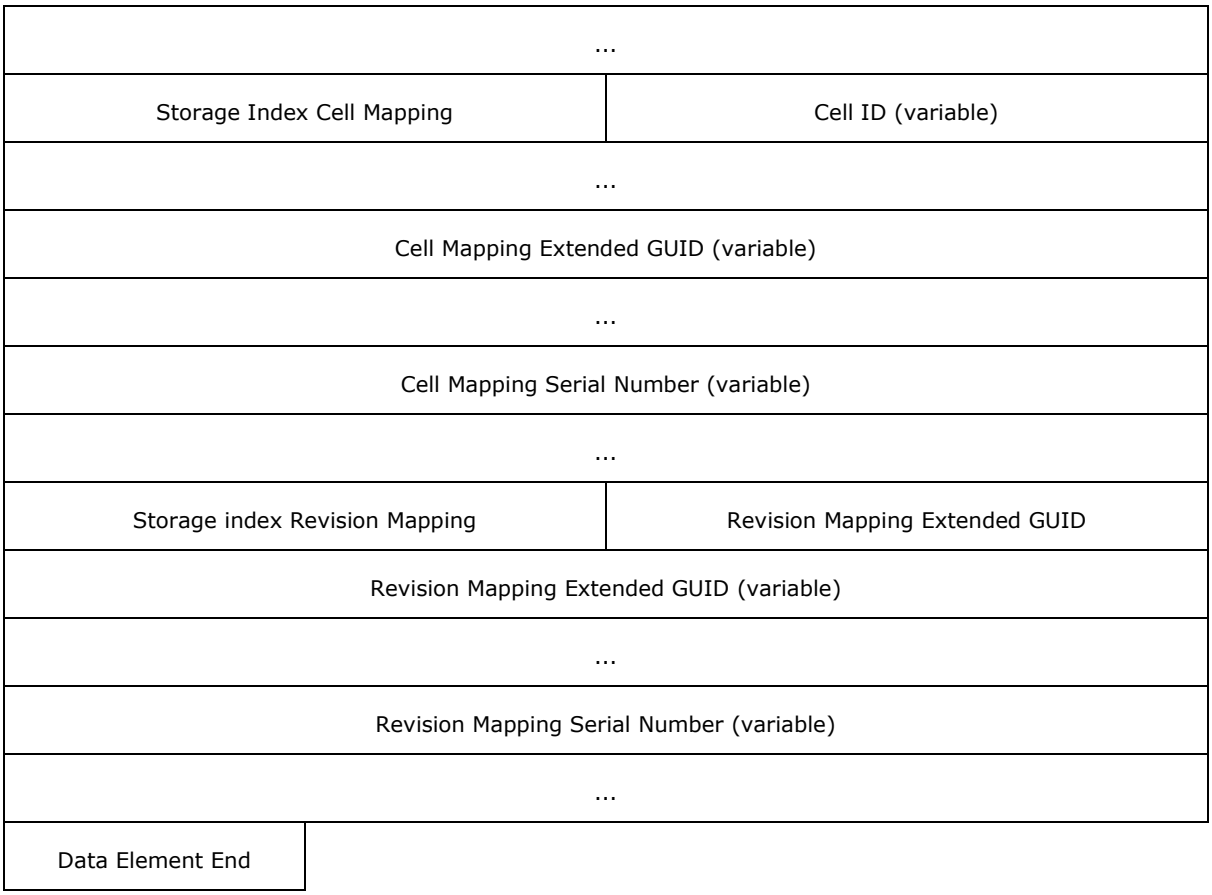
Data element type	Value
Storage Index (section 2.2.1.12.2)	0x01
Storage Manifest (section 2.2.1.12.3)	0x02
Cell Manifest (section 2.2.1.12.4)	0x03
Revision Manifest (section 2.2.1.12.5)	0x04
Object Group (section 2.2.1.12.6)	0x05
Data Element Fragment (section 2.2.1.12.7)	0x06
Object Data BLOB (section 2.2.1.12.8)	0x0A

2.2.1.12.2 Storage Index Data Element

A **Storage Index data element** has the following format.

When serializing a **Storage Index data element**, there is no sequence for **Storage Index Manifest Mapping**, **Storage Index Cell Mapping** and **Storage Index Revision Mapping**.





Data Element Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a data element start.

Data Element Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the data element.

Serial Number (variable): A **serial number** (section 2.2.1.9) that specifies the data element.

Data Element Type (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the value of the **storage index data element** type.

Storage Index Manifest Mapping (2 bytes): Zero or one **16-bit stream object header** that specifies the storage index manifest mappings (with manifest mapping **extended GUID** and **serial number**).

Manifest Mapping Extended GUID (variable): An **extended GUID** that specifies the manifest mapping.

Manifest Mapping Serial Number (variable): A **serial number** that specifies the manifest mapping.

Storage Index Cell Mapping (2 bytes): Zero or more **16-bit stream object header** that specifies the storage index cell mappings (with cell identifier, cell mapping **extended GUID**, and cell mapping **serial number**).

Cell ID (variable): A **cell ID** (section 2.2.1.10) that specifies the cell identifier.

Cell Mapping Extended GUID (variable): An **extended GUID** that specifies the cell mapping.

Cell Mapping Serial Number (variable): A **serial number** that specifies the cell mapping.

Storage Index Revision Mapping (2 bytes): Zero or more **16-bit stream object header** that specifies the storage index revision mappings (with revision and revision mapping **extended GUIDs**, and revision mapping **serial number**).

Revision Extended GUID (variable): An **extended GUID** that specifies the revision.

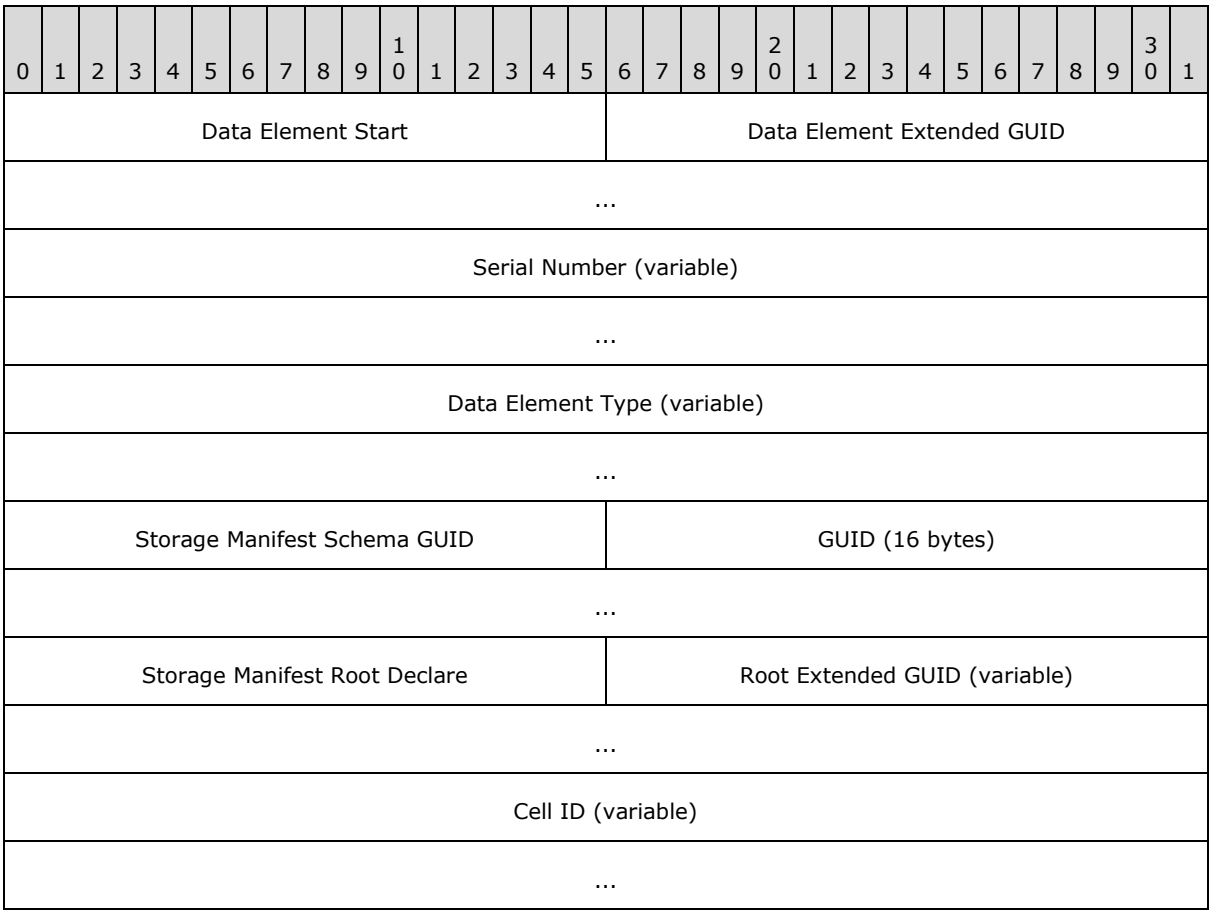
Revision Mapping Extended GUID (variable): An **extended GUID** that specifies the revision mapping.

Revision Mapping Serial Number (variable): A **serial number** that specifies the revision mapping.

Data Element End (1 byte): An **8-bit stream object header** that specifies a data element end.

2.2.1.12.3 Storage Manifest Data Element

A **Storage Manifest data element** has the following format.



Data Element End

Data Element Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a data element start.

Data Element Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the data element.

Serial Number (variable): A **serial number** (section 2.2.1.9) that specifies the data element.

Data Element Type (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the value of the **storage manifest data element** type.

Storage Manifest Schema GUID (2 bytes): A **16-bit stream object header** that specifies a storage manifest schema GUID.

GUID (16 bytes): A GUID that specifies the schema.

Storage Manifest Root Declare (2 bytes): A **16-bit stream object header** that specifies one or more storage manifest root declare.

Root Extended GUID (variable): An **extended GUID** that specifies the root storage manifest.

Cell ID (variable): A **cell ID** (section 2.2.1.10) that specifies the cell identifier.

Data Element End (1 byte): An **8-bit stream object header** (section 2.2.1.5.3) that specifies a data element end.

2.2.1.12.4 Cell Manifest Data Element

A C**ell Manifest data element** has the following format.

<table>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>20</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>30</th><th>1</th></tr>
<tr><td colspan="16">Data Element Start</td><td colspan="16">Data Element Extended GUID</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Serial Number (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Data Element Type (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="16">Cell Manifest Current Revision</td><td colspan="16">Cell Manifest Current Revision Extended GUID</td></tr>
<tr><td colspan="32">...</td></tr>
</table>

Data Element End

Data Element Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a data element start.

Data Element Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the data element.

Serial Number (variable): A **serial number** (section 2.2.1.9) that specifies the data element.

Data Element Type (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the value of the **cell manifest data element** type.

Cell Manifest Current Revision (2 bytes): A **16-bit stream object header** that specifies a cell manifest current revision.

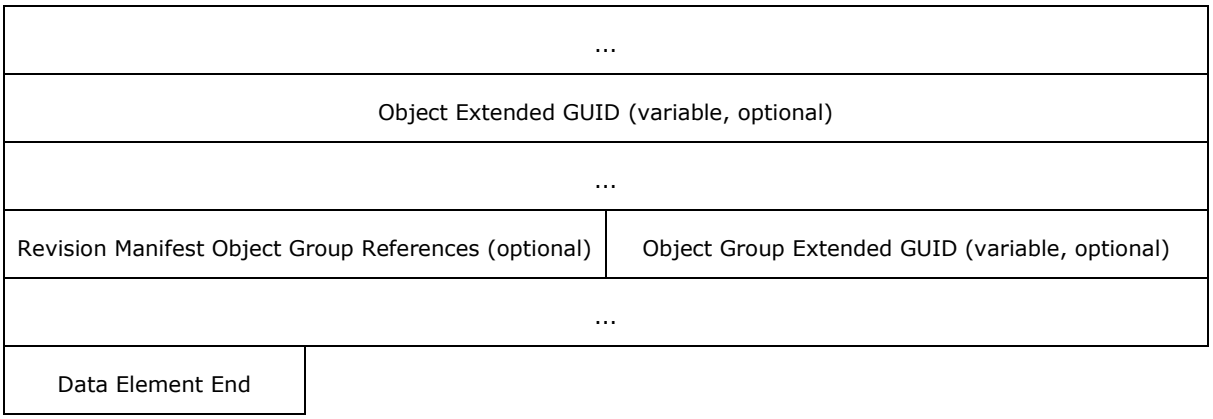
Cell Manifest Current Revision Extended GUID (variable): An **extended GUID** that specifies the revision.

Data Element End (1 byte): An **8-bit stream object header** (section 2.2.1.5.3) that specifies a data element end.

2.2.1.12.5 Revision Manifest Data Elements

A **Revision Manifest data element** has the following format.

<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>20</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>30</td><td>1</td></tr>
<tr><td colspan="16">Data Element Start</td><td colspan="16">Data Element Extended GUID (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Serial Number (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Data Element Type (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="16">Revision Manifest</td><td colspan="16">Revision ID (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Base Revision ID (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="16">Revision Manifest Root Declare (variable, optional)</td><td colspan="16">Root Extended GUID (variable, optional)</td></tr>
</table>



Data Element Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a data element start.

Data Element Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the data element.

Serial Number (variable): A **serial number** (section 2.2.1.9) that specifies the data element.

Data Element Type (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the value of the **revision manifest data element** type.

Revision Manifest (2 bytes): A **16-bit stream object header** that specifies a revision manifest.

Revision ID (variable): An **extended GUID** that specifies the revision identifier represented by this data element.

Base Revision ID (variable): An **extended GUID** that specifies the revision identifier of a base revision that could contain additional information for this revision.

Revision Manifest Root Declare (2 bytes, optional): Zero or more **16-bit stream object header** that specifies a revision manifest root declare, each followed by root and object **extended GUIDs**.

Root Extended GUID (variable): An **extended GUID** that specifies the root revision for each revision manifest root declare.

Object Extended GUID (variable): An **extended GUID** that specifies the object for each revision manifest root declare.

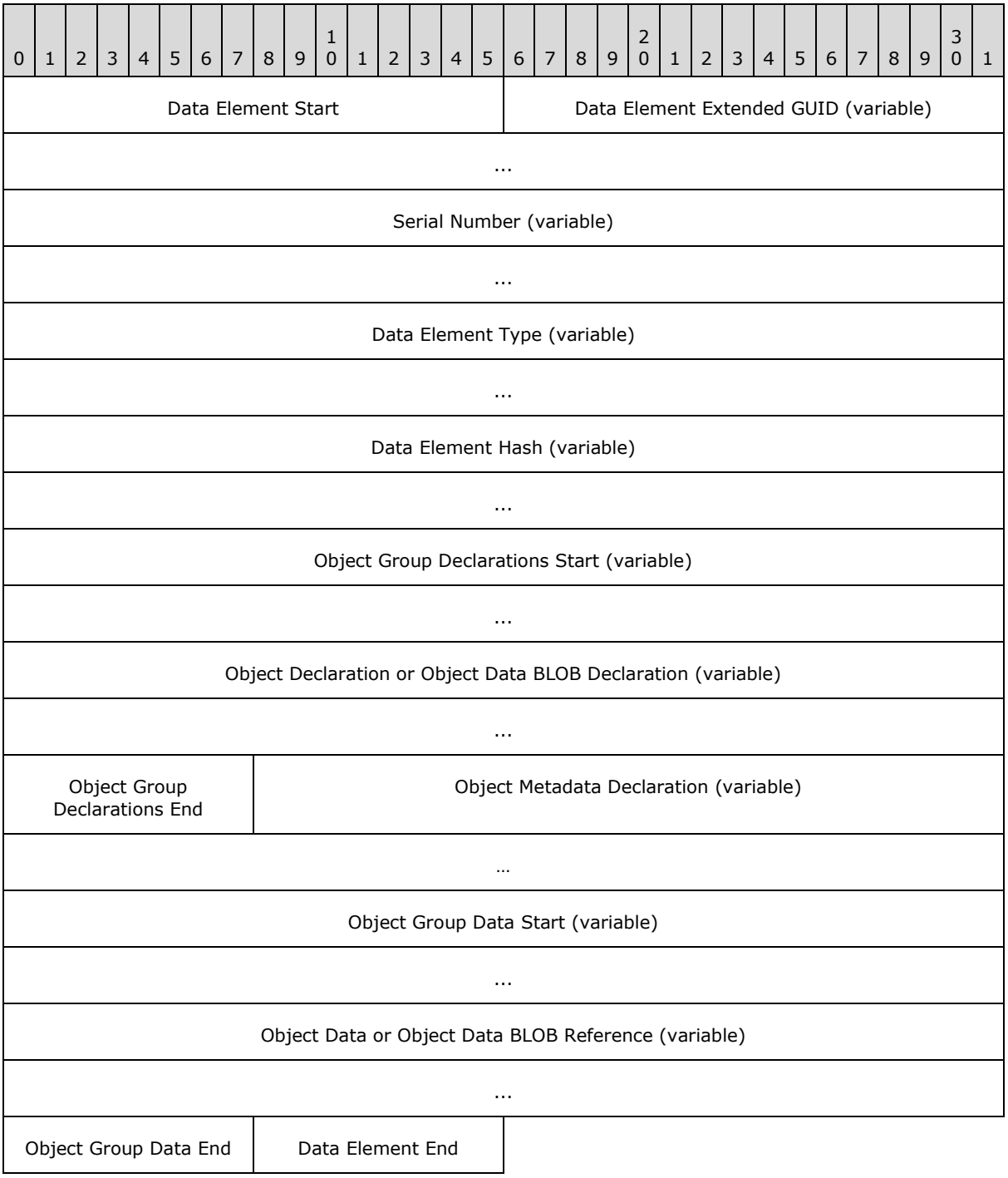
Revision Manifest Object Group References (2 bytes, optional): Zero or more **16-bit stream object header** that specifies a revision manifest object group references, each followed by object group **extended GUIDs**.

Object Group Extended GUID (variable): An **extended GUID** that specifies the object group for each **Revision Manifest Object Group References**.

Data Element End (1 byte): An **8-bit stream object header** (section 2.2.1.5.3) that specifies a data element end.

2.2.1.12.6 Object Group Data Elements

An **Object Group data element** has the following format.



Data Element Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a data element start.

Data Element Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the data element.

Serial Number (variable): A **serial number** (section 2.2.1.9) that specifies the data element.

Data Element Type (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the value of the **object group data element** type.

Data Element Hash (variable): An optional **data element hash** that specifies the value to be used when fetching/injecting **object data** from/to a protocol client cache.

Object Group Declarations Start (variable): A **16-bit** (section 2.2.1.5.1) or **32-bit stream object header** (section 2.2.1.5.2) that specifies an object group declaration start.

Object Declaration / Object Data BLOB Declaration (variable): An optional array of **object declarations** (section 2.2.1.12.6.1) or **object data BLOB declarations** (section 2.2.1.12.6.2) that specifies the object.

Object Group Declarations End (1 byte): An **8-bit stream object header** (section 2.2.1.5.3) that specifies an object group declaration end.

Object Metadata Declaration (variable): If **object metadata** (section 2.2.1.12.6.3.1) exists, this field MUST specify an **object metadata declaration** (section 2.2.1.12.6.3). If no **object metadata** exists, this field must be omitted.

Object Group Data Start (variable): A **16-bit** or **32-bit stream object header** that specifies an **object group** data start.

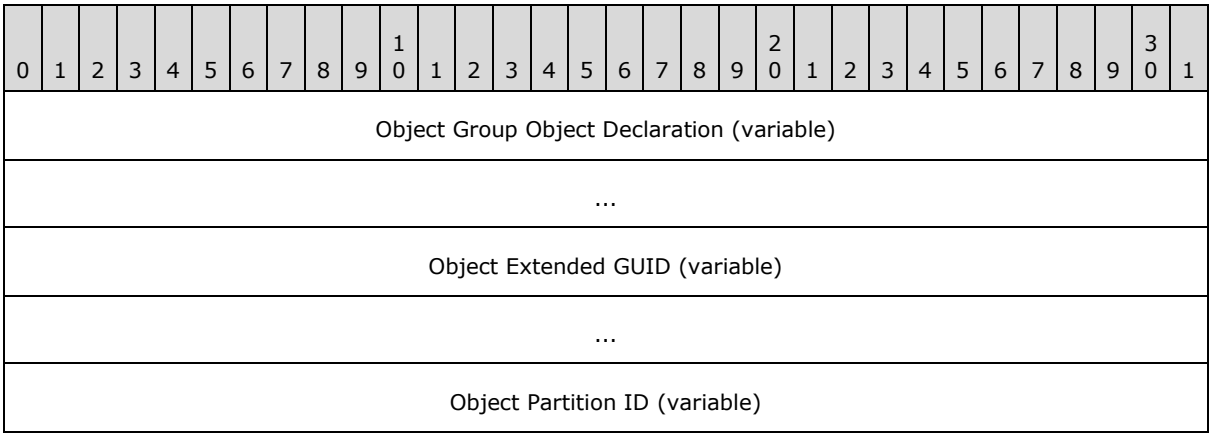
Object Data / Object Data BLOB Reference (variable): An optional array of **object data** (section 2.2.1.12.6.4) or **object data BLOB references** (section 2.2.1.12.6.5) that specifies the **object data** or its references.

Object Group Data End (1 byte): An **8-bit stream object header** that specifies an **object group** data end.

Data Element End (1 byte): An **8-bit stream object header** that specifies a data element end.

2.2.1.12.6.1 Object Declaration

An **Object Declaration** has the following structure.



...
Object Data Size (variable)
...
Object References Count (variable)
...
Cell References Count (variable)
...

Object Group Object Declaration (variable): A **16-bit** (section 2.2.1.5.1) or **32-bit stream object header** (section 2.2.1.5.2) that specifies an **object group** object declaration.

Object Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the object.

Object Partition ID (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the object partition of the object.

Object Data Size (variable): A **compact unsigned 64-bit integer** that specifies the size in bytes of the binary data opaque to this protocol for the declared object. This MUST match the size of the **binary item** (section 2.2.1.3) in the corresponding **object data** (section 2.2.1.12.6.4) for this object.

Object References Count (variable): A **compact unsigned 64-bit integer** that specifies the number of object references.

Cell References Count (variable): A compact unsigned 64-bit integer that specifies the number of cell references.

2.2.1.12.6.2 Object Data BLOB Declaration

An **Object Data BLOB Declaration** has the following 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
Object Group Object Data BLOB Declaration (variable)																															
...																															
Object Extended GUID (variable)																															
...																															
Object Data BLOB EXGUID (variable)																															

...
Object Partition ID (variable)
...
Object Data Size (variable)
...
Object References Count (variable)
...
Cell References Count (variable)
...

Object Group Object Data BLOB Declaration (variable): A **16-bit** (section 2.2.1.5.1) or **32-bit stream object header** section 2.2.1.5.2) that specifies an **object group object data BLOB declaration**.

Object Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the object.

Object Data BLOB EXGUID (variable): An **extended GUID** that specifies the object data BLOB.

Object Partition ID (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the object partition of the object.

Object Data Size (variable): A **compact unsigned 64-bit integer** that specifies the size in bytes of the opaque binary data for the declared object. This MUST match the size of the **binary item** (section 2.2.1.3) in the corresponding object data BLOB referenced by the **object data BLOB reference** (section 2.2.1.12.6.5) for this object.

Object References Count (variable): A compact unsigned 64-bit integer that specifies the number of object references.

Cell References Count (variable): A compact unsigned 64-bit integer that specifies the number of cell references.

2.2.1.12.6.3 Object Metadata Declaration

An **Object Metadata Declaration** has the following structure.<1>

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Object Group Metadata Declarations (variable)																															

…	
Object Metadata (variable)	
…	
Object Group Metadata Declarations End	

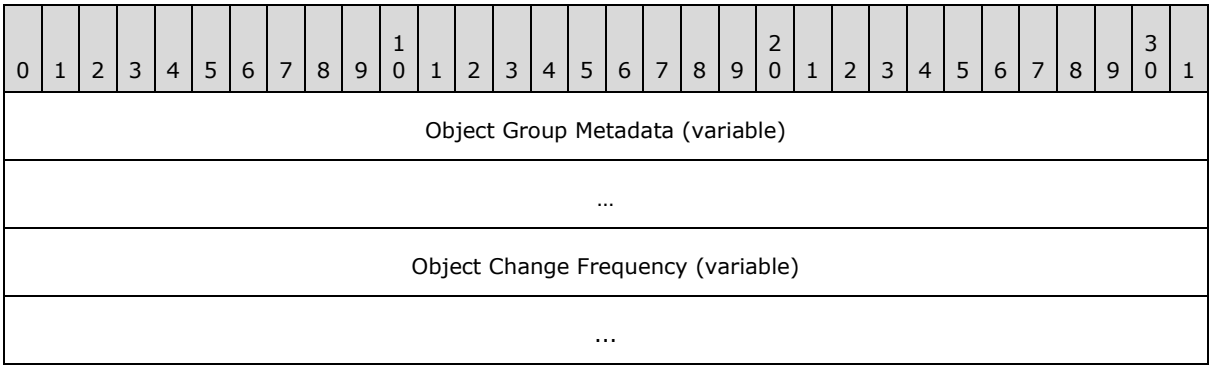
Object Group Metadata Declarations (variable): A **32-bit stream object header** section 2.2.1.5.2) that specifies an object group metadata declarations.

Object Metadata (variable): An array of **object metadata** (section 2.2.1.12.6.3.1) that specifies the object metadata.

Object Group Metadata Declarations End (2 byte): A **16-bit stream object header** (section 2.2.1.5.4) that specifies the end of object group metadata declarations.

2.2.1.12.6.3.1 Object Metadata

An **Object Metadata** has the following structure.



Object Group Metadata (variable): A **32-bit stream object header** (section 2.2.1.5.2) that specifies an object group metadata.

Object Change Frequency (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the expected change frequency of the object. This value MUST be:

- 0, if the change frequency is not known.
- 1, if the object is known to change frequently.
- 2, if the object is known to change infrequently.
- 3, if the object is known to change independently of any other objects.
- 4, if the object is known to change in custom frequencies.

2.2.1.12.6.4 Object Data

An **Object Data** has the following 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
Object Group Object Data or Object Group Object Excluded Data (variable)																															
...																															
Object Extended GUID Array (variable)																															
...																															
Cell ID Array (variable)																															
...																															
Data or Data Size (variable)																															
...																															

Object Group Object Data / Object Group Object Excluded Data (variable): A **16-bit** (section 2.2.1.5.1) or **32-bit stream object header** (section 2.2.1.5.2) that specifies an object group object data or object group object excluded data.

Object Extended GUID Array (variable): An **extended GUID array** (section 2.2.1.8) that specifies the object group.

Cell ID Array (variable): A **cell ID array** (section 2.2.1.11) that specifies the object group.

Data/Data Size (variable): A **binary item** (section 2.2.1.3) that specifies the binary data that is opaque to this protocol in the case of an Object Group Object Data or a **compact unsigned 64-bit integer** that specifies the size in bytes of the opaque binary data for the declared object in the case of an Object Group Object Excluded Data.

2.2.1.12.6.5 Object Data BLOB Reference

An **Object Data BLOB Reference** has the following 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
Object Group Object Data BLOB Reference (variable)																															
...																															
Object Extended GUID Array (variable)																															
...																															
Cell ID Array (variable)																															

...
BLOB Extended GUID (variable)
...

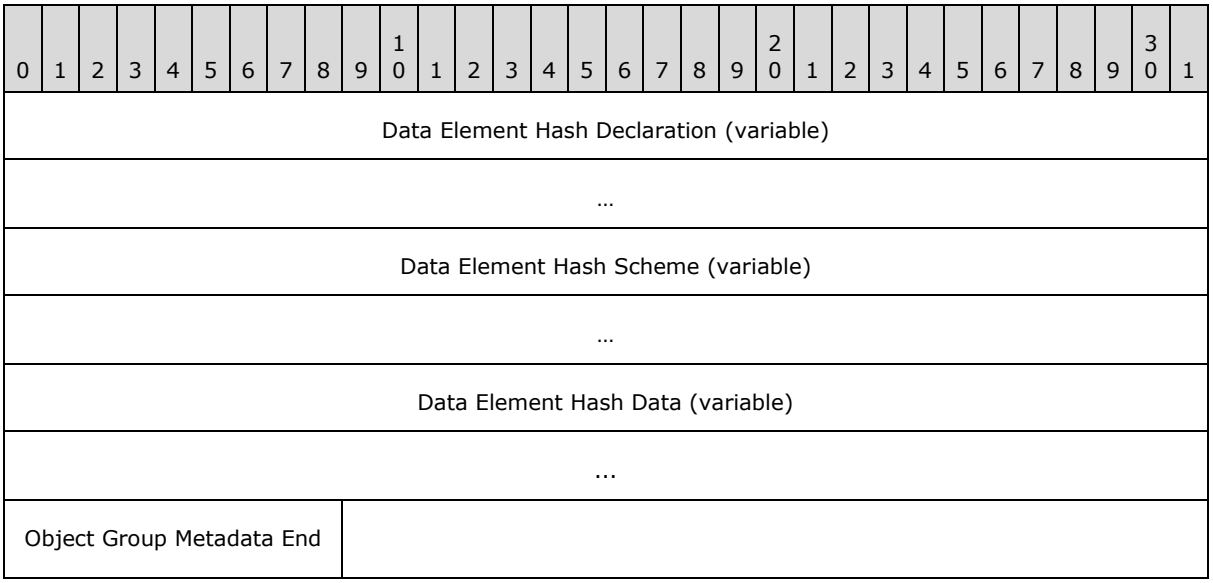
Object Group Object Data BLOB Reference (variable): A **16-bit** (section 2.2.1.5.1) or **32-bit stream object header** (section 2.2.1.5.2) that specifies an object group **object data BLOB reference** (section 2.2.1.12.6.5).

Object Extended GUID Array (variable): An **extended GUID array** (section 2.2.1.8) that specifies the object references.

Cell ID Array (variable): A **cell ID array** (section 2.2.1.11) that specifies the cell references.

BLOB Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the object data BLOB.

2.2.1.12.6.6 Data Element Hash



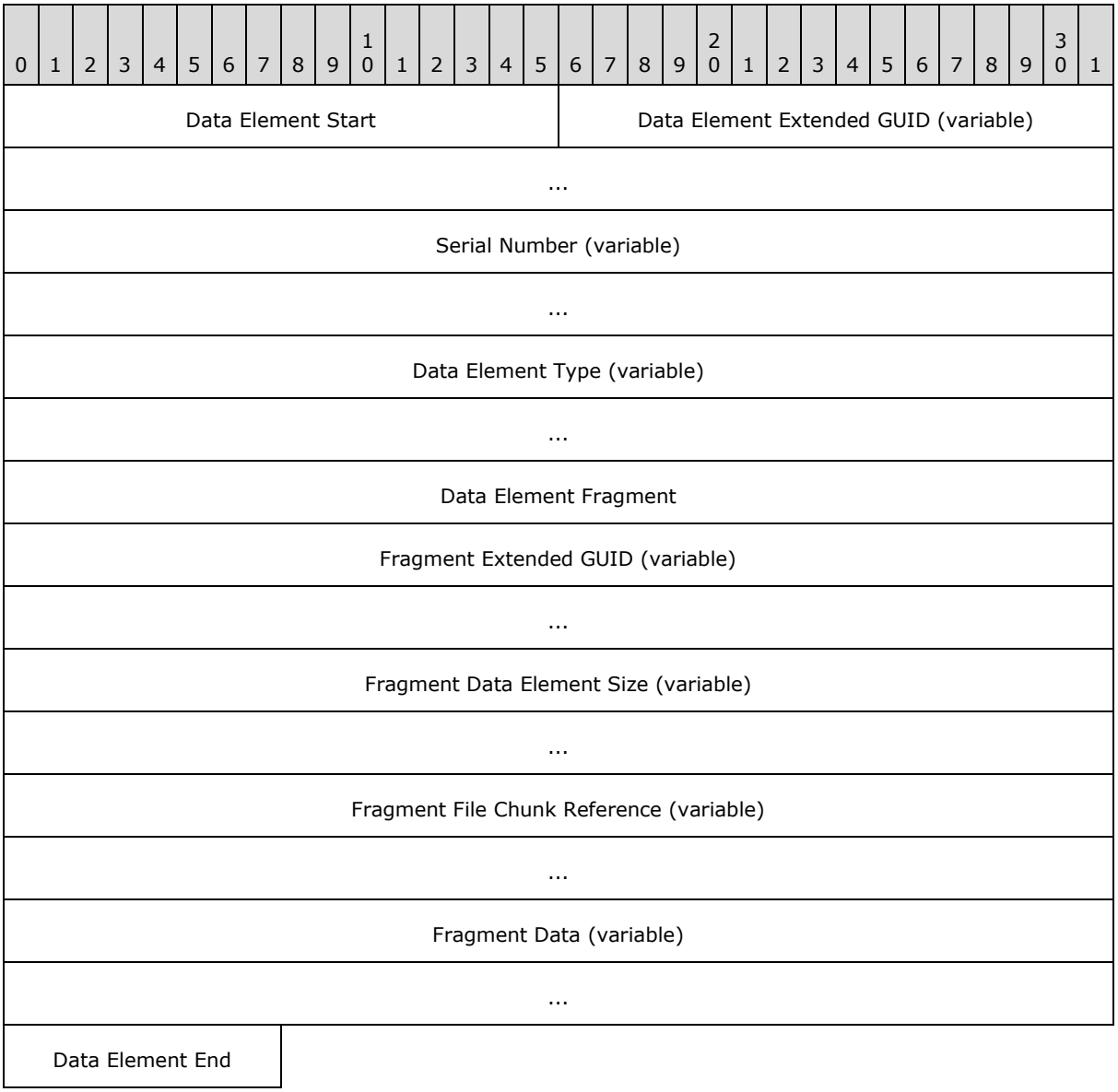
Data Element Hash Declaration (variable): A **16-bit** (section 2.2.1.5.1) or **32-bit stream object header** section 2.2.1.5.2) that specifies a **data element hash declaration**.

Data Element Hash Scheme (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the hash schema. This value MUST be 1, indicating Content Information Data Structure Version 1.0 as specified in [MS-PCCRC] section 2.3.

Data Element Hash Data (variable): A **binary item** (section 2.2.1.3) that specifies the hash. The hash data MUST be encoded in the schema specified by the **data element hash scheme**.

2.2.1.12.7 Data Element Fragment Data Elements

A **Data Element Fragment data element** has the following structure.



Data Element Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a data element start.

Data Element Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the data element.

Serial Number (variable): A **serial number** (section 2.2.1.9) that specifies the data element.

Data Element Type (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the value of the **object data BLOB data element** type (section 2.2.1.12.8).

Data Element Fragment (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a data element fragment.

Fragment Extended GUID (variable): An **extended GUID** that specifies the data element fragment.

Fragment Data Element Size (variable): A **compact unsigned 64-bit integer** that specifies the size in bytes of the fragmented data element.

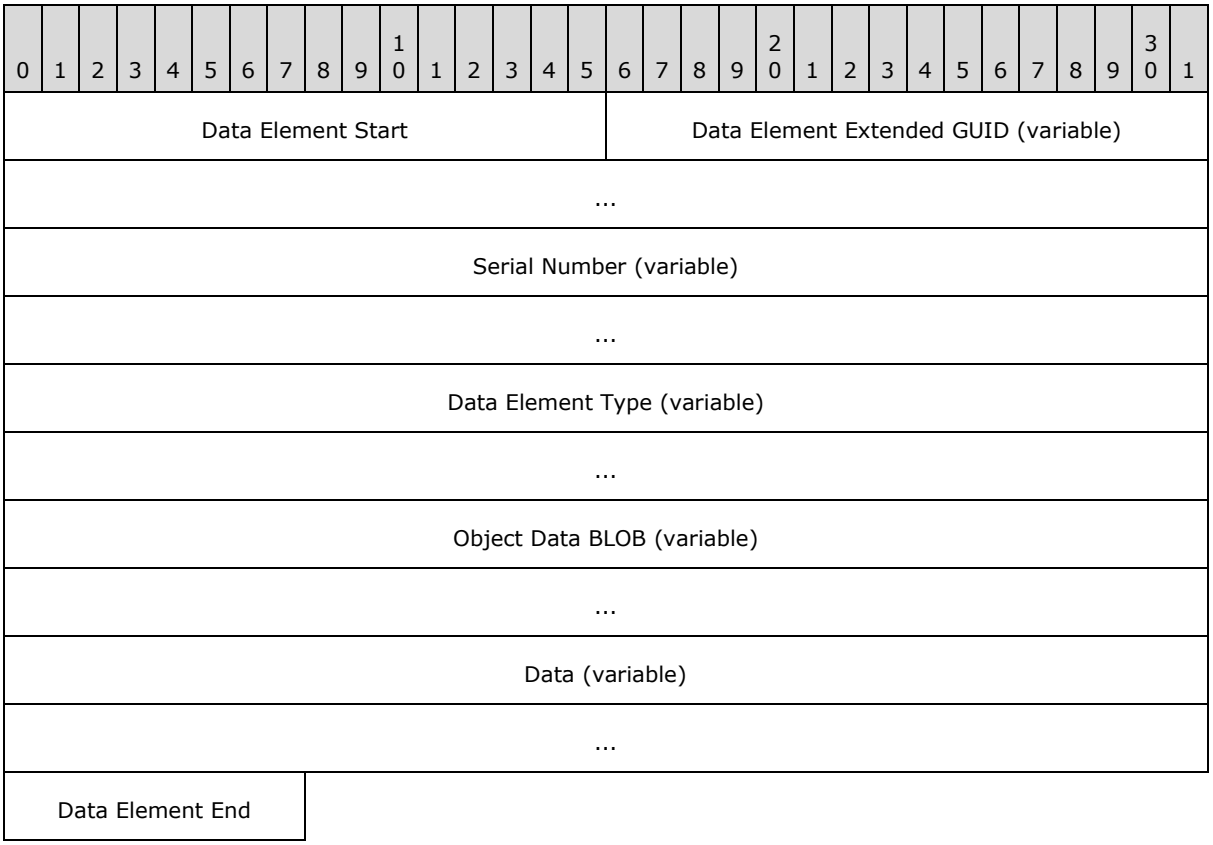
Fragment File Chunk Reference (variable): A **file chunk reference** (section 2.2.1.2) that specifies the data element fragment.

Fragment Data (variable): A byte stream that specifies the binary data opaque to this protocol.

Data Element End (1 byte): An **8-bit stream object header** (section 2.2.1.5.3) that specifies a data element end.

2.2.1.12.8 Object Data BLOB Data Elements

The **Object Data BLOB data element** has the following format.



Data Element Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a data element start.

Data Element Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the data element.

Serial Number (variable): A **serial number** (section 2.2.1.9) that specifies the data element.

Data Element Type (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the value of the **object data BLOB data element** type.

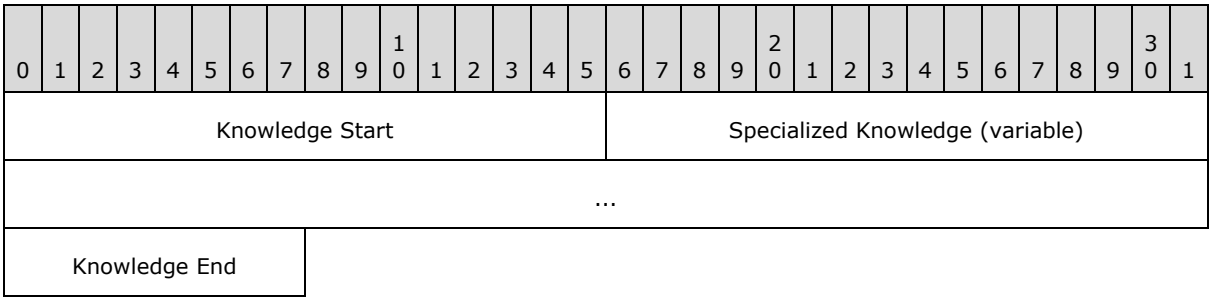
Object Data BLOB (variable): A **16-bit** or **32-bit stream object header** that specifies an object data BLOB.

Data (variable): A byte stream that specifies the binary data opaque to this protocol.

Data Element End (1 byte): An **8-bit stream object header** (section 2.2.1.5.3) that specifies a data element end.

2.2.1.13 Knowledge

The **Knowledge** type specifies what the client knows about a state of a file. It has the following format.



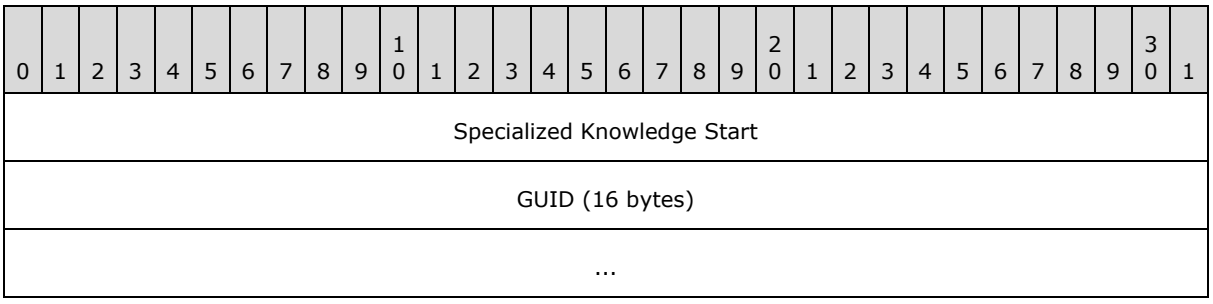
Knowledge Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a **knowledge** (section 2.2.1.13) start.

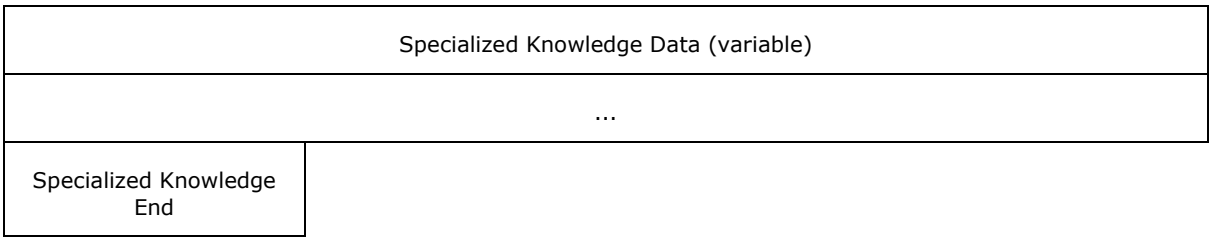
Specialized Knowledge (variable): Zero or more **specialized knowledge** structures (section 2.2.1.13.1).

Knowledge End (1 byte): An **8-bit stream object header** (section 2.2.1.5.3) that specifies a **knowledge** end.

2.2.1.13.1 Specialized Knowledge

Contains the **specialized knowledge** state in the following format.





Specialized Knowledge Start (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies A **specialized knowledge** start.

GUID (16 bytes): A GUID that specifies the type of **specialized knowledge**. The following **GUIDs** detail the type of knowledge contained:

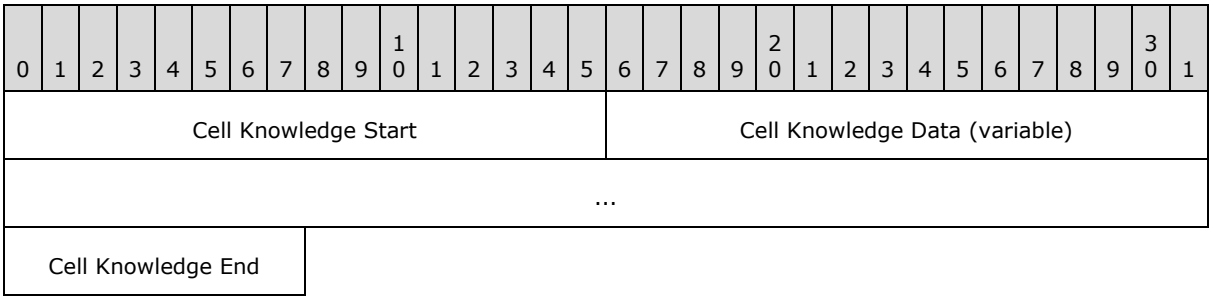
GUID (string representation)	Knowledge Type
{327A35F6-0761-4414-9686-51E900667A4D}	**Cell knowledge** (section 2.2.1.13.2)
{3A76E90E-8032-4D0C-B9DD-F3C65029433E}	**Waterline knowledge** (section 2.2.1.13.4)
{0ABE4F35-01DF-4134-A24A-7C79F0859844}	**Fragment knowledge** (section 2.2.1.13.3)
{10091F13-C882-40FB-9886-6533F934C21D}	**Content tag knowledge** (section 2.2.1.13.5)

Specialized Knowledge Data (variable): The data for the specific knowledge type.

Specialized Knowledge End (2 bytes): A **16-bit stream object header** (section 2.2.1.5.4) that specifies **specialized knowledge** end.

2.2.1.13.2 Cell Knowledge

Specifies the data element knowledge of the client in the following format.



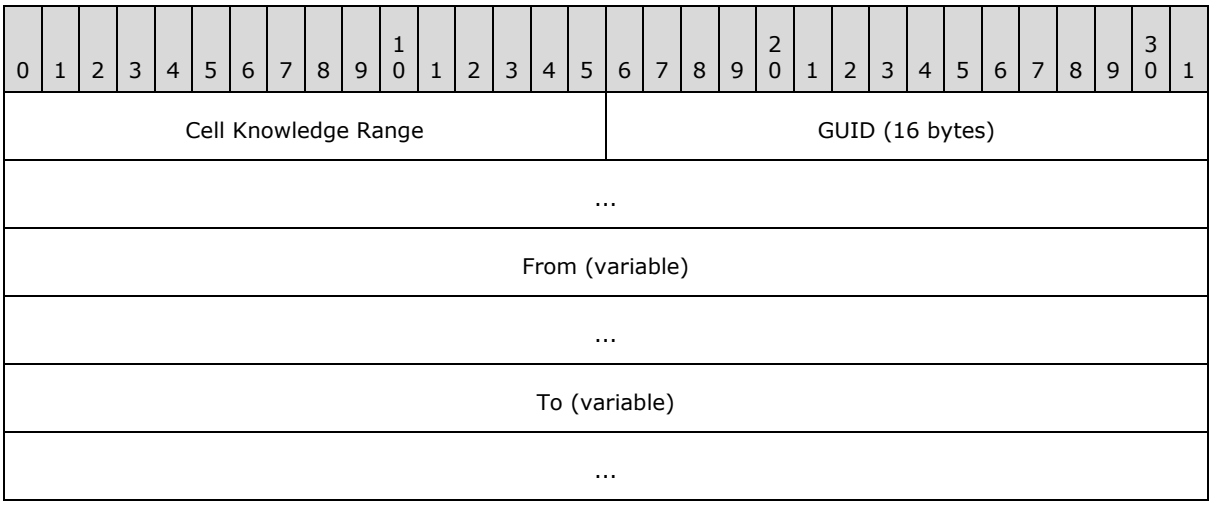
Cell Knowledge Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a **cell knowledge** start.

Cell Knowledge Data (variable): A **cell knowledge entry** (section 2.2.1.13.2.2) or **cell knowledge range** (section 2.2.1.13.2.1) that specifies one or more data element knowledge references.

Cell Knowledge End (1 byte): An **8-bit stream object header** (section 2.2.1.5.3) that specifies the **cell knowledge** end.

2.2.1.13.2.1 Cell Knowledge Range

A **cell knowledge range** of data elements specifies knowledge about a range of cells in the following format.



Cell Knowledge Range (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies the start of a **cell knowledge range**.

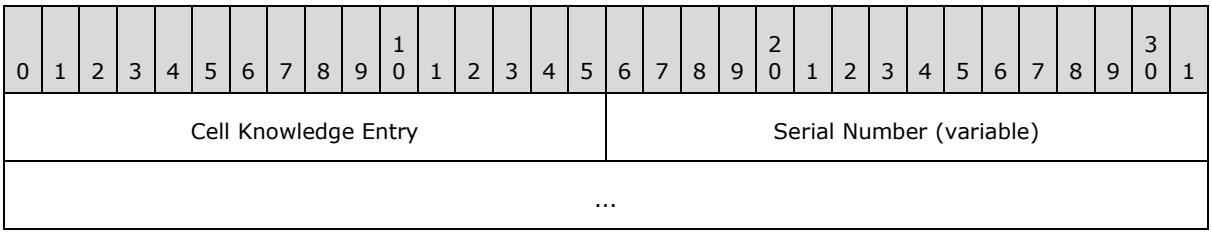
GUID (16 bytes): A GUID that specifies the data element. Combined with the **From** sequence number, it forms the starting **serial number** (section 2.2.1.9) of the range. Combined with the **To** sequence number, it forms the ending **serial number** of the range.

From (variable): A **compact unsigned 64-bit integer** section 2.2.1.1() that specifies the starting sequence number. When combined with the GUID, it forms the **serial number** of the starting data element in the range.

To (variable): A **compact unsigned 64-bit integer** that specifies the ending sequence number. When combined with the GUID, it forms the **serial number** of the ending data element in the range.

2.2.1.13.2.2 Cell Knowledge Entry

A **cell knowledge entry** specifies the knowledge for a single cell in the following format.

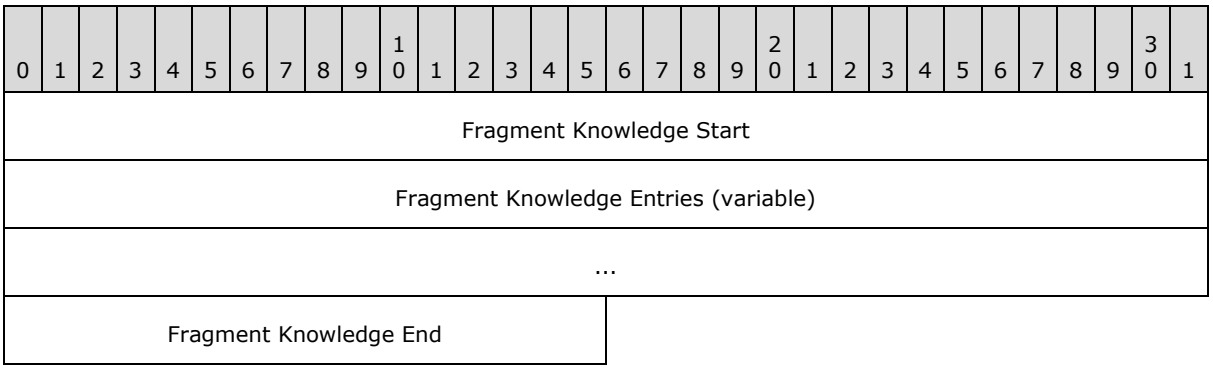


Cell Knowledge Entry (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1), that specifies a **cell knowledge entry**.

Serial Number (variable): A **serial number** (section 2.2.1.9) that specifies the cell.

2.2.1.13.3 Fragment Knowledge

The **Fragment Knowledge** tells the client which fragments of a large data element have been uploaded to the server in the following format.



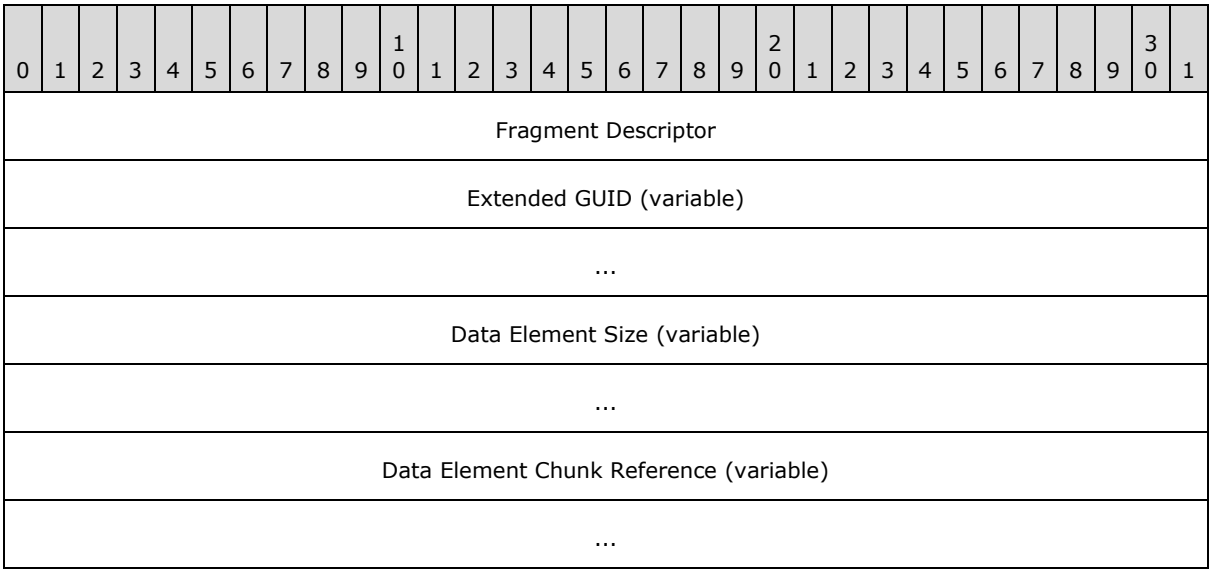
Fragment Knowledge Start (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **fragment knowledge** start.

Fragment Knowledge Entries (variable): One or more **fragment knowledge entry** (section 2.2.1.13.3.1) structures specifying the fragments which have been uploaded.

Fragment Knowledge End (2 bytes): A **16-bit stream object header** (section 2.2.1.5.4) that specifies a **Fragment knowledge** end.

2.2.1.13.3.1 Fragment Knowledge Entry

The **Fragment Knowledge Entry** tells the client that the specified fragment of a large data element has been uploaded to the server in the following format.



Fragment Descriptor (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **fragment knowledge entry**.

Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the data element this **fragment knowledge entry** contains knowledge about.

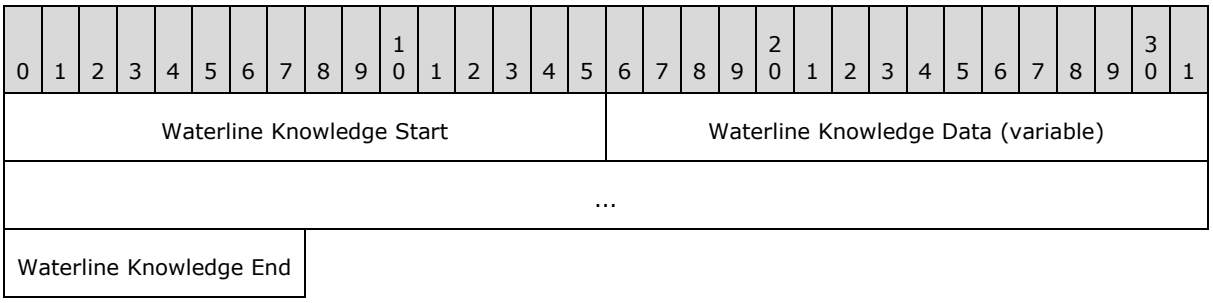
Data Element Size (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) specifying the size in bytes of the data element specified by the preceding **Extended GUID**.

Data Element Chunk Reference (variable): A **file chunk reference** (section 2.2.1.2) specifying which part of the data element with the preceding GUID this **fragment knowledge entry** contains knowledge about.

2.2.1.13.4 Waterline Knowledge

The **Waterline Knowledge** specifies the current server waterline, which is the **serial number** (section 2.2.1.9) greater than or equal to the **serial number** of all the cells on the server that the client has downloaded. It allows server implementations to improve performance by skipping over cells the client has already downloaded for some operations.

It has the following format.



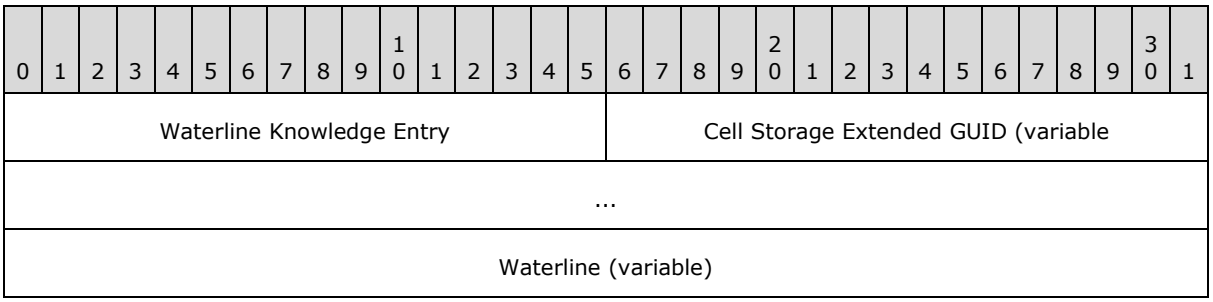
Waterline Knowledge Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a **waterline knowledge** start.

Waterline Knowledge Data (variable): One or more **waterline knowledge entries** (section 2.2.1.13.4.1) that specify what the server has already delivered to the client or what the client has already received from the server.

Waterline Knowledge End (1 byte): An **8-bit stream object header** (section 2.2.1.5.3) that specifies the **waterline knowledge** end.

2.2.1.13.4.1 Waterline Knowledge Entry

The **Waterline Knowledge Entry** specifies the current server waterline for the specified cell storage in the following format.



...
Reserved (variable)
...

Waterline Knowledge Entry (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies a **waterline knowledge entry**.

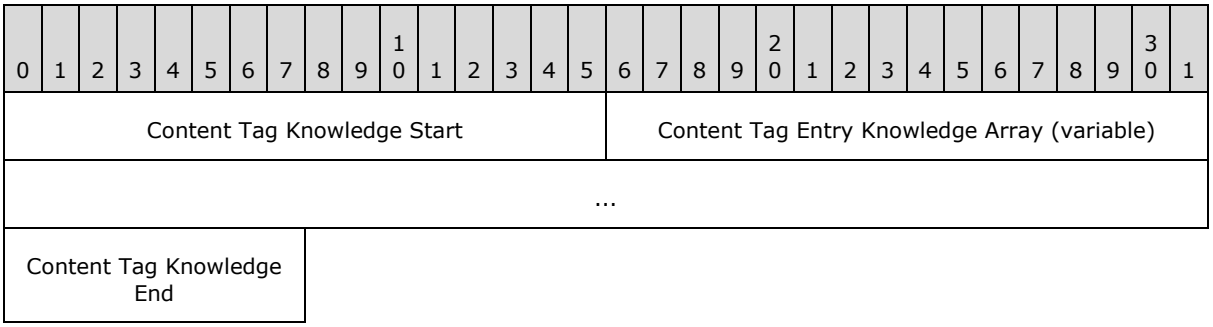
Cell Storage Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the cell storage this entry specifies the waterline for.

Waterline (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies a sequential **serial number** (section 2.2.1.9).

Reserved (variable): A **compact unsigned 64-bit integer** that specifies a reserved field that MUST have value of zero and MUST be ignored.

2.2.1.13.5 Content Tag Knowledge

The **Content Tag Knowledge** specifies the content tag for each BLOB heap in the following format.



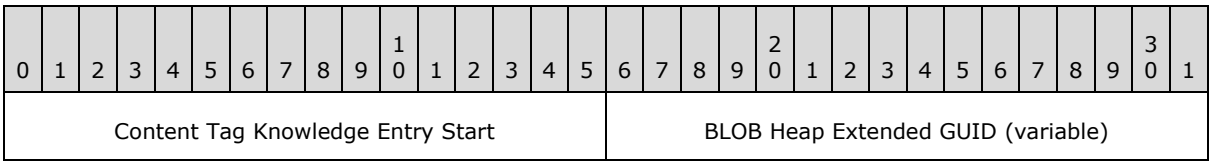
Content Tag Knowledge Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies the **content tag knowledge** start.

Content Tag Knowledge Entry Array (variable): An array of **content tag knowledge entries** (section 2.2.1.13.5.1) that specifies the BLOB heap entries.

Content Tag Knowledge End (1 byte): An **8-bit stream object header** (section 2.2.1.5.3) that specifies the **content tag knowledge** end.

2.2.1.13.5.1 Content Tag Knowledge Entry

The **Content Tag Knowledge Entry** stores a BLOB heap GUID and its associated content tag in the following format.



...
Clock Data (variable)
...

Content Tag Knowledge Entry Start (2 bytes): A **16-bit stream object header** (section 2.2.1.5.1) that specifies the start of a **content tag entry**.

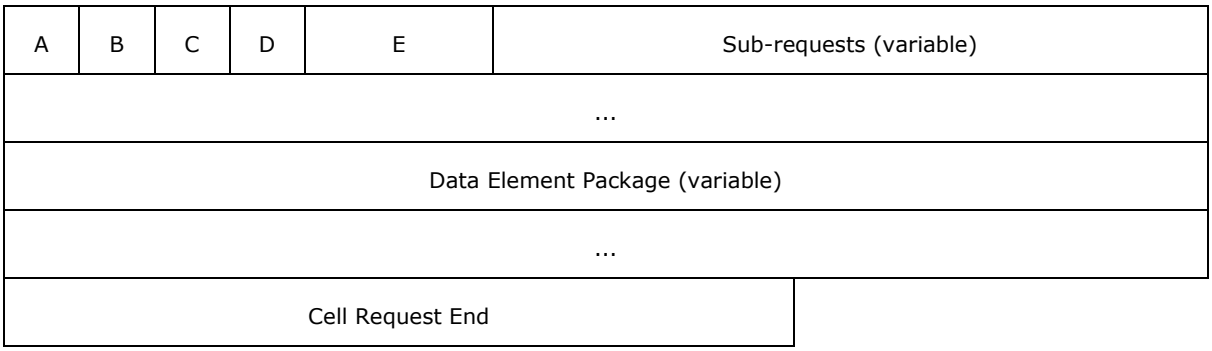
BLOB Heap Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the BLOB heap this content tag is for.

Clock Data (variable): A **binary item** (section 2.2.1.3) that specifies changes when the contents of the BLOB heap change on the server.

2.2.2 Request Message Syntax

A request message specifies the format used to contain **sub-requests** (section 2.2.2.1) in the following format.<2>

<table>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>20</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>30</th><th>1</th></tr>
<tr><td colspan="21">Protocol Version</td><td colspan="11">Minimum Version</td></tr>
<tr><td colspan="32">Signature</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Request Start</td></tr>
<tr><td colspan="32">User Agent Start</td></tr>
<tr><td colspan="32">User Agent GUID</td></tr>
<tr><td colspan="32">GUID (16 bytes)</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">User Agent Version</td></tr>
<tr><td colspan="21">User Agent End</td><td colspan="11">Request Hashing Options Declaration</td></tr>
<tr><td colspan="21">...</td><td colspan="11">Request Hashing Schema (variable)</td></tr>
<tr><td colspan="32">...</td></tr>
</table>



Protocol Version (2bytes): An unsigned integer that specifies the protocol schema version number used in this request. This value MUST be 12.

Minimum Version (2 bytes): An unsigned integer that specifies the oldest version of the protocol schema that this schema is compatible with. This value MUST be 11.

Signature (8 bytes): An unsigned integer that specifies a constant signature that identifies this as a request. This MUST be set to 0x9B069439F329CF9C.

Request Start (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a request start.

User Agent Start (4 bytes): A **32-bit stream object header** that specifies a user agent start.

User Agent GUID (4 bytes): A **32-bit stream object header** that specifies a user agent GUID.

GUID (16 bytes): A GUID that specifies the user agent.

User Agent Version (4 bytes): A **32-bit stream object header** that specifies a user agent version.

Version (4 bytes): An unsigned integer that specifies the version of the client. The integer MUST be greater than or equal to 0xFA12994.

User Agent End (2 bytes): A **16-bit stream object header** (section 2.2.1.5.4) that specifies a user agent end.

Request Hashing Options Declaration: A 32-bit stream object header that specifies a request hashing options declaration.<3>

Request Hashing Schema: A **compact unsigned 64-bit integer** that specifies the Hashing Schema being requested that must be set to 1 indicating Content Information Data Structure Version 1.0 as specified in [MS-PCCRC] section 2.3.

A – Reserved (1 bit): A reserved bit that MUST be set to zero and MUST be ignored.

B – Reserved (1 bit): A reserved bit that MUST be set to zero and MUST be ignored.

C – Request Data Element Hashes Instead of Data (1 bit): If set, a bit that specifies to exclude object data and instead return data element hashes; otherwise, object data is included.

D – Request Data Element Hashes (1 bit): If set, a bit that specifies to include data element hashes (if available) when returning data elements; otherwise data element hashes should not

be returned. If data element hashes are returned they MUST be encoded in the schema specified by **Request Hashing Schema**.

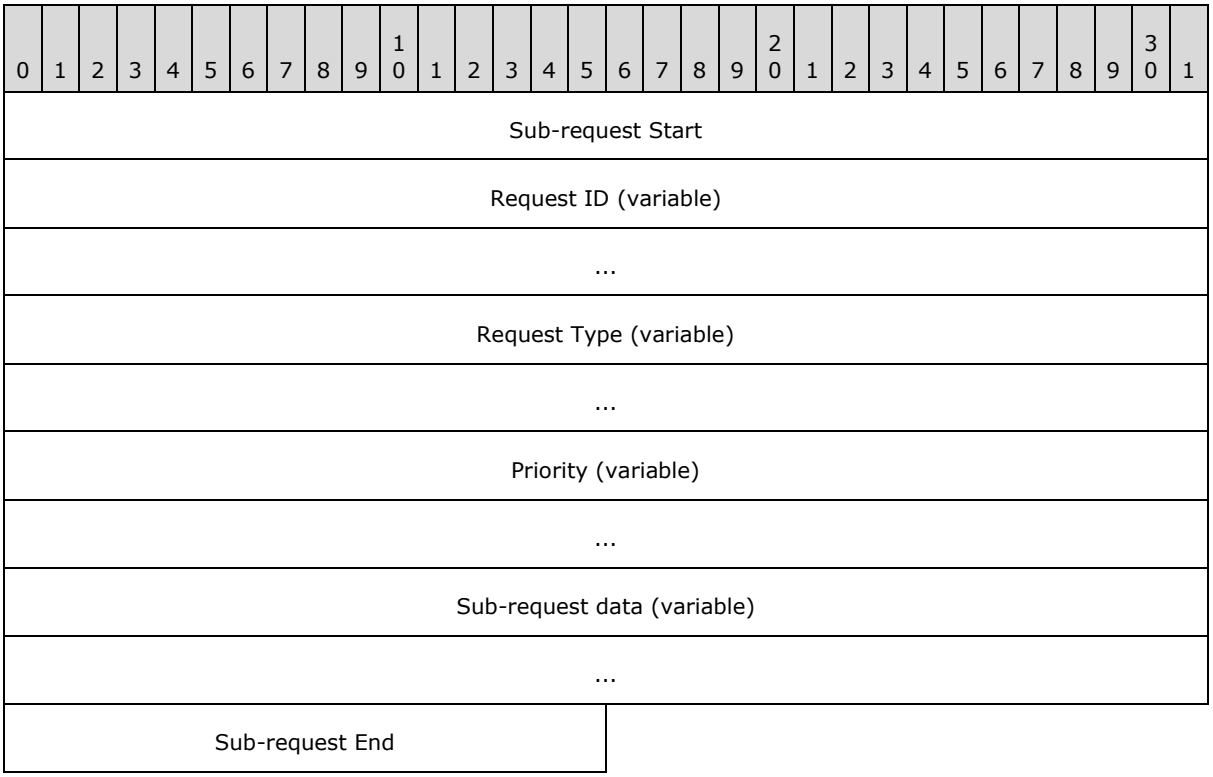
E - Reserved1 (4 bits): A 4-bit reserved field that MUST be set to zero and MUST be ignored.**Sub-requests (variable):** An array of **sub-requests** (section 2.2.2.1), that specifies request information to execute on the server.

Data Element Package (variable): A **data element package** (section 2.2.1.12) that specifies the serialized data elements for the request. **Put Changes** (section 2.2.2.1.4) is the only **sub-request** type that can be used to reference data elements within this package.

Cell Request End (2 bytes): A **16-bit stream object header** that specifies a cell request end.

2.2.2.1 Sub-Requests

A **sub-request** has the following format.



Sub-request Start (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **sub-request** start.

Request ID (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that MUST be less than 0xFFFFFFFF that specifies the request number for each **sub-request**. The **Request ID** for this **sub-request** MUST be unique within the **sub-requests** in the request.

Request Type (variable): A **compact unsigned 64-bit integer** that specifies the **request type** (section 2.2.1.6).

Priority (variable): A **compact unsigned 64-bit integer** that specifies the priority of the **sub-request**. The server executes the set of **sub-requests** contained in a request in ascending

numerical priority. **Sub-requests** with the same priority are executed in any order with respect to each other.

Target Partition Id (variable): An optional **target partition id** (section 2.2.2.1.1) that specifies the target partition for this **sub-request**.<4>

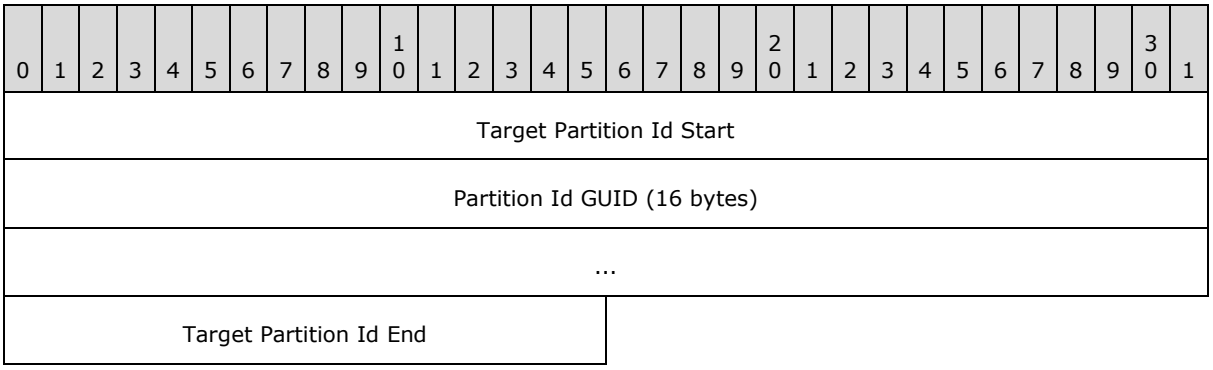
Sub-request data (variable): A structure that specifies additional data for this **sub-request**. The structure used depends on the value of the **Request Type**, as specified by the following table:

Value	Meaning
1	Query access (section 2.2.2.1.2).
2	Query changes (section 2.2.2.1.3).
5	Put changes (section 2.2.2.1.4).
11	Allocate Extended Guid Range (section 2.2.2.1.5).

Sub-request End (2 bytes): A **16-bit stream object header** (section 2.2.1.5.4) that specifies a **sub-request** end.

2.2.2.1.1 Target Partition Id

The **Target partition Id** structure has the following format.



Target Partition Id Start (variable): A **32-bit stream object header** (section 2.2.1.5.2) that specifies the beginning of a **target partition id**.

Partition Id GUID (16 bytes): A GUID that specifies the partition.

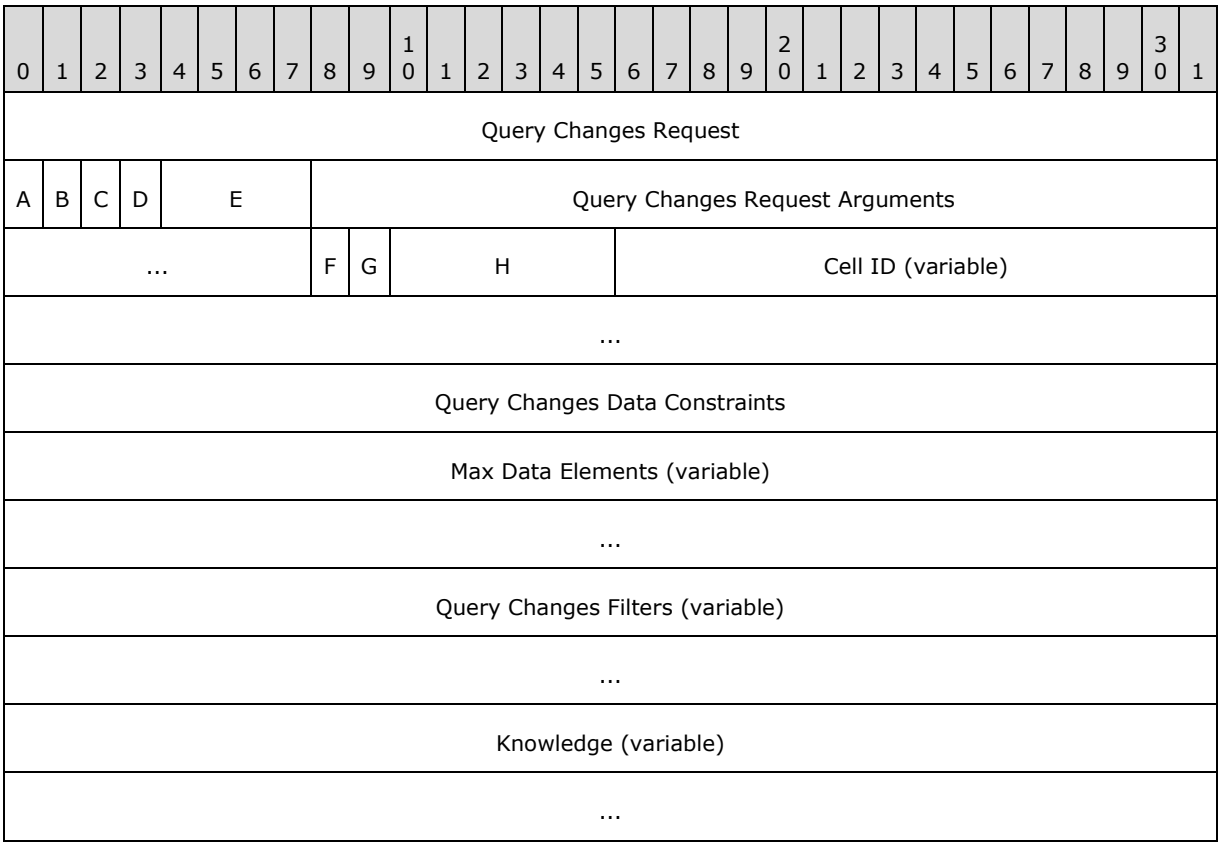
Target Partition Id End (2 bytes): A **16-bit stream object header** (section 2.2.1.5.4) that specifies the end of a **target partition id**.

2.2.2.1.2 Query Access

Query access does not have any **sub-request** data.

2.2.2.1.3 Query Changes

The **Query Changes sub-request** has the following format.



Query Changes Request (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies the beginning of a **Query Changes** request.

A – Reserved (1 bit): A reserved bit that MUST be set to zero and MUST be ignored.

B – Allow Fragments (1 bit): If set, a bit that specifies to allow fragments; otherwise, it does not allow fragments.

C –Exclude Object Data (1 bit): If set, a bit that specifies to exclude object data; otherwise, object data is included.

D – Include Filtered Out Data Elements In Knowledge (1 bit): If set, a bit that specifies to include the **serial numbers** (section 2.2.1.9) of filtered out data elements in the response **knowledge**; otherwise, the **serial numbers** of filtered out data elements are not included in the response **knowledge**.

E - Reserved1 (4 bits): A 4-bit reserved field that MUST be set to zero and MUST be ignored.

Query Changes Request Arguments (4 bytes): A **32-bit stream object header** that specifies a **Query Changes** request arguments.

F – Include Storage Manifest (1 bit): If set, a bit that specifies to include the storage manifest; otherwise, the storage manifest is not included.

G – Include Cell Changes (1 bit): If set, a bit that specifies to include cell changes; otherwise, cell changes are not included.

H - Reserved2 (6 bits): A 6-bit reserved field that MUST be set to zero and MUST be ignored.

Cell ID (variable): A **cell ID** (section 2.2.1.10) that specifies if the **Query Changes** are scoped to a specific cell. If the **Cell ID** is 0x0000, no scoping restriction is specified.

Query Changes Data Constraints (4 bytes): A **32-bit stream object header** that specifies a **Query Changes** data constraints; this is optional if no maximum data elements constraint is required.

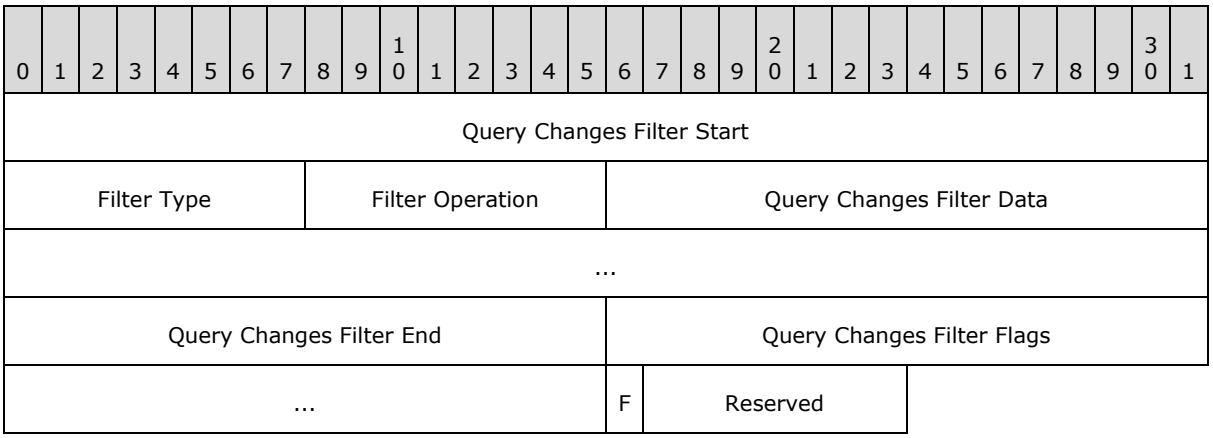
Max Data Elements (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies limit of DoS mitigation at which the server starts breaking up the results into partial results. If the client specifies anything higher, the server truncates it to this value.

Query Changes Filters (variable): An optional ordered array of **filters** (section 2.2.2.1.3.1) that specifies how the results of the query will be filtered before it is returned to the client.

Knowledge (variable): An optional **knowledge** (section 2.2.1.13) that specifies what the client knows about a state of a file.

2.2.2.1.3.1 Filters

Filters have the following format.



Query Changes Filter Start (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **Query Changes** filter start.

Filter Type (1 byte): An unsigned integer that specifies filter type as follows:

Value	Type
1	All filter (section 2.2.2.1.3.1.1).
2	Data element type filter (section 2.2.2.1.3.1.2).
3	Storage index referenced data elements filter (section 2.2.2.1.3.1.3).
4	Cell ID filter (section 2.2.2.1.3.1.4).
5	Custom filter (section 2.2.2.1.3.1.5).
6	Data element IDs filter (section 2.2.2.1.3.1.6).

Value	Type
7	Hierarchy filter (section 2.2.2.1.3.1.7).

Filter Operation (1 byte): A flag that specifies how the filter is applied to the data elements before they are added to the response **data element package** (section 2.2.1.12). This field MUST be set to zero or one. A value of zero specifies that any data elements matching the filter will be excluded from the response **data element package**. A value of one specifies that any data elements matching the filter will be included in the response **data element package**, even if they have been excluded by another filter prior to this filter in the ordered array of filters.

Query Changes Filter Data (variable): A structure that specifies additional data based on the filter type.

Query Changes Filter End (2 bytes): A **16-bit stream object header** (section 2.2.1.5.4) that specifies the end of a **Query Changes** filter.

Query Changes Filter Flags (4 bytes): An optional **32-bit stream object header** that specifies the beginning of **Query Changes** filter flags.

F – Fail if Unsupported (1 bit): If set, a bit that specifies to one to allow failure if a filter is not supported; otherwise, unsupported filters are ignored. This bit is only sent if **Query Changes Filter Flags** are specified.

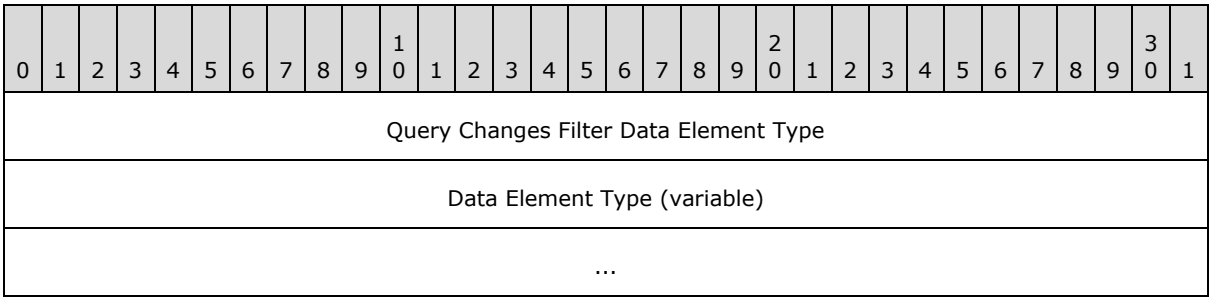
Reserved (7 bits): A 7-bit reserved field that MUST be set to zero, and MUST be ignored if **Query Changes Filter Flags** are specified.

2.2.2.1.3.1.1 All Filter

The **All Filter** specifies a filter that matches all data elements. This filter does not contain any data.

2.2.2.1.3.1.2 Data Element Type Filter

The **Data Element Type Filter** specifies a filter that matches data elements of a specific type in the following format.



Query Changes Filter Data Element Type (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies the beginning of a **Query Changes** filter data element type.

Data Element Type (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the data element type the filter matches:

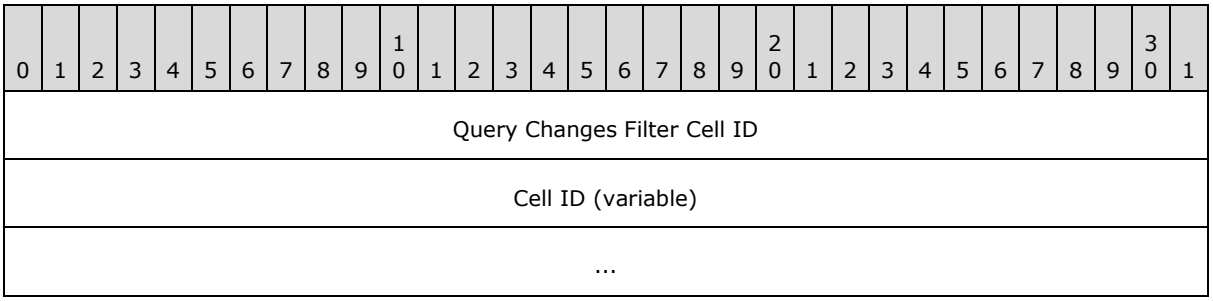
Value	Data element type
0	None.
1	Storage Index.
2	Storage Manifest.
3	Cell Manifest.
4	Revision Manifest.
5	Object Group.
6	Data element fragment.
10	Object data BLOB.

2.2.2.1.3.1.3 Storage Index Referenced Data Elements Filter

The **Storage Index Referenced Data Elements Filter** specifies a filter that matches any data element referenced by a value in the storage index. This filter is not currently supported by the server.

2.2.2.1.3.1.4 Cell ID Filter

The **Cell ID Filter** specifies a filter that matches a data element connected to the data element sub-graph whose root is the storage index key with a specified **cell ID** (section 2.2.1.10), in the following format.

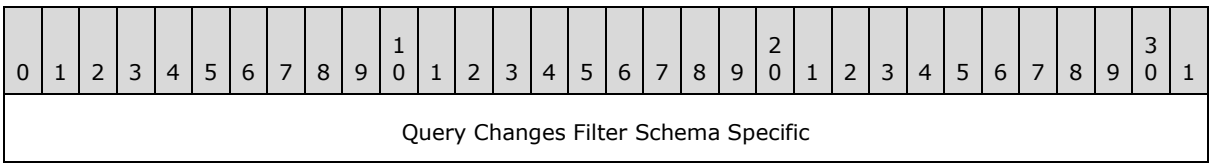


Query Changes Filter Cell ID (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **Query Changes** filter cell identifier.

Cell ID (variable): A **cell ID** (section 2.2.1.10) that specifies the root of the data element sub-graph to which data elements MUST be connected to match the filter.

2.2.2.1.3.1.5 Custom Filter

The **Custom Filter** specifies a custom filter to apply in the following format.



Schema GUID (16 bytes)
...
Schema Filter Data (variable)
...

Query Changes Filter Schema Specific (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **Query Changes** filter schema specific.

Schema GUID (16 bytes): A GUID that specifies the schema specific filter opaque to this protocol.

Schema Filter Data (variable): A byte stream that specifies the schema filters data opaque to this protocol.

2.2.2.1.3.1.6 Data Element IDs Filter

The **Data Element IDs Filter** specifies a filter that matches data elements who's **Data Element IDs** are contained in a specified list in the following format.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Query Changes Filter Data Element IDs																															
Data Element IDs (variable)																															
...																															

Query Changes Filter Data Element IDs (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **Query Changes** filter data element identifiers.

Data Element IDs (Variable): An **extended GUID array** (section 2.2.1.8) that specifies the data element identifiers.

2.2.2.1.3.1.7 Hierarchy Filter

The **Hierarchy Filter** specifies a filter that matches any data element connected to the data element sub-graph whose root is the specified storage index key up to a specified depth.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Query Changes Filter Hierarchy																															
Depth								Storage Index Extended GUID Array																							

...

Query Changes Filter Hierarchy (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **Query Changes** filter hierarchy.

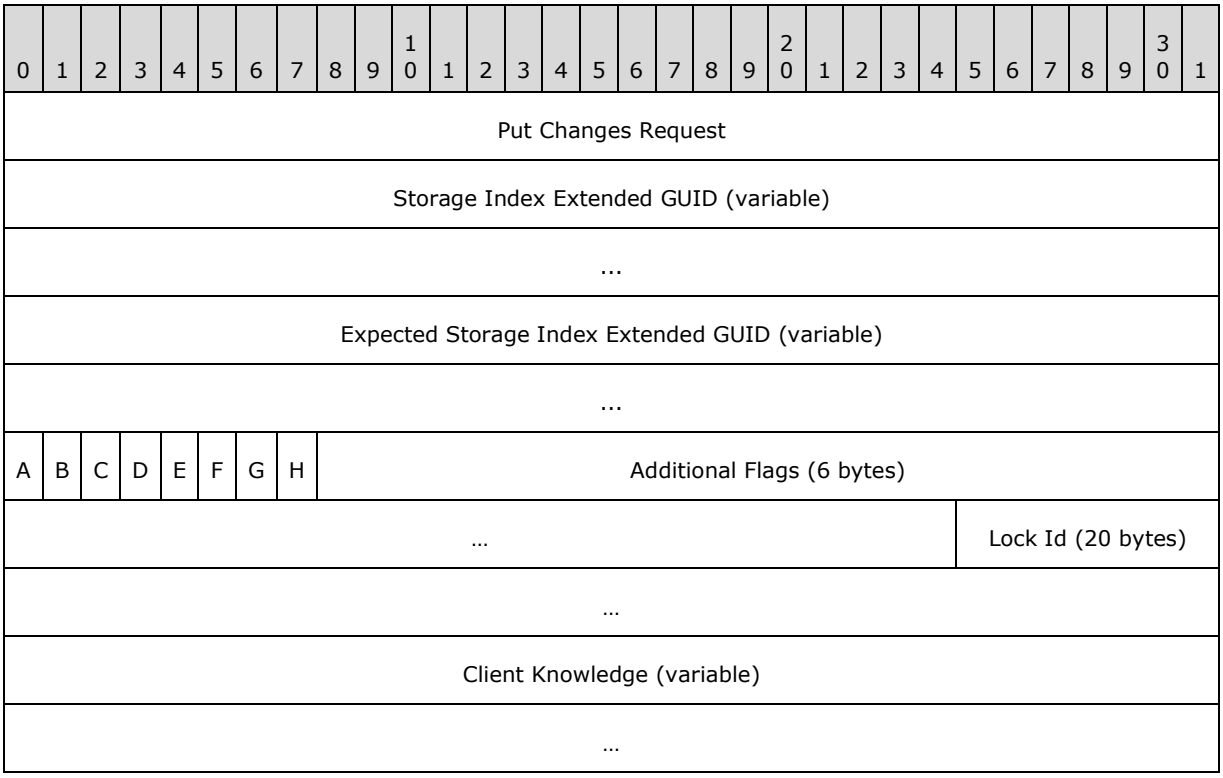
Depth (1 byte): An unsigned integer that specifies the depth of the sub-graph and MUST be one of the following values:

Value	Meaning
0	Index values corresponding to the specified keys only.
1	First data elements referenced by the storage index values corresponding to the specified keys only.
2	Single level. All data elements under the sub-graphs rooted by the specified keys stopping at any storage index entries.
3	Deep. All data elements and storage index entries under the sub-graphs rooted by the specified keys.

Storage Index Extended GUID Array (variable): An **extended GUID array** (section 2.2.1.8) that specifies the storage index keys.

2.2.2.1.4 Put Changes

The **Put Changes sub-reques**t has the following format.



Put Changes Request (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **Put Changes** request.

Storage Index Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies the data element identifier of a storage index in the request's **data element package** (section 2.2.1.12). The storage index specifies the changes that will be applied by the protocol server.

Expected Storage Index Extended GUID (variable): An **extended GUID** that specifies the data element identifier of a storage index, and the expected storage index in the **data element package** of the request. When the protocol server applies updates to the server's storage index, it first checks the specified expected storage index. If the expected storage index was specified and the key that is to be updated in the protocol server's storage index exists in the expected storage index, the corresponding values in the protocol server's storage index and the expected storage index MUST match; otherwise, the protocol server MUST return a **Cell Error** failure value of 12 indicating a coherency failure as specified in section 2.2.3.2.1. If the values match the protocol server MUST ensure that the storage index update is atomically updated such that no other update is overwritten. If the expected storage index was not specified or the key that is to be updated in the protocol server's storage index does not exist in the expected storage index, the **Imply Null Expected if No Mapping** flag MUST be evaluated. If this flag is zero, the protocol server MUST apply the change without checking the current value; otherwise, if the flag specifies one, the protocol server MUST only apply the change if no mapping exists (the key that is to be updated in the protocol server's storage index doesn't exist or it maps to nil). If a mapping exists, the protocol server MUST return a **Cell Error** failure value of 12 indicating a coherency failure as specified in section 2.2.3.2.1.

A - Imply Null Expected if No Mapping (1 bit): A bit that specifies the behavior of checking the current storage index value prior to update, if no expected storage index entry is specified by the client.

The expected storage index is the basis for what the client believes is the current state of the storage index, if set to one; otherwise, the expected storage index is not specified.

B - Partial (1 bit): A bit that specifies that this is a partial **Put Changes**, and not the full changes.

C - Partial Last (1 bit): A bit that specifies if this is the last **Put Changes** in a partial set of changes.

D - Favor Coherency Failure Over Not Found (1 bit): A bit that specifies to force a coherency check on the server, if a **Referenced Data Element Not Found** (section 2.2.3.2.1) failure occurred. This MAY result in a **Coherency Failure** returned instead of **Referenced Data Element Not Found**.

E - Abort Remaining Put Changes on Failure (1 bit): If set, a bit that specifies to abort remaining **Put Changes** on failure. Its value is ignored by the server.

F - Multi-Request Put Hint (1 bit): A bit that specifies to reduce the number of auto coalesces during multi-request put scenarios. If only one request for a **Put Changes**, this bit is zero.

G - Return Complete Knowledge If Possible (1 bit): A bit that specifies to return the complete **knowledge** (section 2.2.1.13) from the server, provided that this request has exclusive access to the **knowledge**. Exclusive **knowledge** access is only granted on Coalesce, and therefore complete **knowledge** will not be returned in non-coalescing **sub-requests**.

H - Last Writer Wins On Next Change (1 bit): A bit that specifies to allow the **Put Changes** to be subsequently overwritten on the next **Put Changes**, even if a client is not coherent with this change.

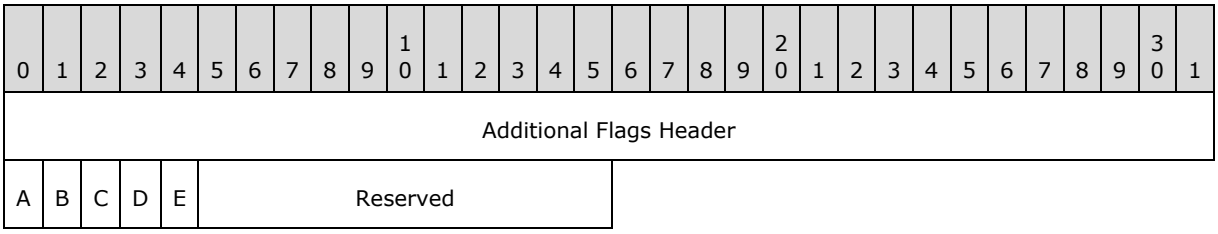
Additional Flags (6 bytes): An optional **Additional Flags** structure (section 2.2.2.1.4.1) with various flags that specify the desired behavior for the **Put Changes** request.

Lock Id (20 bytes): A **Lock Id** structure (section 2.2.2.1.4.2) that specifies the lock ID of the file on the server.

Client Knowledge (variable): An optional **knowledge** (section 2.2.1.13) that represents the knowledge of the client. It does not affect the **Put Changes** but allows the server to optimize the **Resultant Knowledge** in the **Put Changes Response.**

2.2.2.1.4.1 Additional Flags

Additional Flags has the following format:



Additional Flags Header (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies the start of an Additional Flags structure.

A – Return Applied Storage Index Id Entries (1 bit): A bit that specifies that the storage indexes that are applied to the storage as part of the **Put Changes** will be returned in a **Storage Index** specified in the **Put Changes Response** by the **Applied Storage Index Id** (section 2.2.3.1.3).

B – Return Data Elements Added (1 bit): A bit that specifies that the **Data Elements** that are added to the storage as part of this **Put Changes** will be return in a **Data Element Collection** specified in the **Put Changes Response** by the **Data Elements Added** collection (section 2.2.3.1.3).

C – Check for Id Reuse (1 bit): A bit that specifies that the server should attempt to check the **Put Changes Request** for the re-use of previously used Ids. This may occur when ID allocations are used and a client rollback occurs.

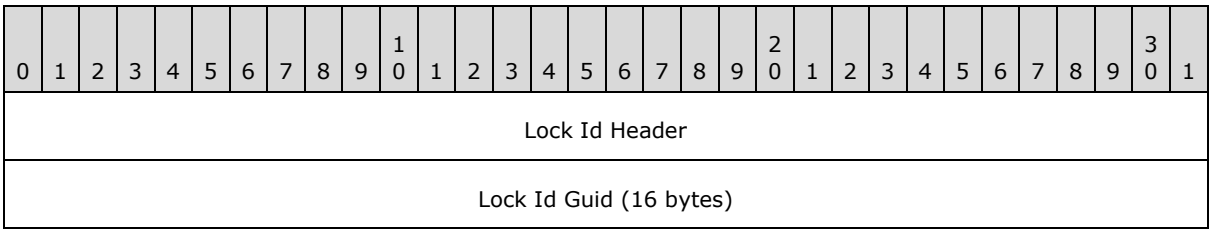
D – Coherency Check Only Applied Index Entries (1 bit): A bit that specifies that only the index entries that are actually applied as part of the change will be checked for coherency.

E – Full File Replace Put (1 bit): A bit that specifies that the **Put Changes** request should be treated as a full file save with no dependencies on any pre-existing state. This flag is not required for a full file save, but if specified, the server MAY bypass checks that would otherwise be unnecessary.

Reserved (11 bits): An 11-bit reserved field that MUST be set to zero, and MUST be ignored.

2.2.2.1.4.2 Lock Id

Lock Id has the following format:

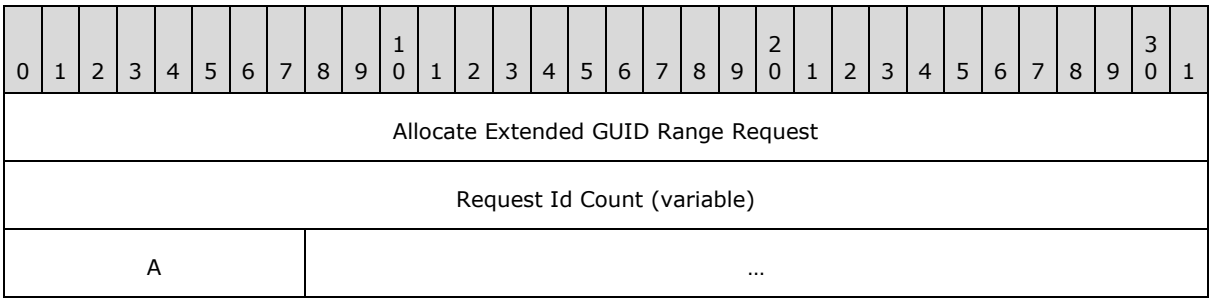


Lock Id Header (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a Lock Id start.

Lock Id Guid (16 bytes): A **GUID** that specifies the Lock Id of the lock on the file on the server. This is the same as the **BypassLockId** in [MS-FSSHTTP] section 2.3.3.1.

2.2.2.1.5 Allocate Extended GUID Range

The **Allocate Extended GUID Range sub-request** has the following format.



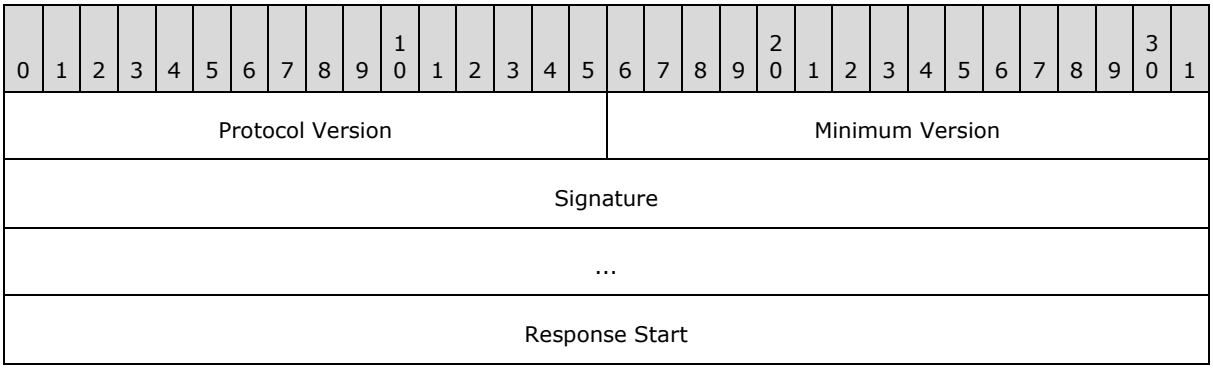
Allocate Extended GUID Range Request (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies an **Allocate ExtendedGuid Range** request.

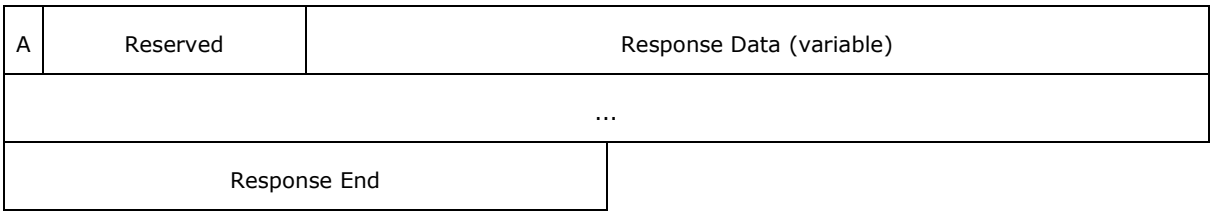
Request Id Count (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the number of **extended Guids** (section 2.2.1.7) to allocate.

A – Reserved (8 bits): An 8-bit reserved field that MUST be set to zero and MUST be ignored.

2.2.3 Response Message Syntax

A response message specifies the format used to contain **sub-responses** (section 2.2.3.1) matching **sub-requests** (section 2.2.2.1) in the following format.





Protocol Version (2bytes): An unsigned integer that specifies the protocol schema version number used in this request. This value MUST be 12.

Minimum Version (2 bytes): An unsigned integer that specifies the oldest version of the protocol schema that this schema is compatible with. This value MUST be 11.

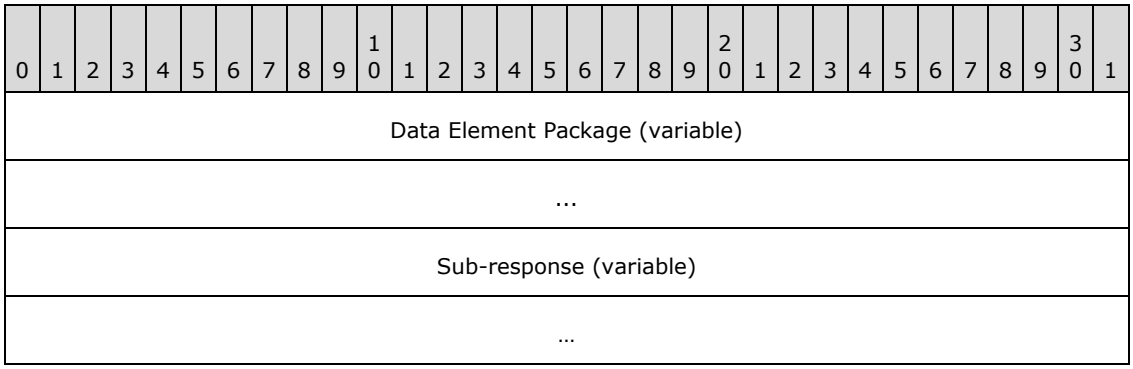
Signature (8 bytes): An unsigned integer that specifies a constant signature, to identify this as a response. This MUST be set to 0x9B069439F329CF9D.

Response Start (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a response start.

A - Status (1 bit): If set, a bit that specifies that the request has failed and a **response error** (section 2.2.3.2) MUST follow.

Reserved (7 bits): A 7-bit reserved field that MUST be set to zero and MUST be ignored.

Response Data (variable): A **response error** that specifies the error information if the request failed. If the request did not fail, the response data is specified by the following structure:



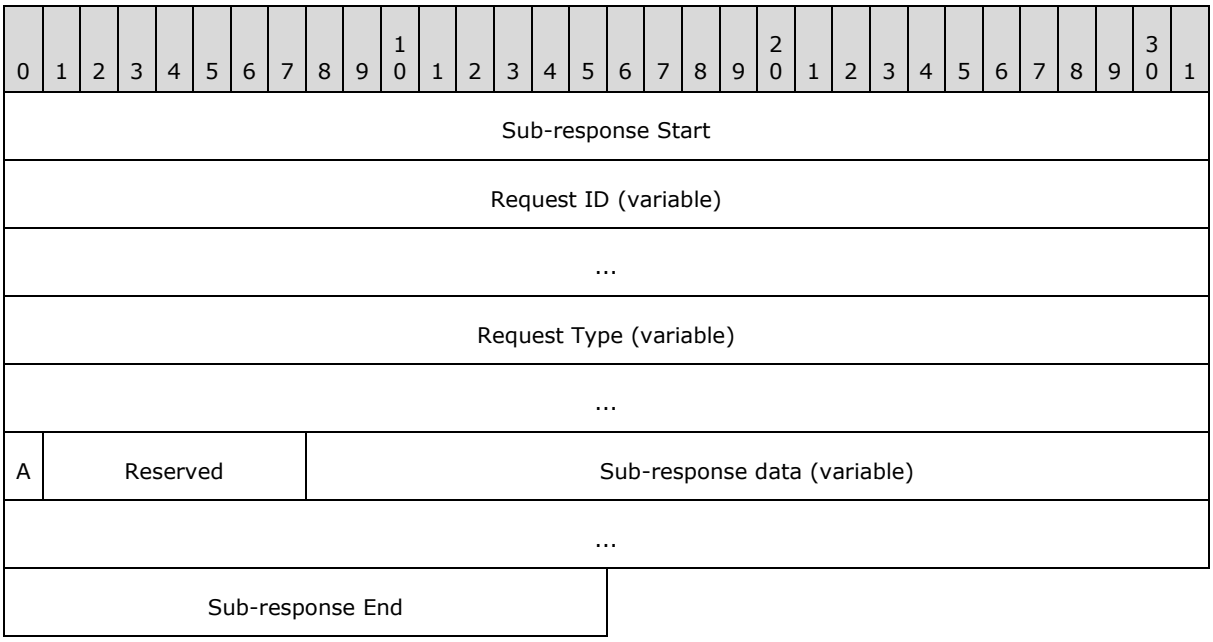
Data Element Package (variable): An optional **data element package** structure (section 2.2.1.12) that specifies data elements corresponding to the **sub-responses** (section 2.2.3.1). If the **sub-responses** do not reference data elements or no data elements are available for the **sub-response**, this structure MUST be ignored if present.

Sub-response (variable): Specifies an array of **sub-responses** corresponding to the **sub-requests** as specified in section 2.2.2.1**.**

Response End (2 bytes): A **16-bit stream object header** (section 2.2.1.5.4) that specifies a response end.

2.2.3.1 Sub-Responses

Specifies the **sub-responses** corresponding to each **sub-request** (section 2.2.2.1) in the following format.



Sub-response Start (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **sub-response** start.

Request ID (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the request number this **sub-response** is for.

Request Type (variable): A **compact unsigned 64-bit integer** that specifies the request type (section 2.2.1.6) matching the request.

A - Status (1 bit): If set, a bit that specifies the **sub-request** has failed. A **response error** (section 2.2.3.2) MUST follow.

Reserved (7 bits): A 7 bit reserved field that MUST be set to zero and MUST be ignored.

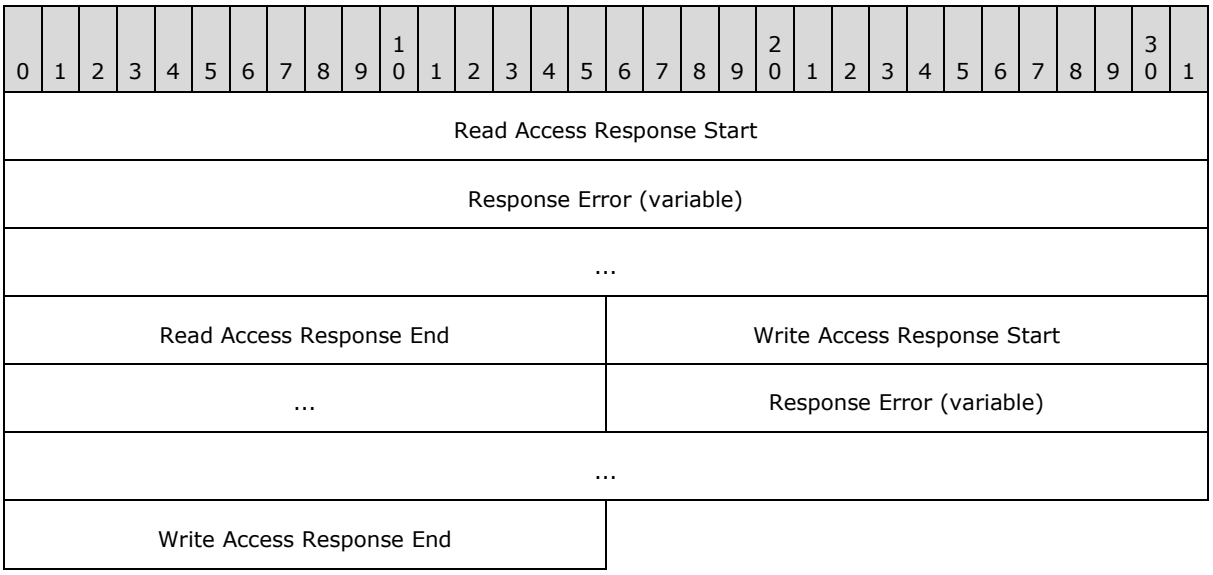
Sub-response data (variable): A **response error** that specifies the error information about failure of the **sub-request**, or depending on the **request type** (section 2.2.1.6), specifies additional data. See the following table for details:

Value	Meaning
1	Query access (section 2.2.3.1.1).
2	Query changes (section 2.2.3.1.2).
5	Put changes (section 2.2.3.1.3).

Sub-response End (2 bytes): A **16-bit stream object header** (section 2.2.1.5.4) that specifies a **sub-response** end.

2.2.3.1.1 Query Access

The **Query Access** specifies the access permissions requested for read/write to the file in the following format.



Read Access Response Start (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a read access response start.

Response Error (variable): A **response error** (section 2.2.3.2) that specifies read access permission. This error is received in response to any read request made by the client. If read operations will succeed, the **response error** will have an error type of HRESULT error, and the HRESULT error code will be zero.

Read Access Response End (2 bytes): A **16-bit stream object header** (section 2.2.1.5.4) that specifies a read access response end.

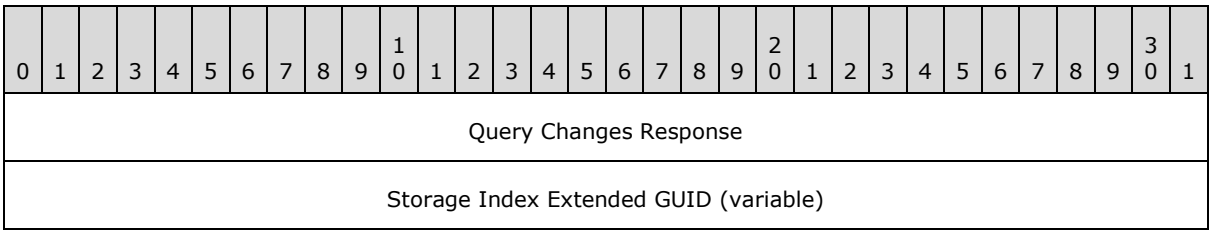
Write Access Response Start (4 bytes): A **32-bit stream object header** that specifies a write access response start.

Response Error (variable): A **response error** that specifies write access permission. This error is received in response to a **Put Changes** request made by the client. If the **Put Changes** operation will succeed, the **response error** will have an error type of HRESULT error, and the HRESULT error code will be zero.

Write Access Response End (2 bytes): A **16-bit stream object header** that specifies a write access response end.

2.2.3.1.2 Query Changes

The **Query Changes** returns the set of changes the server has for the data elements in the following format.



...		
P	Reserved	Knowledge (variable)
...		

Query Changes Response (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **Query Changes** response.

Storage Index Extended GUID (variable): An **extended GUID** (section 2.2.1.7) that specifies storage index.

P (1 bit): If set, a bit that specifies that the result is a partial result and not the full results.

Reserved (7 bits): A 7-bit reserved field that MUST be set to zero and MUST be ignored.

Knowledge (variable): A **knowledge** (section 2.2.1.13) that specifies the current state of the file on the server.

2.2.3.1.3 Put Changes

Specifies the resulting knowledge of a **Put Changes** request in the following format.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Put Changes Response																															
Applied Storage Index Id (variable)																															
...																															
Data Elements Added (variable)																															
...																															
Resultant Knowledge (variable)																															
...																															

Put Changes Response (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies a **Put Changes** response.

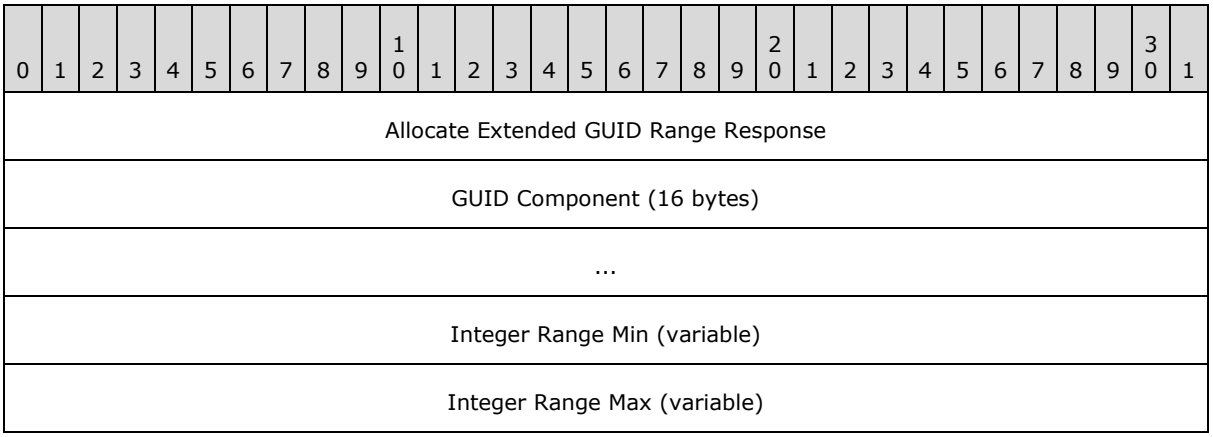
Applied Storage Index Id (variable): An **extended GUID** (section 2.2.1.7) that specifies the applied storage index ID.

Data Elements Added (variable): An **extended GUID array** (section 2.2.1.8) that specifies the data element identifiers of the added data elements.

Resultant Knowledge (variable): A **knowledge** (section 2.2.1.13) that specifies the current state of the file on the server after the changes is merged.

2.2.3.1.4 Allocate ExtendedGuid Range

The **Allocate ExtendedGUID Range sub-response** has the following format.



Allocate ExtendedGuid Range Response (4 bytes): A **stream object header** (section 2.2.1.5) that specifies an **allocate extendedGUID range** response.

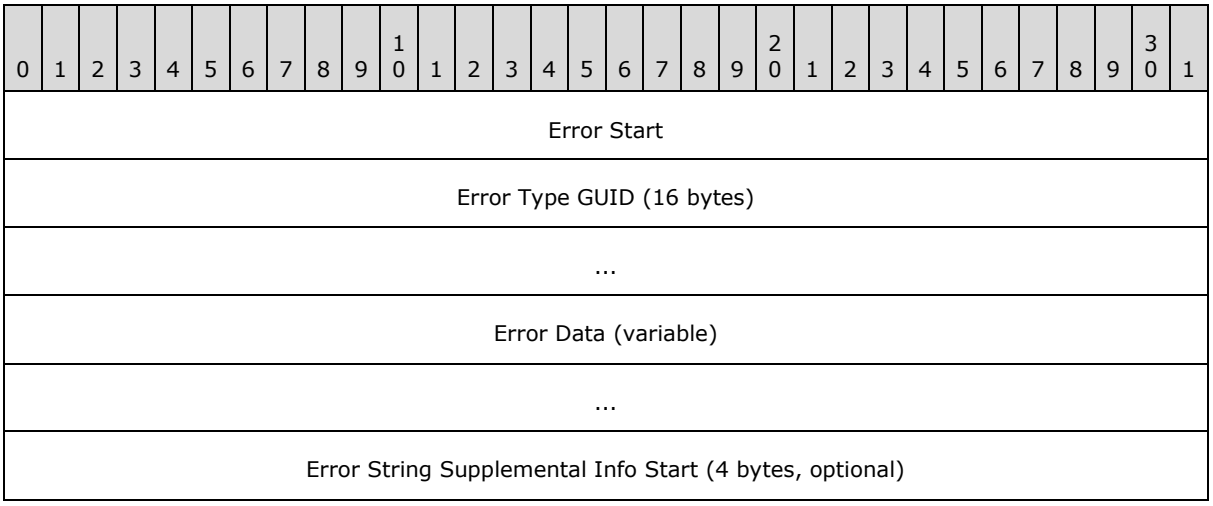
GUID Component (16 bytes): A GUID that specifies the GUID portion of the reserved **extended GUIDs** (section 2.2.1.7).

Integer Range Min (variable): A **compact unsigned 64-bit integer** (section 2.2.1.1) that specifies the first integer element in the range of **extended GUIDs**.

Integer Range Max (variable): A **compact unsigned 64-bit integer** with a minimum allowed value of 1000 and maximum allowed value of 100000 which specifies the last + 1 integer element in the range of **extended GUIDs**.

2.2.3.2 Response Error

The **Response Error** specifies a response, a sub-response error or status of a request in the following format.



Error String Supplemental Info (variable, optional)
...
Chained Error (variable)
...
Error End

Error Start (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies an error start.

Error Type GUID (16 bytes): A GUID that specifies the error type. The following table contains the possible values for the error type:

GUID (string representation)	Error type
{5A66A756-87CE-4290-A38B-C61C5BA05A67}	Cell error (section 2.2.3.2.1).
{7AFEAEBF-033D-4828-9C31-3977AFE58249}	Protocol error (section 2.2.3.2.2).
{32C39011-6E39-46C4-AB78-DB41929D679E}	Win32 error (section 2.2.3.2.3).
{8454C8F2-E401-405A-A198-A10B6991B56E}	HRESULT error (section 2.2.3.2.4).

Error Data (variable): A structure that specifies the error data based on the **Error Type GUID**.

Error String Supplemental Info Start (4 bytes, optional): Zero or one **32-bit stream object header** (section 2.2.1.5.2) that specifies an error string supplemental info start.

Error String Supplemental Info (variable): A **string item** (section 2.2.1.4) that specifies the supplemental information of the error string for the error string supplemental info start.

Chained Error (variable): An optional **response error** that specifies the chained error information.

Error End (2 bytes): A **16-bit stream object header** (section 2.2.1.5.4) that specifies an error end.

2.2.3.2.1 Cell Error

Specifies a **Cell error** in 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
Error Cell																															
Error Code																															

Error Cell (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies an error cell.

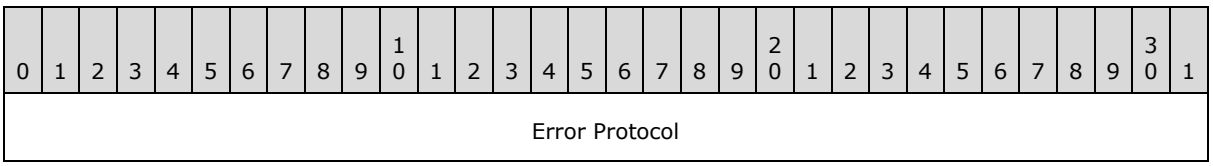
Error Code (4 bytes): An unsigned integer that specifies the error code. The following table contains the possible error codes:

Error Code	Error
1	Unknown error.
2	Invalid object.
3	Invalid partition.
4	Request not supported.
5	Storage read only.
6	Revision ID not found.
7	Bad token.
8	Request not finished.
9	Incompatible token.
11	Scoped cell storage.
12	Coherency failure.
13	Cell storage state de-serialization failure.
15	Incompatible protocol version.
16	Referenced data element not found.
18	Request stream schema error.
19	Response stream schema error.
20	Unknown request.
21	Storage failure.
22	Storage write only.
23	Invalid serialization.
24	Data element not found.
25	Invalid implementation.
26	Incompatible old storage.
27	Incompatible new storage.
28	Incorrect context for data element ID.
29	Object group duplicate objects.
31	Object reference not found in revision.

Error Code	Error
32	Merge cell storage state conflict.
33	Unknown **Query Changes** filter.
34	Unsupported **Query Changes** filter.
35	Unable to provide knowledge.
36	Data element missing ID.
37	Data element missing **serial number**.
38	Request argument invalid.
39	Partial changes not supported.
40	Store busy, retry later.
41	GUID identifier table not supported.
42	Data element cycle.
43	Fragment knowledge error.
44	Fragment size mismatch.
45	Fragments incomplete.
46	Fragment invalid.
47	Aborted after failed **Put Changes**.
79	Upgrade failed because there are no upgradeable contents.
106	Unable to allocate additional extended GUIDs.
108	Site is in read-only mode.
111	Multi-Request partition reached quota.
112	Extended GUID collision.
113	Upgrade failed because of insufficient permissions.
114	Upgrade failed because of server throttling.
115	Upgrade failed because the upgraded file is too large.

2.2.3.2.2 Protocol Error

Specifies a **Protocol error**. A protocol error is returned from the protocol server when a malformed or incompatible request is specified in the following format.



Error Code

Error Protocol (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies an error protocol.

Error Code (4 bytes): An unsigned integer that specifies the error code. The following table contains the possible error codes:

Error Code	Error
1	Unknown error.
50	Request format error, incomplete request.
61	Unknown internal error
108	Request format error, invalid request.
142	Request format error, stream object invalid
143	Request format error, stream object unexpected
144	Request format error, stream object compound nesting error
145	Request format error, invalid request.
All other values	Unspecified server error.

2.2.3.2.3 Win32 Error

A **Win32 error** code as specified in [MS-ERREF] section 2.2 in the following format.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Error Win32																															
Error Code																															

Error Win32 (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies an error win32.

Error Code (4 bytes): An unsigned integer that specifies the **Win32 error** code.

2.2.3.2.4 HRESULT Error

An **HRESULT error** code as specified in [MS-ERREF] section 2.1 in the following format.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Error HRESULT																															

Error Code

Error HRESULT (4 bytes): A **32-bit stream object header** (section 2.2.1.5.2) that specifies an **Error HRESULT**.

Error Code (4 bytes): An unsigned integer that specifies the **HRESULT error** code. Zero means that no error occurred.

3 Protocol Details

The Binary Requests for File Synchronization via SOAP Protocol operates between a protocol client (the requester) and a protocol server (the responder).

3.1 Server Details

3.1.1 Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

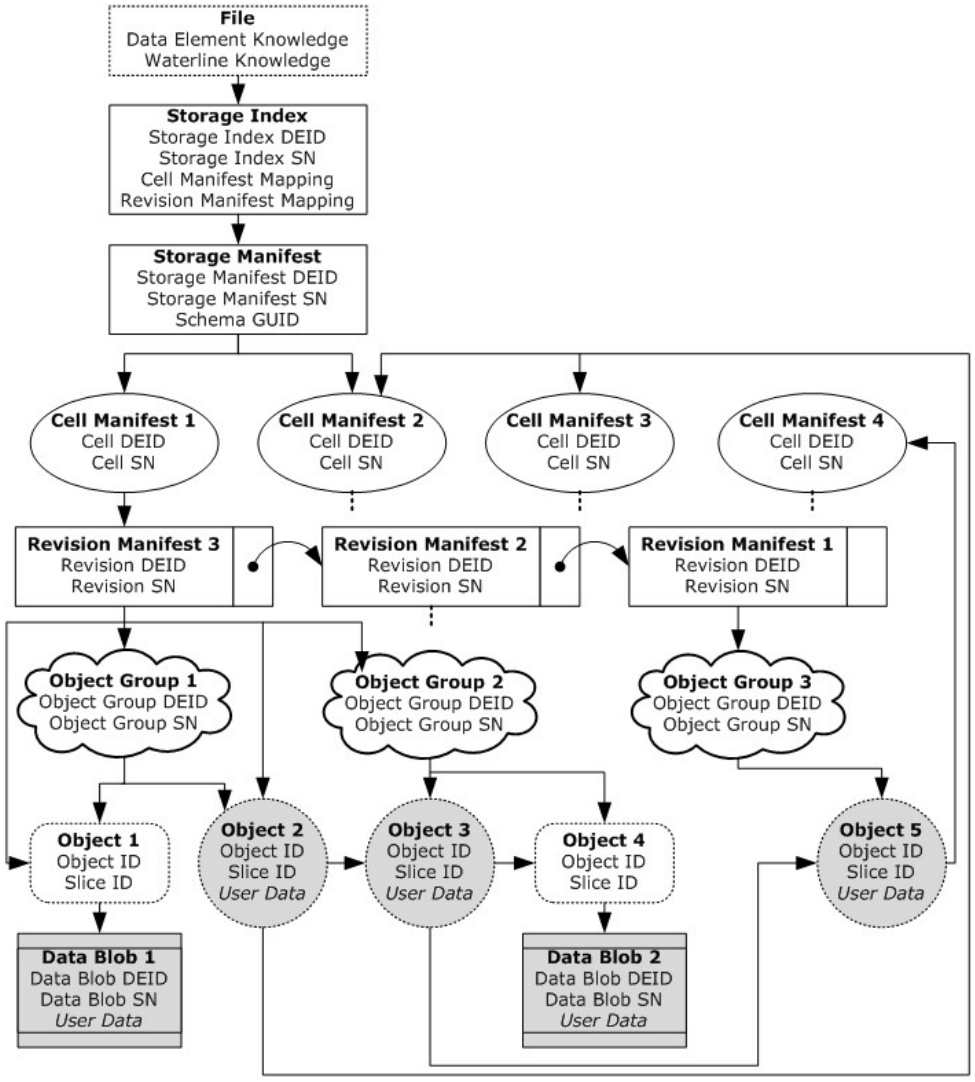


Figure 1: Abstract Data Model

The protocol server maintains a tree of several kinds of data nodes to represent the current state of the file. The nodes of the tree collectively store information about the data elements of this protocol. A data element in this context is a portion of a file, or data about a file, that can be synchronized as a unit. The types of nodes used in this data model include:

File: The root item of the tree of nodes that collectively represent the current state of a file available on the server. This node maintains the current knowledge information as specified in section 2.2.1.13, and also refers to the Storage Index node.

- **Data Element Knowledge:** The current **cell knowledge** (section 2.2.1.13.2) of the file.
- **Waterline Knowledge:** The current **waterline knowledge** (section 2.2.1.13.4) of the file.

Storage Index: The node of the tree that maintains data for the **storage index data element** (section 2.2.1.12.2), and which refers to the current **storage manifest data element** (section 2.2.1.12.3) of the file.

- **Storage Index DEID:** The data element identifier of the Storage Index.
- **Storage Index SN:** The current **serial number** (section 2.2.1.9) of the Storage Index.
- **Cell Manifest Mapping:** The map of **cell IDs** (section 2.2.1.10) to the Data Element identifiers of their Cell Manifests. When nodes refer to a **cell ID**, this mapping allows that reference to be resolved to the **cell manifest data element** (section 2.2.1.12.4) that contains the data for the Cell.
- **Revision Manifest Mapping:** The map of Revision identifiers to the Data Element identifiers of their Revision Manifests. When other nodes refer to a Revision identifier, this mapping allows that reference to be resolved to the **revision manifest data element** (section 2.2.1.12.5) that contains the data for the Revision.

Storage Manifest: The node of the tree that maintains data for the current **storage manifest data element** of the file.

- **Storage Manifest DEID:** The Data Element identifier of the Storage Manifest.
- **Storage Manifest SN:** The current **serial number** of the Storage Manifest.
- **Schema GUID:** The GUID that identifies the schema of the User Data and the organization of the nodes in the File. Users of this protocol would generally assign a unique **Schema GUID** for each different kind of document or file format, so that applications will know how to interpret the contents of the File.
- **Root Cell Set:** The ordered set of root **cell ID** references to the root Cells for this File. This is the set of Cells that are required directly by the File. The Cell Manifest Mapping of the Storage Index is used to resolve each **cell ID** reference into the Data Element identifier of the Cell Manifest for the Cell. It is possible, through references from Objects, for additional Cells to be used by a File even though they are not members of this Root Cell Set.

Cell Manifest: The **cell manifest data element** refers to a set of Revision data for the Cell. Users of this protocol would generally use Cells to represent major divisions of the File that have limited interdependency.

- **Cell DEID:** The immutable Data Element identifier of the Cell Manifest for this Cell.
- **Cell SN:** The current **serial number** of the Cell Manifest for this Cell.

- **Current Revision:** A Revision ID reference to the current Revision Manifest for this Cell. The Revision Manifest Mapping of the Storage Index is used to resolve each Revision identifier reference into the Data Element identifier of the Revision Manifest.

Revision Manifest: The **revision manifest data element** is part of a linked list which represents state information for a Cell. This node also refers to Object Groups and root Objects for the Revision. Users of this protocol would generally build a linked list of Revisions to represent the state of a Cell as a series of incremental changes.

- **Revision DEID:** The immutable Data Element identifier of the Revision Manifest for this Revision.
- **Revision SN:** The current **serial number** of the Revision Manifest for this Revision.
- **Root Object Set:** The set of root Object identifier references to the root Object nodes for this Revision. The data pertaining to each Object identifier is found in an Object Group Set of this same Revision Manifest, or in an Object Group Set of a prior Revision Manifest along the linked list. It is possible, through references from other Objects, for an Object to be used by a Revision even though it is not a member of this Root Object Set.
- **Object Group Set:** The set of Data Element identifier references to the Object Group nodes for this Revision.
- **Base Revision:** A Revision identifier reference to the prior Revision node in the linked list. The Revision Manifest Mapping of the Storage Index is used to resolve each Revision identifier reference into the Data Element identifier of the Revision Manifest of the prior Revision. If all of the data for an Object is unchanged, it is possible to rely on the data for that Object stored by a prior Revision somewhere along the linked list. This allows the protocol to be used for incremental updates to the File.

Object Group: A Data Element that identifies a group of Objects that can be referenced by Revisions or other Objects. Users of this protocol would generally group objects together when they tend to be modified together.

- **Object Group DEID:** The immutable Data Element identifier of this Object Group.
- **Object Group SN:** The current **serial number** of this Object Group.
- **Object Set:** The set of Object nodes included in this Object Group's Data Element. Object nodes do not represent separate Data Elements, but are included in their containing Object Groups. There are two types of Object nodes, those that store their User Data internally, and those that refer to a separate Data Element for their User Data.

Object with User Data stored internally: A node that is part of an Object Group's Object Set, and which contains User Data and reference information for an Object Partition of an Object. Users of this protocol would generally use this kind of Object to enable references to other Objects and Cells.

- **Object ID:** The **extended GUID** (section 2.2.1.7) by which other nodes reference this Object.
- **Object Partition ID:** An 8-bit unsigned integer that identifies this Object Partition of the Object. Users of this protocol would generally use Object Partitions to divide Objects into pieces that tend to be updated at different times.
- **User Data:** The stream of data for this Object Partition of this Object. Contents are opaque to this protocol. Users of this protocol would generally use this stream to store portions of the contents of their file format.

- **Object Reference Set:** The ordered set of Object identifier references to other Objects that this Object Partitions of this Object depends on. All of the Objects in this set MUST be in the same Cell as this Object. The data pertaining to each Object identifier is found in an Object Group Set of the Revision Manifest of the Cell, or in an Object Group Set of a prior Revision Manifest along the linked list.
- **Cell Reference Set:** The ordered set of **cell ID** references to other Cells that this Object Partitions of this Object depends on. The Cell Manifest Mapping of the Storage Index is used to resolve each cell identifier reference into the Data Element identifier of the Cell Manifest for the Cell.

Object with User Data stored externally: A node that is part of an Object Group's Object Set, and which references User Data for an Object Partition of an Object. Users of this protocol would generally use this kind of Object when the User Data is large and there is no need for references to other Objects or Cells.

- **Object ID:** The **extended GUID** by which other nodes reference this Object.
- **Object Partition ID:** An 8-bit unsigned integer that identifies this Object Partitions of the Object. Users of this protocol would generally use Object Partitions to divide Objects into pieces that tend to be updated at different times.
- **Data Blob Reference:** A Data Element identifier reference to the Data Blob node that contains the User Data for this Object Partitions of this Object.

Data Blob: A Data Element that contains User Data for an Object Partition of an Object:

- **Data Blob DEID:** The immutable Data Element identifier of this Data Blob.
- **Data Blob SN:** The current **serial number** of this Data Blob.
- **User Data:** The stream of data for an Object Partition of an Object. Contents are opaque to this protocol. Users of this protocol would generally use this stream to store portions of the contents of their file format.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Message Processing Events and Sequencing Rules

Each request received by the server contains a variable number of **sub-requests** (section 2.2.2.1). The server MUST reply to a well-formed request with a response, as specified in section 2.2.3, which includes a **sub-response** for each **sub-request**. The server MUST reply to requests that are poorly formed or have bad parameters with an error response, as specified in sections 2.2.3 and 2.2.3.2.

This protocol includes the **sub-requests** described in the following table.

Sub-request	Description
Query Access	Gets the read access and write access properties of the file.

Sub-request	Description
Query Changes	Gets a full or partial set of data elements for the current state of the file.
Put Changes	Client submits a set of data elements to update the state of the file on the server.

3.1.4.1 Query Access Sub-Request Processing

The **Query Access sub-request**, as specified in section 2.2.2.1.2, is used by a protocol client to determine if the File is expected to allow read access, and if the File is expected to allow write access, as determined by the underlying system. The server MUST reply back to the client with a **Query Access sub-response**, as specified in section 2.2.3.1.1.

3.1.4.2 Query Changes Sub-Request Processing

The **Query Changes sub-request**, as specified in section 2.2.2.1.3, is used by the protocol client to get a full or partial set of Data Elements for the current state of the File. The server MUST reply back to the client with a **Query Changes sub-response**, as specified in section 2.2.3.1.2.<5>

If **Request Data Element Hashes**, as specified in section 2.2.2, is 1 or **Request Data Element Hashes Instead of Data**, as specified in section 2.2.2, is 1 the server MUST return **Data Element Hashes**, as specified in section 2.2.2, in the schema specified by **Request Hashing Schema,** as specified in section 2.2.2, for **Object Groups Data Elements**, as specified in section 2.2.1.12.6, the protocol server has chosen to hash with the specified schema.

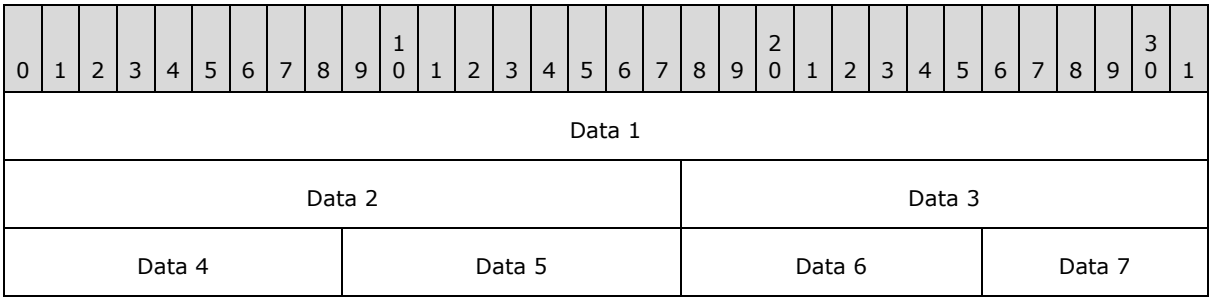
If **Request Data Element Hashes Instead of Data** is 1 the protocol server MUST return an **Object Group Object Excluded Data** for the **Object Declaration** of each **Object Group Data Element** that includes a **Data Element Hash**.

A **Data Element Hash** returned by the protocol server MUST be computed by hashing a byte stream consisting of an ordered concatenation of the **Object Data Data** binary items, as specified in section 2.2.1.12.6.4, in the **Object Group Data Element** being hashed. The concatenation order MUST be by sorted order from smallest to largest using the **Object Order Compare Method** as specified below.

Object Order Compare Method

The **Object Order Compare Method** utilizes the **Object Extended GUID**, as specified in section 2.2.1.12.6.1, and the **Object Partition ID**, as specified in section 2.2.1.12.6.1, to compare **Object** 'A' and **Object** 'B' for the purpose of creating the sorted smallest to largest concatenation order.

For the purposes of the **Object Order Compare Method** the 16 byte **GUID** portion of an **Object Extended GUID**, as specified in section 2.2.1.7, is specified as follows:



Data 8	Data 9	Data 10	Data 11

Data 1: An unsigned 32 bit integer specifying an opaque value.

Data 2: An unsigned 16 bit integer specifying an opaque value.

Data 3: An unsigned 16 bit integer specifying an opaque value

Data 4: An unsigned 8 bit integer specifying an opaque value.

Data 5: An unsigned 8 bit integer specifying an opaque value.

Data 6: An unsigned 8 bit integer specifying an opaque value.

Data 7: An unsigned 8 bit integer specifying an opaque value.

Data 8: An unsigned 8 bit integer specifying an opaque value.

Data 9: An unsigned 8 bit integer specifying an opaque value.

Data 10: An unsigned 8 bit integer specifying an opaque value.

Data 11: An unsigned 8 bit integer specifying an opaque value.

The table below specifies the compare method:

Step	Data to Compare	Compare Method	A < B	A = B	A > B
1.	**Object Extended GUID : Value**	▪ Unsigned 32 bit integer comparison	A is less than B. Stop.	Continue to step 2.	A is greater than B. Stop
2.	**Object Extended GUID: Data 1**	▪ Unsigned 32 bit integer comparison	A is less than B. Stop.	Continue to step 3.	A is greater than B. Stop
3.	**Object Extended GUID: Data 2**	Unsigned 16 bit integer comparison	A is less than B. Stop.	Continue to step 4.	A is greater than B. Stop
4.	**Object Extended GUID: Data 3**	Unsigned 16 bit integer comparison	A is less than B. Stop.	Continue to step 5.	A is greater than B. Stop
5.	**Object Extended GUID: Data 4**	Unsigned 8 bit integer comparison	A is less than B. Stop.	Continue to step 6.	A is greater than B. Stop
6.	**Object Extended GUID: Data 5**	Unsigned 8 bit integer comparison	A is less than B. Stop.	Continue to step 7.	A is greater than B. Stop
7.	**Object Extended GUID: Data 6**	Unsigned 8 bit integer comparison	A is less than B. Stop.	Continue to step 8.	A is greater than B. Stop
8.	**Object Extended GUID: Data 7**	Unsigned 8 bit integer comparison	A is less than B.	Continue to step 9.	A is greater than B. Stop

Step	Data to Compare	Compare Method	A < B	A = B	A > B
			Stop.		
9.	**Object Extended GUID: Data 8**	Unsigned 8 bit integer comparison	A is less than B. Stop.	Continue to step 10.	A is greater than B. Stop
10.	**Object Extended GUID: Data 9**	Unsigned 8 bit integer comparison	A is less than B. Stop.	Continue to step 11.	A is greater than B. Stop
11.	**Object Extended GUID: Data 10**	Unsigned 8 bit integer comparison	A is less than B. Stop.	Continue to step 12.	A is greater than B. Stop
12.	**Object Extended GUID: Data 11**	Unsigned 8 bit integer comparison	A is less than B. Stop.	Continue to step 13.	A is greater than B. Stop
13.	**Object Partition ID**	▪ Unsigned 64 bit integer comparison.	A is less than B.	Undefined.	A is greater than B.

3.1.4.3 Put Changes Sub-Request Processing

The **Put Changes sub-request**, as specified in section 2.2.2.1.4, is used by a protocol client to submit local changes in the contents of a File to the protocol server. The protocol server incorporates the submitted changes into the data model so that they will be available on subsequent calls regarding this File. The server MUST reply back to the client with a **Put Changes sub-response,** as specified in section 2.2.3.1.3.

3.1.4.4 Allocate ExtendedGuid Range Sub-Request Processing

The **Allocate Extended GUID Range sub-request**, as specified in section 2.2.2.1.5, is used by a protocol client to request a unique range of **ExtendedGUID** values.<6>. The server MUST reply back to the client with an **Allocate ExtendedGuid Range sub-response**, as specified in section 2.2.3.1.4.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

3.2 Client Details

3.2.1 Abstract Data Model

The protocol client maintains the same abstract data model as the protocol server.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Message Processing Events and Sequencing Rules

Requests from protocol clients result in responses from the protocol server. The protocol client MUST update the data model as required by **sub-responses** from the protocol server.

3.2.4.1 Query Access Sub-Response Processing

Protocol clients use the **Query Access sub-response** data, as specified in section 2.2.3.1.1, to indicate to the user when the File is in a read-only state.

3.2.4.2 Query Changes Sub-Response Processing

Protocol clients MUST update their data model to incorporate the new Data Elements present in the **Query Changes sub-response**, as specified in section 2.2.3.1.2.<7>

3.2.4.2.1 Query Changes Request Processing When Data Element Hashes and Data Are Returned

If **Request Data Element Hashes**, as specified in section 2.2.2, is 1 and the protocol client chooses to inject **Object Group Data Elements** which have the optional **Data Element Hash** specified into the local cache, the protocol client MUST inject the data using the **Data Element Hash** and a byte stream consisting of an ordered concatenation of the **Object Data Data** binary items, as specified in section 2.2.1.12.6.4. The concatenation order MUST be by sorted order from smallest to largest using the **Object Order Compare Method** as specified section 3.1.4.2.

3.2.4.2.2 Query Changes Request Processing When Data Element Hashes Are Returned in place of Data

If **Request Data Element Hashes Instead of Data**, as specified in section 2.2.2, is 1, the protocol client MUST retrieve the excluded data from the protocol client local cache to successfully complete the sub-request. If the protocol client is unable to retrieve all excluded data from the protocol client local cache via hash lookup, the protocol client MUST retry the **Query Changes Sub-Request** with **Request Data Element Hashes Instead of Data** set to 0.

The protocol client MUST populate the excluded Object Data in an **Object Group Data Element** by sequentially reading the data for each object from the byte stream returned by the protocol client local cache lookup, in the correct object order. The amount of data to be read for each object is specified by **Object Data Data Size** as specified in section 2.2.1.12.6.4. The object order MUST be by sorted order from smallest to largest using the **Object Order Compare Method** as specified section 3.1.4.2.

3.2.4.3 Put Changes Sub-Response Processing

Protocol clients MUST update their data model to incorporate any new **serial numbers** (section 2.2.1.9) for Data Elements included in the **Put Changes sub-response** data, as specified in section

2.2.3.1.3. Protocol clients would generally track the difference between the protocol server's **knowledge** (section 2.2.1.13) and their own local set of known **serial numbers**.

3.2.4.4 Allocate ExtendedGuid Range Sub-Response Processing

The client requests this range of unique **extended GUIDs** (section 2.2.1.7) from the server. If the client uses these **ExtendedGuids** in subsequent **Put Changes sub-requests** (section 2.2.2.1.4), a server compliant with this specification may be able to store data more efficiently than one operating with client allocated **extended GUIDs**.

3.2.5 Timer Events

None.

3.2.6 Other Local Events

None.

4 Protocol Examples

4.1 Query Changes Request

Considering a client that needs to send a request to **Query Changes** (section 2.2.2.1.3), it would create a request as follows:

```
0x00000000: 0C 00 0B 00 9C CF 29 F3 39 94 06 9B 06 02 00 00
0x00000010: EE 02 00 00 AA 02 20 00 7E B8 31 E7 45 DD AA 44
0x00000020: AB 80 0C 75 FB D1 53 0E 7A 02 08 00 C4 27 A1 0F
0x00000030: 77 01 16 02 06 00 03 05 00 8A 02 02 00 00 DA 02
0x00000040: 06 00 03 00 00 CA 02 08 00 08 00 80 03 84 00 41
0x00000050: 0B 01 AC 02 00 55 03 01
```

<table>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>20</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>30</th><th>1</th></tr>
<tr><td colspan="16">Protocol Version</td><td colspan="16">Minimum Version</td></tr>
<tr><td colspan="32">Signature</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">Cell Request Start</td></tr>
<tr><td colspan="32">User Agent Start</td></tr>
<tr><td colspan="32">User Agent GUID</td></tr>
<tr><td colspan="32">GUID</td></tr>
<tr><td colspan="32">...</td></tr>
<tr><td colspan="32">User Agent Version</td></tr>
<tr><td colspan="32">Version</td></tr>
<tr><td colspan="16">User Agent End</td><td colspan="16">Sub-request Start</td></tr>
<tr><td colspan="16">...</td><td colspan="8">Request ID</td><td colspan="8">Request Type</td></tr>
<tr><td colspan="8">Priority</td><td colspan="24">Query Changes</td></tr>
<tr><td colspan="8">...</td><td colspan="2">A</td><td colspan="2">B</td><td colspan="4">C</td><td colspan="16">D</td></tr>
<tr><td colspan="16">...</td><td colspan="8">E</td><td colspan="8">Cell ID</td></tr>
<tr><td colspan="8">...</td><td colspan="24">Query Changes Data Constraints</td></tr>
</table>

...	Maximum Data Elements	
...	Knowledge Start	Knowledge End
Sub-request End	Data Element Package Start	
Reserved	E	Cell Request End

Protocol Version (2 bytes): 0x000C specifies the protocol version of this request.

Minimum Version (2 bytes): 0x000B specifies the minimum version of the protocol schema with which this request is compatible.

Signature: 0x9B069439F329CF9C specifies the signature of this request.

Cell Request Start (4 bytes): 0x00000206 specifies a **32-bit stream object header** (section 2.2.1.5.2) for a cell request start. Decoded, this has a type of 0x40, length 0, and is compound.

User Agent Start (4 bytes): 0x000002EE specifies a **32-bit stream object header** for user agent start. Decoded, this has a type of 0x5D, length 0, and is compound.

User Agent GUID (4 bytes): 0x002002AA specifies a **32 bit stream object header** for a user agent GUID. Decoded, this has a type of 0x55, length 16.

GUID (16 bytes): {"E731B87E-DD45-44AA-AB80-0C75FBD1530E"} is the GUID of the user agent.

User Agent Version (4 bytes): 0x0008027A specifies a **32-bit stream object header** for user agent version. Decoded, this has a type of 0x2F, length 4.

Version (4 bytes): 0x2EE127B4 specifies the version of the client.

User Agent End (2 bytes): 0x0177 specifies a **16-bit stream object header** for user agent end.

Sub-request Start (4 bytes): 0x00060216 specifies a **32-bit stream object header** for a **sub-request** (section 2.2.2.1) start. Decoded, this has a type of 0x42, length 3.

Request Id (1 byte): 0x03 specifies the request number as a **compact unsigned 64-bit integer** (section 2.2.1.1) for this request. Decoded, this represents a value of 0x1.

Request Type (1 byte): 0x05 specifies the **request type** (section 2.2.1.6) as a **compact unsigned 64-bit integer**. Decoded, this represents a value of 0x02.

Priority (1 byte): 0x00 specifies the priority of this **sub-request** as a **compact unsigned 64-bit integer**.

Query Changes (4 bytes): 0x0002028A specifies a **32-bit stream object header** for **Query Changes** request (section 2.2.2.1.3). Decoded, this has a type of 0x51, length 1.

A - Reserved (1 bit): zero specifies a reserved bit.

B - Allow Fragments (1 bit): zero specifies that fragments are not allowed.

C - Reserved (6 bits): zero specified reserved bits.

D - Query Changes Request Arguments (4 bytes): 0x000602DA specifies a **32-bit stream object header** for **Query Changes** request arguments. Decoded, this has a type of 0x5b, length 3.

E - Include Storage Manifest/Cell Changes/ Reserved (1 byte): 0x03 specifies that the storage manifest and cell changes are to be included.

Cell ID (2 bytes): 0x0000 specifies the **cell ID** (section 2.2.1.10) that **Query Changes** are scoped to. Decoded, this represents two **null extended GUIDS** (section 2.2.1.7.1), so no scoping restriction is specified.

Query Changes Data Constraints (4 bytes): 0x000802CA specifies a **32-bit stream object header** for **Query Changes** data constraints. Decoded, this has a type of 0x59, length 4.

Maximum Data Elements (4 bytes): 0x03800008 specifies the maximum data elements to return as a **compact unsigned 64-bit integer**. Decoded, this has a value of 3670016.

Knowledge Start (2 bytes): 0x0084 specifies the **16-bit stream object header** for a **knowledge** (section 2.2.1.13) start. This has a type of 0x10, length 0.

Knowledge End (1 byte): 0x41 specifies the **8-bit stream object header** (section 2.2.1.5.3) for a **knowledge** end.

Sub-Request End (2 bytes): 0x010B specifies the **16-bit stream object header** for **sub-request** end. Decoded, this has a type of 0x21.

Data Element Package Start (2 bytes): 0x02AC specifies the **16-bit stream object header** for a data element package start. Decoded, this has a type of 0x15, length 1, and is compound.

Reserved (1 byte): 0x00 specifies a reserved byte.

E - Data Element Package End (1 byte): 0x55 specifies the **8-bit stream object header** for the data element package end. This stream object was started in the **Request Header** section (section 4.3.1).

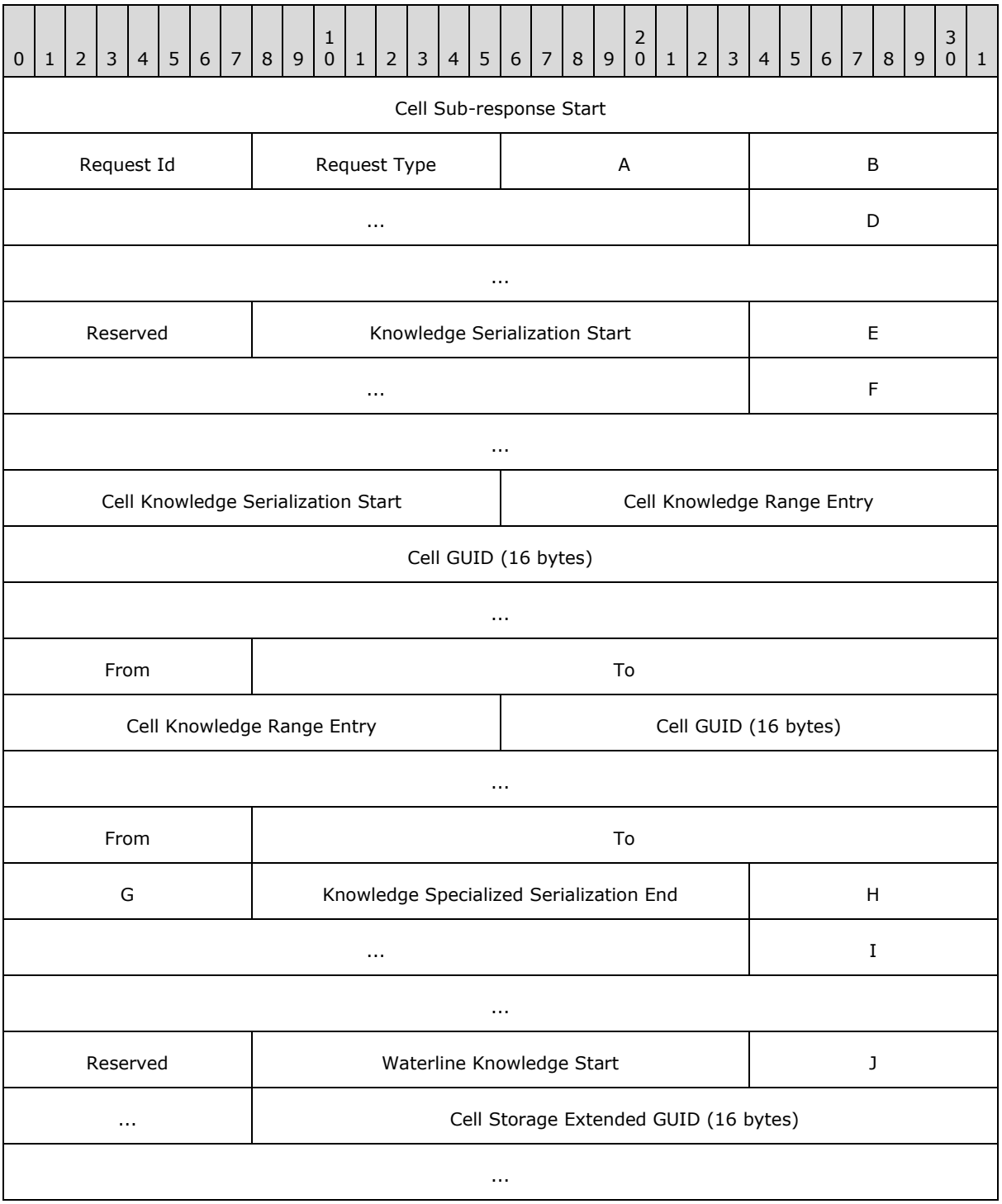
Cell Request End (2 bytes): 0x0103 specifies the **16-bit stream object header** for cell request end. This stream object was started in the **Request Header** section.

4.2 Query Changes Response

This section provides an example of a **Query Changes** response sub-request (section 2.2.3.1.2).

```
0x00000000:   0E 02 06 00 03 05 00 FA   02 24 00 0C FD 98 0D A0
0x00000010:   FD 40 99 4D 93 0A 63 22   D7 68 91 36 00 84 00 26
0x00000020:   02 20 00 F6 35 7A 32 61   07 14 44 96 86 51 E9 00
0x00000030:   66 7A 4D A4 00 78 28 80   93 0A E2 55 FD A5 BC 90
0x00000040:   37 45 1C 9D 86 E9 49 00   1C F9 08 78 28 7F 6C F5
0x00000050:   1D AA 02 5A 43 90 37 45   1C 9D 86 E9 49 00 FC F8
0x00000060:   08 51 13 01 26 02 20 00   0E E9 76 3A 32 80 0C 4D
0x00000070:   B9 DD F3 C6 50 29 43 3E   4C 01 20 2A 0C 7F 6C F5
0x00000080:   1D AA 02 5A 43 90 37 45   1C 9D 86 E9 49 FC F8 08
0x00000090:   00 A5 13 01 41 07 01 8B   01 0D 0A 33 39 0D 0A 0D
0x000000A0:   0A 2D 2D 75 75 69 64 3A   66 31 65 62 62 66 35 33
0x000000B0:   2D 65 62 39 39 2D 34 36   62 64 2D 61 63 63 30 2D
```

```
0x000000C0:   34 32 65 35 65 62 61 32   36 35 30 35 2B 69 64 3D
0x000000D0:   36 30 33 38 2D 2D 0D 0A   0D 0A 30 0D 0A 0D 0A
```



Waterline		
Reserved	K	Knowledge Specialized Serialization End
L	Cell Sub response End	

Cell Sub-response Start (4 bytes): 0x0E020600 specifies the **32-bit stream object header** (section 2.2.1.5.2) for a cell sub-response start. Decoded, this has a type of 0x041, length 3, and is compound.

Request Id (1 byte): 0x03 specifies the request number as a **compact unsigned 64-bit integer** (section 2.2.1.1) for this request. Decoded, this represents a value of 0x1.

Request Type (1 byte): 0x05 specifies the **request type** (section 2.2.1.6) as a **compact unsigned 64-bit integer**. Decoded, this represents a value of 0x2.

A - Status / Reserved (1 byte): 0x00 represents the status bit and reserved field of the cell response (section 2.2.3).

B - Query Changes Response (4 bytes): 0xFA022400 specifies the **32-bit stream object header** for a **Query Changes** response. Decoded, this has a type of 0x05F, length 18.

D - Storage Index Extended GUID (16 bytes): {"A00D98FD-40FD-4D99-930A-6322D7689136"} 0x1 specifies the **extended GUID** (section 2.2.1.7) of the cell storage this is the waterline of. Decoded, from 0x0C FD 98 0D A0 FD 40 99 4D 93 0A 63 22 D7 68 91 36.

Reserved (1 byte): 0x00 specifies a reserved byte.

Knowledge Serialization Start (2 bytes): 0x8400 specifies the **16-bit stream object header** (section 2.2.1.5.1) for a **knowledge** (section 2.2.1.13) serialization start. Decoded, this has a type of 0x10, length 0, and is compound.

E - Knowledge Specialized Serialization Start (4 bytes): 0x26022000 specifies the **32-bit stream object header** for **knowledge** specialized serialization start. Decoded, this has a type of 0x044, length 16, and is compound.

F - Cell Knowledge GUID (16 bytes): 0x F6357A32 6107 1444 9686 51E900667A4D specifies a GUID that when decoded has the string representation of {"327A35F6-0761-4414-9686-51E900667A4D"}, which indicates that this **knowledge** specialized serialization is a **cell knowledge** (section 2.2.1.13.2) serialization.

Cell Knowledge Serialization Start (2 bytes): 0xA400 specifies the **16-bit stream object header** for a **cell knowledge** (section 2.2.1.13.2) serialization start. Decoded, this represents a type of 0x14, length zero, and is compound.

Cell Knowledge Range Entry (2 bytes): 0x7828 specifies the **16-bit stream object header** for a **cell knowledge range** (section 2.2.1.13.2.1) entry. Decoded, this represents a type of 0x0F, length 20.

Cell GUID (16 bytes): 0x80930AE2 55FD A5BC 9037 451C9D86E949 specifies the GUID of the cell storage this **knowledge** is about. Decoded, this has the string representation {"E20A9380-FD55-BCA5-9037-451C9D86E949"}.

From (1 byte): 0x00 specifies the beginning of the cell range as a **compact unsigned 64-bit integer**. Decoded, this represents a value of zero.

To (3 bytes): 0x1CF908 specifies the end of the cell range as a **compact unsigned 64-bit integer**. Decoded, this represents a value of 73507.

Cell Knowledge Range Entry (2 bytes): 0x7828 specifies the **16-bit stream object header** for a **cell knowledge range** entry. Decoded, this represents a type of 0x0F, length 20.

Cell GUID (16 bytes): 0x7F6CF51D AA02 5A43 9037 451C9D86E949 specifies the GUID of the cell storage this knowledge is about. Decoded, this has the string representation {"1DF56C7F-02AA-435A-9037-451C9D86E949"}.

From (1 byte): 0x00 specifies the beginning of the cell range as a **compact unsigned 64-bit integer**. Decoded, this represents a value of zero.

To (3 bytes): 0xFCF808 specifies the end of the cell range as a **compact unsigned 64-bit integer**. Decoded, this represents a value of 73503.

G - Cell Knowledge Serialization End (1 byte): 0x51 specifies an **8-bit stream object header** (section 2.2.1.5.3) for a **cell knowledge** header end. Decoded, this has a type of 0x14.

Knowledge Specialized Serialization End (2 bytes): 0x1301 specifies a **16-bit stream object header** for **knowledge** specialized serialization end. Decoded, this has a type of 0x044.

H - Knowledge Specialized Serialization Start (4 bytes): 0x26022000 specifies a **32-bit stream object header** for a **knowledge** specialized serialization start. Decoded, this has a type of 0x044, length of 16, and is compound.

I - Waterline Knowledge GUID (16 bytes): 0x0EE9763A32800C4DB9DDF3C65029433E when decoded has string representation {"3A76E90E-8032-4D0C-B9DD-F3C65029433E"} which indicates that this **knowledge** specialized serialization is a **waterline knowledge** (section 2.2.1.13.4) serialization.

Reserved (1 byte): 0x00 specifies a reserved byte.

Waterline Knowledge Start (2 bytes): 0x4C01 specifies a **16-bit stream object header** for a **waterline knowledge** (section 2.2.1.13.4) start. Decoded, this has type 0x29, length 0, and is compound.

J - Waterline Knowledge Entry (2 bytes): 0x202A specifies a **16-bit stream object header** for a **waterline knowledge** entry. Decoded, this has type 0x04, length 21.

Cell Storage Extended GUID (16 bytes): {"1DF56C7F-02AA-435A-9037-451C9D86E949"} 0x1 specifies the **extended GUID** of the cell storage this is the waterline of. Decoded, from 0x0C 7F 6C F5 1D AA 02 5A 43 90 37 45 1C 9D 86 E9 49.

Waterline (4 bytes): 0xFCF808 specifies the waterline as a **compact unsigned 64-bit integer**. Decoded, this represents a value of 75503.

Reserved (1 byte): 0x00 specifies a reserved byte.

K - Waterline Knowledge End (1 byte): 0xA5 specifies an **8-bit stream object header** for **waterline knowledge** end. Decoded, this has a type of 0x29.

Knowledge Specialized Serialization End (2 bytes): 0x1301 specifies a **16-bit stream object header** for a **knowledge** specialized serialization end. Decoded, this has a type of 0x044.

L - Knowledge Serialization End (1 byte): 0x41 specifies an **8-bit stream object header** for **knowledge** serialization end. Decoded, this has a type of 0x10.

Cell Sub response End (2 bytes): 0x0701 specifies a **16-bit stream object header** (section 2.2.1.5.4) for cell sub-response end. Decoded, this has a value of 0x041.

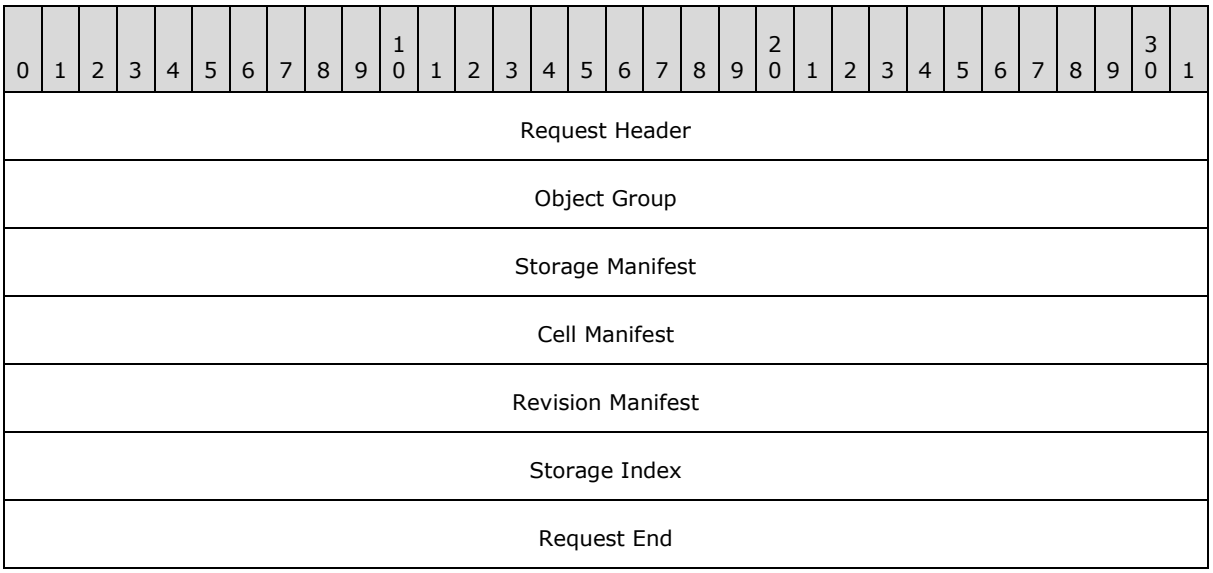
4.3 Put Changes Request

This section provides an example of a **Put Changes** request (section 2.2.2.1.4) saving a document through the protocol.

```
0x00000000: 0C 00 0B 00 9C CF 29 F3 39 94 06 9B 06 02 00 00
0x00000010: EE 02 00 00 AA 02 20 00 7E B8 31 E7 45 DD AA 44
0x00000020: AB 80 0C 75 FB D1 53 0E 7A 02 08 00 B4 27 E1 2E
0x00000030: 77 01 16 02 06 00 03 0B 00 D2 02 26 00 0C 8E 2E
0x00000040: 2E 05 D1 C0 86 48 9C 51 29 D6 61 71 4F 67 00 48
0x00000050: 0B 01 AC 02 00 0C 56 0C 8E FC 0B 2C 04 9B 61 4C
0x00000060: AB 49 48 45 E6 03 EC A0 80 47 AF 30 54 71 6E 9B
0x00000070: 40 98 06 70 7E 81 8D C1 02 01 00 00 00 00 00 00
0x00000080: 00 0B EC 00 C0 32 80 13 38 0C DE AF 7C 55 4E 95
0x00000090: 0E 65 7A D3 A3 FA 63 01 00 00 11 03 21 2F 00 75
0x000000A0: F4 00 B2 00 EC 03
                    502 Binary Bytes
0x00000290:                                        79 05
               Additional Object Groups


0x00006600:                                              0C 56
0x00006610: 0C 99 FA 30 D7 2C 12 88 42 B7 22 0A 12 5C FD A7
0x00006620: E5 80 47 AF 30 54 71 6E 9B 40 98 06 70 7E 81 8D
0x00006630: C1 02 32 00 00 00 00 00 00 00 05 60 20 94 33 B9
0x00006640: 0E 1D 57 E9 41 AA D3 88 0D 92 D3 19 55 38 66 14
0x00006650: B9 FA DE 84 A3 AA 0D 4A A3 A8 52 0C 77 AC 70 73
0x00006660: 0C B9 FA DE 84 A3 AA 0D 4A A3 A8 52 0C 77 AC 70
0x00006670: 73 0C 65 46 2A 6F C8 42 C7 46 BA B4 E2 8F DC E1
0x00006680: E3 2B 05 0C 58 60 0C 8E FC 0B 2C 04 9B 61 4C AB
0x00006690: 49 48 45 E6 03 EC A0 80 47 AF 30 54 71 6E 9B 40
0x000066A0: 98 06 70 7E 81 8D C1 02 33 00 00 00 00 00 00 00
0x000066B0: 07 58 22 0C 3A FE 28 71 BE DC 01 43 BD 84 71 6C
0x000066C0: 45 6C 80 8A 05 0C 56 0C 05 A9 D1 DF 9C 9B 2E 42
0x000066D0: B2 59 81 7A F3 51 14 54 80 47 AF 30 54 71 6E 9B
0x000066E0: 40 98 06 70 7E 81 8D C1 02 34 00 00 00 00 00 00
0x000066F0: 00 09 D0 24 0C 3A FE 28 71 BE DC 01 43 BD 84 71
0x00006700: 6C 45 6C 80 8A 00 50 4C 14 B9 FA DE 84 A3 AA 0D
0x00006710: 4A A3 A8 52 0C 77 AC 70 73 80 13 38 0C DE AF 7C
0x00006720: 55 4E 95 0E 65 7A D3 A3 FA 63 01 00 00 11 C8 22
0x00006730: 0C 8E FC 0B 2C 04 9B 61 4C AB 49 48 45 E6 03 EC
0x00006740: A0  . . .
       More Revision Manifest Object Group references
0x00006AB0:                                        0C 56 0C 8E
0x00006AC0: 2E 2E 05 D1 C0 86 48 9C 51 29 D6 61 71 4F 67 80
0x00006AD0: 0A 4E D0 67 25 4F E5 43 91 48 B7 28 D3 AB 89 77
0x00006AE0: 01 00 00 00 00 00 00 00 03 88 54 0C 99 FA 30 D7
0x00006AF0: 2C 12 88 42 B7 22 0A 12 5C FD A7 E5 80 B8 50 CF
0x00006B00: AB 8E 91 64 BF 98 06 70 7E 81 8D C1 02 3E 00 00
0x00006B10: 00 00 00 00 00 70 9A 0C B9 FA DE 84 A3 AA 0D 4A
0x00006B20: A3 A8 52 0C 77 AC 70 73 0C 65 46 2A 6F C8 42 C7
0x00006B30: 46 BA B4 E2 8F DC E1 E3 2B 60 0C 8E FC 0B 2C 04
```

```
0x00006B40: 9B 61 4C AB 49 48 45 E6 03 EC A0 80 B8 50 CF AB
0x00006B50: 8E 91 64 BF 98 06 70 7E 81 8D C1 02 40 00 00 00
0x00006B60: 00 00 00 00 68 76 0C 3A FE 28 71 BE DC 01 43 BD
0x00006B70: 84 71 6C 45 6C 80 8A 0C 05 A9 D1 DF 9C 9B 2E 42
0x00006B80: B2 59 81 7A F3 51 14 54 80 B8 50 CF AB 8E 91 64
0x00006B90: BF 98 06 70 7E 81 8D C1 02 3F 00 00 00 00 00 00
0x00006BA0: 00 05 55 03 01
```



Request Header (4 bytes): Contents of the **Request Header** (section 4.3.1).

Object Group (4 bytes): Contents of an **Object Group** (section 4.3.2).

Storage Manifest (4 bytes): Contents of a **Storage Manifest** (section 4.3.3).

Cell Manifest (4 bytes): Contents of **Cell Manifest** (section 4.3.4).

Revision Manifest (4 bytes): Contents of **Revision Manifest** (section 4.3.5).

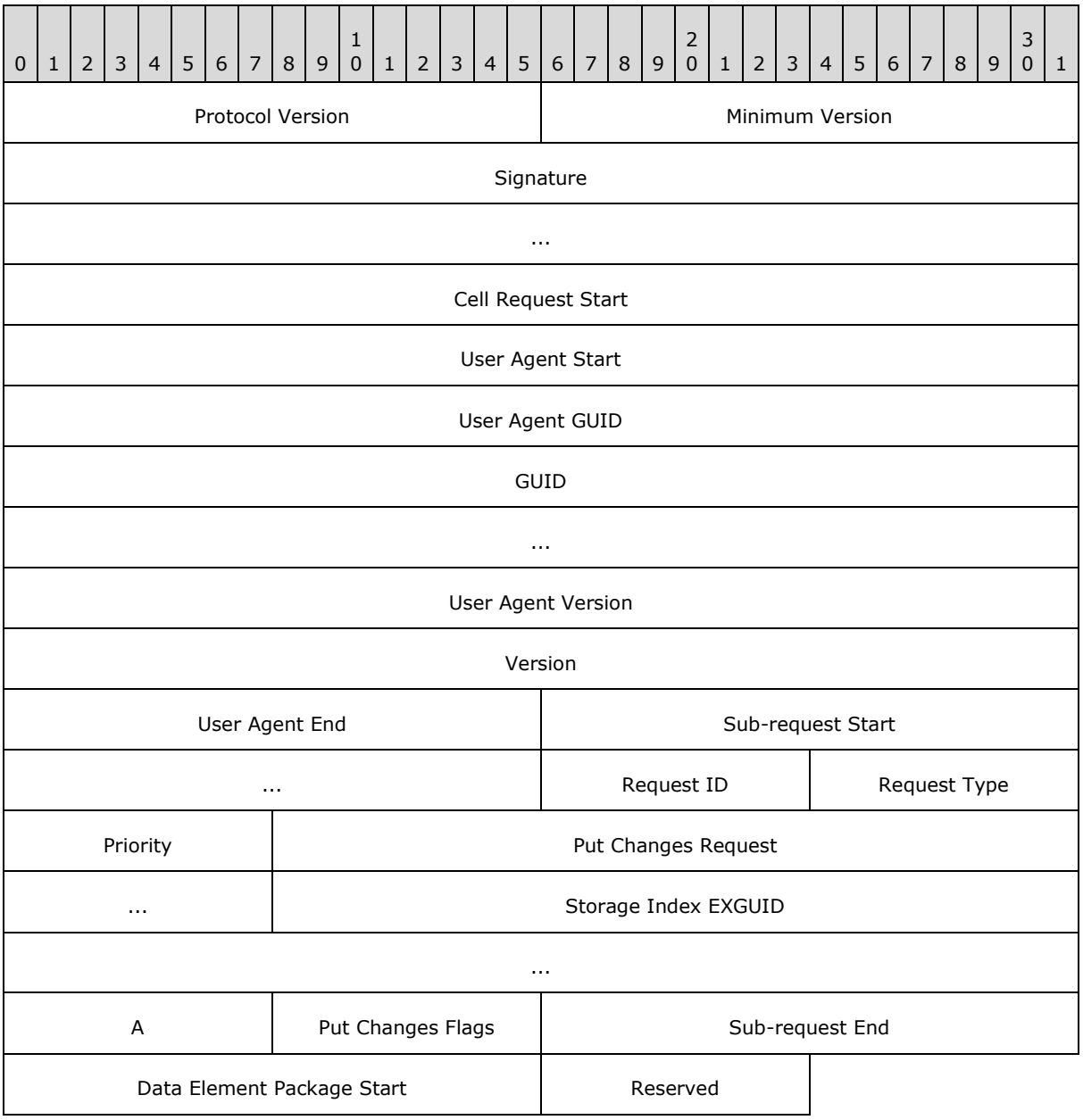
Storage Index (4 bytes): Contents of **Storage Index** (section 4.3.6).

Request End (4 bytes): Contents of **Request End** (section 4.3.7).

4.3.1 Request Header

This is the **Request Header** that is part of the **Put Changes Request** example (section 4.3).

```
Header:
0x00000000: 0C 00 0B 00 9C CF 29 F3 39 94 06 9B 06 02 00 00
0x00000010: EE 02 00 00 AA 02 20 00 7E B8 31 E7 45 DD AA 44
0x00000020: AB 80 0C 75 FB D1 53 0E 7A 02 08 00 B4 27 E1 2E
0x00000030: 77 01 16 02 06 00 03 0B 00 D2 02 26 00 0C 8E 2E
0x00000040: 2E 05 D1 C0 86 48 9C 51 29 D6 61 71 4F 67 00 48
0x00000050: 0B 01
```



Protocol Version (2 bytes): 0x000C specifies the protocol version of this request.

Minimum Version (2 bytes): 0x000B specifies the minimum version of the protocol schema with which this request is compatible.

Signature: 0x9B069439F329CF9C specifies the signature of this request.

Cell Request Start (4 bytes): 0x00000206 specifies a **32-bit stream object header** (section 2.2.1.5.2) for a cell request start. Decoded, this has a type of 0x40, length 0, and is compound.

User Agent Start (4 bytes): 0x000002EE specifies a **32-bit stream object header** for user agent start. Decoded, this has a type of 0x5D, length 0, and is compound.

User Agent GUID (4 bytes): 0x002002AA specifies a **32-bit stream object header** for a user agent GUID. Decoded, this has a type of 0x55, length 16.

GUID: {"E731B87E-DD45-44AA-80AB80-0C75FBD1530E"} is the GUID of the user agent.

User Agent Version (4 bytes): 0x0008027A specifies a **32-bit stream object header** for user agent version. Decoded, this has a type of 0x2F, length 4.

Version (4 bytes): 0x2EE127B4 specifies the version of the client.

User Agent End (2 bytes): 0x0177 specifies a **16-bit stream object header** (section 2.2.1.5.4) for user agent end.

Sub-request Start (4 bytes): 0x00060216 specifies a **32-bit stream object header** for sub-request start. Decoded, this has a type of 0x42, length 3.

Request Id (2 bytes): 0x03 specifies the request number as a **compact unsigned 64-bit integer** (section 2.2.1.1) for this request. Decoded, this represents a value of 0x1.

Request Type (2 bytes): 0x0B specifies the **request type** (section 2.2.1.6) as a **compact unsigned 64-bit integer**. Decoded, this represents a value of 0x05.

Priority (2 bytes): 0x00 specifies the priority of this sub-request as a **compact unsigned 64-bit integer**.

Put Changes Request (4 bytes): 0x002602D2 specifies a **32-bit stream object header** for **Put Changes** request (section 2.2.2.1.4). Decoded, this has a type of 0x5A, length 9.

Storage Index EXGUID: {"052E2E8E-C0D1-4886-9C51-29D661714F67"} 0x01 specifies the **Storage Index Extended GUID** (see section 2.2.1.12.2) decoded from 0C 8E 2E 2E 05 D1 C0 86 48 9C 51 29 D6 61 71 4F 67.

A - Expected Storage Index EXGUID (1 byte): {"000000-0000-0000-0000-00000000"} 0x00 specifies the expected **Storage Index Extended GUID** decoded from 0x00.

Put Changes Flags (1 byte): 0x48 specifies the flags on the **Put Changes** request (section 2.2.2.1.4).

Sub-Request End (2 bytes): 0x010B specifies the **16-bit stream object header** for **sub-request** end. Decoded, this has a type of 0x21.

Data Element Package Start (2 bytes): 0x02AC specifies the **16-bit stream object header** for a **data element package** (section 2.2.1.12) start. Decoded, this has a type of 0x15, length 1, and is compound.

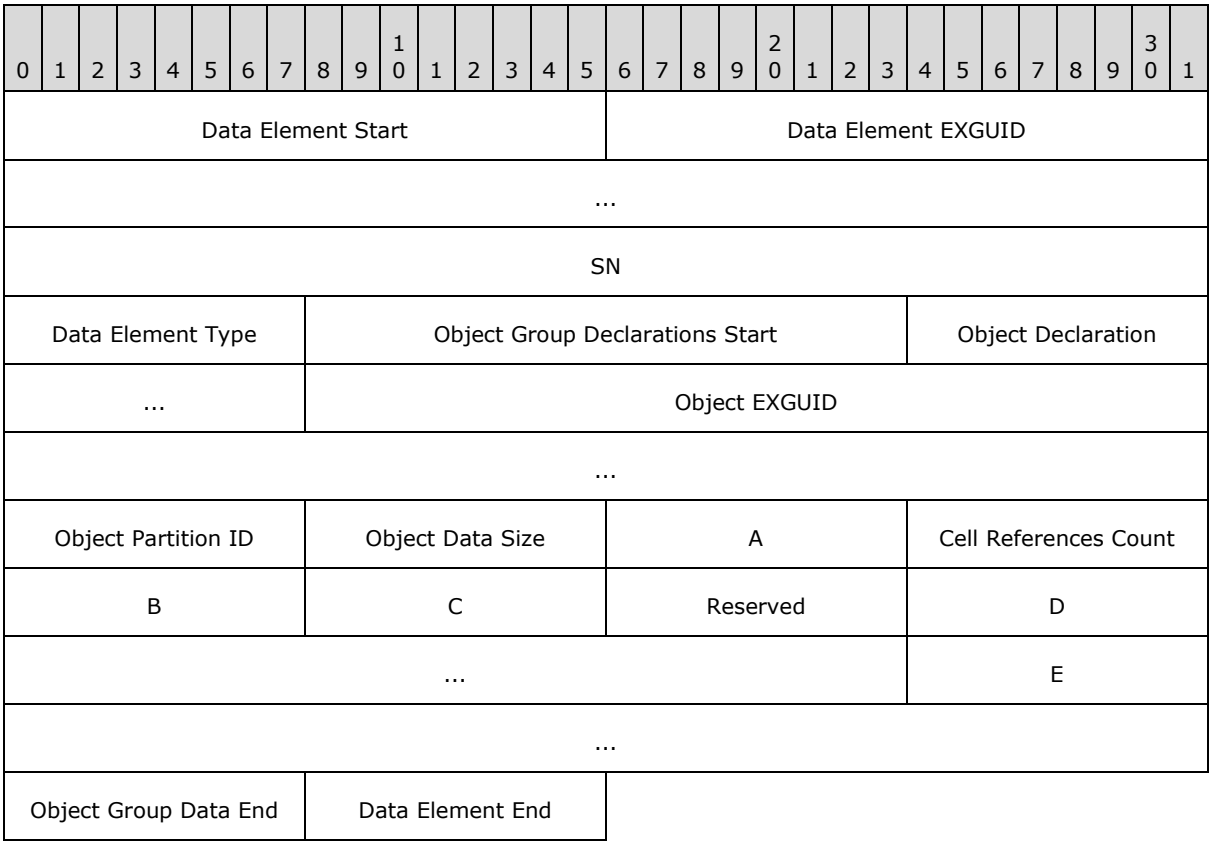
Reserved (1 byte): 0x00 specifies a reserved byte.

4.3.2 Object Group

This is the **Object Group** that is part of the **Put Changes Request** example (section 4.3).

```
0x00000050:       AC 02 00 0C 56 0C 8E FC 0B 2C 04 9B 61 4C
0x00000060: AB 49 48 45 E6 03 EC A0 80 47 AF 30 54 71 6E 9B
0x00000070: 40 98 06 70 7E 81 8D C1 02 01 00 00 00 00 00 00
```

```
0x00000080: 00 0B EC 00 C0 32 80 13 38 0C DE AF 7C 55 4E 95
0x00000090: 0E 65 7A D3 A3 FA 63 01 00 00 11 03 21 2F 00 75
0x000000A0: F4 00 B2 00 EC 03
                    502 Binary Bytes
0x00000290:                                   79 05
```



Data Element Start (2 bytes): 0x560C specifies the **16-bit stream object header** (section 2.2.1.5.1) for a data element start. Decoded, this has a type of 0x1, length 43, and is compound.

Data Element EXGUID: {"2C0BFC8E-9B04-4C61-AB49-4845E603ECA0"} 0x01 specifies the **Data Element Extended GUID** (see section 2.2.1.12.2) decoded from 0C 8E FC 0B 2C 04 9B 61 4C AB 49 48 45 E6 03 EC A0.

SN (4 bytes): (0x80 {"5430AF47-6E71-409B-9806-707E818DC102"} 0x01) specifies the **serial number** (section 2.2.1.9) decoded from 80 47 AF 30 54 71 6E 9B 40 98 06 70 7E 81 8D C1 02 01 00 00 00 00 00 00.

Data Element Type (1 byte): 0x0B specifies the data element type as a **compact unsigned 64-bit integer** (section 2.2.1.1). Decoded, this represents a data element type of 0x5.

Object Group Declarations Start (2 bytes): 0x00EC specifies the **16-bit stream object header** for **object group declaration** (section 2.2.1.12.6.1) start. Decoded, this has a type of 0x1D, length 0 and is compound.

Object Declaration (1 byte): 0x32C0 specifies the **8-bit stream object header** for an **object declaration** start. Decoded, this has a type of 0x18, length 25.

Object EXGUID: {"4E557CAF-0E95-7A65-D3A3- A3FA6301000011"} 0xDE0C3813 specifies the **object extended GUID** decoded from 80 13 38 0C DE AF 7C 55 4E 95 0E 65 7A D3 A3 FA 63 01 00 00 11.

Object Partition ID (1 byte): 0x03 specifies an object partition identifier as a **compact unsigned 64-bit integer** with a decoded value of 0x01.

Object Data Size (1 byte): 0x21 specifies the size of bytes of the object as a **compact unsigned 64-bit integer**. Decoded, this represents 0x16.

A - Object References Count (1 byte): 0x2f specifies the number of object references as a **compact unsigned 64-bit integer** with a decoded value of 0x17.

Cell References Count (1 byte): 0x00 specifies the number of cell references as a **compact unsigned 64-bit integer** with a decoded value of 0x00.

B - Object Group Declaration end (1 byte): 0x75 specifies the **8-bit stream object header** (section 2.2.1.5.3) for an **object group declaration** end.

C - Cell Object Group Data Header (1 byte): 0x00F4 specifies the **8-bit stream object header** for a cell object group data header. Decoded, this has a type of 0x0E, length 0x01.

Reserved (1 byte): Set to 0x00.

D - Cell Object Group Object Data (4 bytes): 0x03EC00B2 specifies the **16-bit stream object header** for a cell object group object data. Decoded, this has a type of 0x16, length 502.

E - More Object Group Data Elements: The rest of the **object group data elements** (section 2.2.1.12.6) have been omitted from this example.

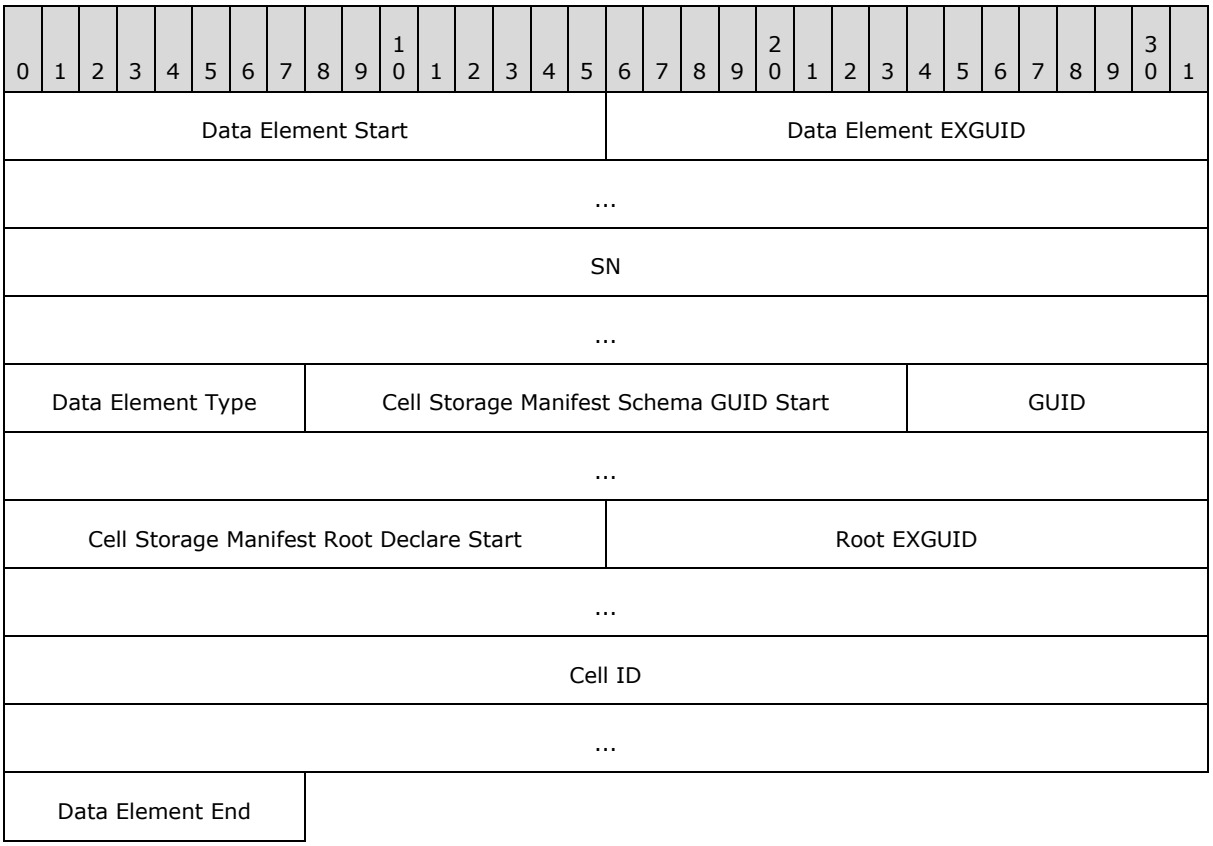
Object Group Data End (1 byte): 0x79 specifies the **8-bit stream object header** for object group data end.

Data Element End (1 byte): 0x05 specifies the **8-bit stream object header** for data element end.

4.3.3 Storage Manifest

This is the **Storage Manifest** that is part of the **Put Changes Request** example (section 4.3).

```
0x00006600:                                             0C 56
0x00006610: 0C 99 FA 30 D7 2C 12 88 42 B7 22 0A 12 5C FD A7
0x00006620: E5 80 47 AF 30 54 71 6E 9B 40 98 06 70 7E 81 8D
0x00006630: C1 02 32 00 00 00 00 00 00 00 05 60 20 94 33 B9
0x00006640: 0E 1D 57 E9 41 AA D3 88 0D 92 D3 19 55 38 66 14
0x00006650: B9 FA DE 84 A3 AA 0D 4A A3 A8 52 0C 77 AC 70 73
0x00006660: 0C B9 FA DE 84 A3 AA 0D 4A A3 A8 52 0C 77 AC 70
0x00006670: 73 0C 65 46 2A 6F C8 42 C7 46 BA B4 E2 8F DC E1
0x00006680: E3 2B 05
```



Data Element Start (2 bytes): 0x560C specifies the **16-bit stream object header** (section 2.2.1.5.1) for a data element start. Decoded, this has a type of 0x1, length 43, and is compound.

Data Element EXGUID: {"D730FA99-122C-4288-22B7-E5A7FD5C120A"} 0x01 specifies a string representation of the **Data Element Extended GUID** (see section 2.2.1.12.2) decoded from 0C 99 FA 30 D7 2C 12 88 42 B7 22 0A 12 5C FD A7 E5.

SN: 80 {"5430AF47-6E71-409B-9806-707E818DC102"} 0x32 specifies a string representation of the **serial number** (section 2.2.1.9) decoded from 80 47 AF 30 54 71 6E 9B 40 98 06 70 7E 81 8D C1 02 32 00 00 00 00 00 00 00.

Data Element Type (1 byte): 0x05 specifies the data element type (section 2.2.1.12.1) as a **compact unsigned 64-bit integer** (section 2.2.1.1). Decoded, this represents a data element type of 0x2.

Cell Storage Manifest Schema GUID Start (2 bytes): 0x2060 specifies the **16-bit stream object header** for a cell storage manifest schema GUID. Decoded, this has a type of 0x0C, length 16.

GUID: {"0EB93394-571D-41E9-AAD3-880D92D31955"} specifies a string representation of the schema GUID.

Cell Storage Manifest Root Declare (2 bytes): 0x6638 specifies the **16-bit stream object header** for a cell storage manifest root declare. Decoded, this has a type of 0x07, length 51.

Root EXGUID: {"84DEFAB9-AAA3-4A0D-A3A8-520C77AC7073"} 0x02 specifies a string representation of the root storage manifest extended GUID decoded from 14 B9 FA DE 84 A3 AA 0D 4A A3 A8 52 0C 77 AC 70 73.

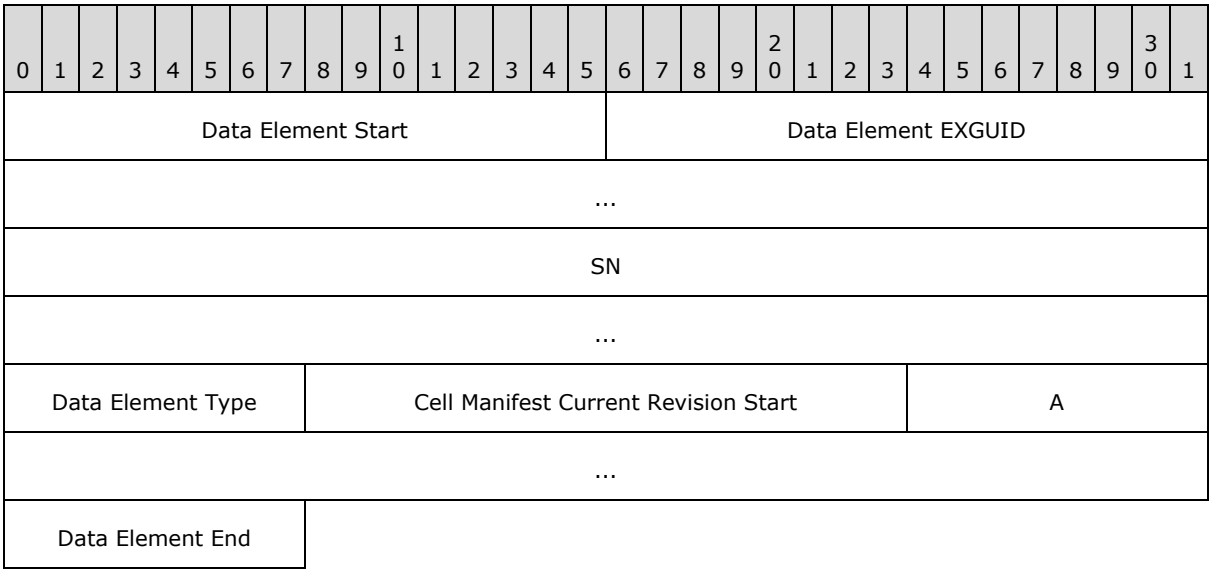
Cell ID: {"84DEFAB9-AAA3-52A8-0C77-520C77AC7073"} 0x01, {"6F2A4665-42C8-46C7-BAB4-E28FDCE1E32B"} 0x01 specifies a string representation of the **cell ID** (section 2.2.1.10) decoded from 0C B9 FA DE 84 A3 AA 0D 4A A3 A8 52 0C 77 AC 70 73 0C 65 46 2A 6F C8 42 C7 46 BA B4 E2 8F DC E1 E3 2B

Data Element End (1 byte): 0x05 specifies the **8-bit stream object header** (section 2.2.1.5.3) for data element end.

4.3.4 Cell Manifest

This is the **Cell Manifest** that is part of the **Put Changes Request** example (section 4.3).

```
0x00006680:          0C 58 60 0C 8E FC 0B 2C 04 9B 61 4C AB
0x00006690: 49 48 45 E6 03 EC A0 80 47 AF 30 54 71 6E 9B 40
0x000066A0: 98 06 70 7E 81 8D C1 02 33 00 00 00 00 00 00 00
0x000066B0: 07 58 22 0C 3A FE 28 71 BE DC 01 43 BD 84 71 6C
0x000066C0: 45 6C 80 8A 05
```



Data Element Start (2 bytes): 0x580C specifies the **16-bit stream object header** (section 2.2.1.5.1) for a data element start. Decoded, this has a type of 0x1, length 44, and is compound.

Data Element EXGUID: {"2C0BFC8E-9B04-4C61-AB49-4845E603ECA0"} 0x31 specifies a string representation of the **Data Element Extended GUID** (see section 2.2.1.12.2) decoded from 60 0C 8E FC 0B 2C 04 9B 61 4C AB 49 48 45 E6 03 EC A0

SN: 0x80 {"5430AF47-6E71-409B-0698-02C18D817E70"} 0x33 specifies the **serial number** (section 2.2.1.9) decoded from 80 47 AF 30 54 71 6E 9B 40 98 06 70 7E 81 8D C1 02 33 00 00 00 00 00 00 00.

Data Element Type (1 byte): 0x07 specifies the data element type (section 2.2.1.12.1) as a **compact unsigned 64-bit integer** (section 2.2.1.1). Decoded, this represents a data element type of 0x3.

Cell Manifest Current Revision Start (2 bytes): 0x2258 specifies the **16-bit stream object header** for a cell manifest current revision start. Decoded, this has a type of 0x0B, length 17.

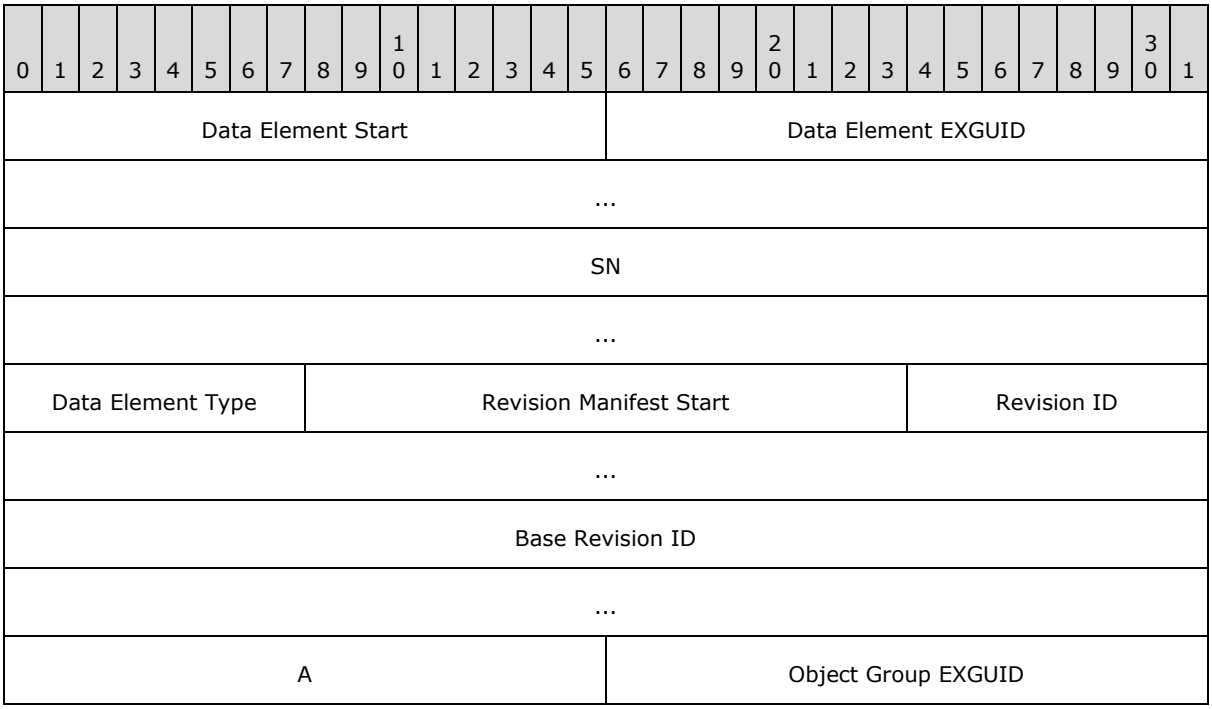
A - Cell Manifest Current Revision EXGUID: {"7128FE2A-DCBE-4301-BD84-716C456C808A"} 0x01 specifies a string representation of the current revision **extended GUID** (section 2.2.1.7) decoded from 0C 3A FE 28 71 BE DC 01 43 BD 84 71 6C 45 6C 80 8A.

Data Element End (1 byte): 0x05 specifies the **8-bit stream object header** (section 2.2.1.5.3) for data element end.

4.3.5 Revision Manifest

This is the **Revision Manifest** that is part of **the Put Changes Request** example (section 4.3).

```
0x000066c0                  0C 56 0C 05 A9 D1 DF 9C 9B 2E 42
0x000066D0: B2 59 81 7A F3 51 14 54 80 47 AF 30 54 71 6E 9B
0x000066E0: 40 98 06 70 7E 81 8D C1 02 34 00 00 00 00 00 00
0x000066F0: 00 09 D0 24 0C 3A FE 28 71 BE DC 01 43 BD 84 71
0x00006700: 6C 45 6C 80 8A 00 50 4C 14 B9 FA DE 84 A3 AA 0D
0x00006710: 4A A3 A8 52 0C 77 AC 70 73 80 13 38 0C DE AF 7C
0x00006720: 55 4E 95 0E 65 7A D3 A3 FA 63 01 00 00 11 C8 22
0x00006730: 0C 8E FC 0B 2C 04 9B 61 4C AB 49 48 45 E6 03 EC
0x00006740: A0  . . .
       More Revision Manifest Object Group references
0x0006AB0                                   05
```



...
More Revision Manifest Object Group References
...
Data Element End

Data Element Start (2 bytes): 0x560C specifies the **16-bit stream object header** (section 2.2.1.5.1) for a data element start. Decoded, this has a type of 0x1, length 43, and is compound.

Data Element EXGUID: {"DFD1A905-9B9C-422E-42B2-817AF3511454"} 0x01 specifies the data element **extended GUID** (section 2.2.1.7) decoded from 0C 05 A9 D1 DF 9C 9B 2E 42 B2 59 81 7A F3 51 14 54.

SN: (0x80 {"5430AF47-6E71-409B-9806-707E818DC102"} 0x34) specifies a string representation of the **serial number** (section 2.2.1.9) decoded from 80 47 AF 30 54 71 6E 9B 40 98 06 70 7E 81 8D C1 02 34 00 00 00 00 00 00 00.

Data Element Type (1 byte): 0x09 specifies the data element type (section 2.2.1.12.1) as a **compact unsigned 64-bit integer** (section 2.2.1.1). Decoded, this represents a data element type of 0x4.

Revision Manifest Start (2 bytes): 0x24D0 specifies the **16-bit stream object header** for revision manifest start. Decoded, this has type 0x1A, length 18.

Revision ID: {"84DEFAB9-0DAA-A34A-A852-520C77AC7073"} 0x02 specifies the revision identifier, in the form of an **extended GUID** decoded from 14 B9 FA DE 84 A3 AA 0D 4A A3 A8 52 0C 77 AC 70 73.

Base Revision ID: {"4E557CAF-0E95-7A65-D3A3- FA6301000011"} 0xDE0C3813 specifies the base revision identifier, in the form of an **extended GUID** decoded from 80 13 38 0C DE AF 7C 55 4E 95 0E 65 7A D3 A3 FA 63 01 00 00 11.

A - Revision Manifest Object Group Reference Start (2 bytes): 0x22C8 specifies the **16-bit stream object header** for revision manifest object group reference start. Decoded, this has a type of 0x19, length 17.

Object Group EXGUID: {"2C0BFC8E-9B04-4C61-AB49-4445E603ECA0"} 0x01 specifies the object group **extended GUID** decoded from 0C 8E FC 0B 2C 04 9B 61 4C AB 49 44 45 E6 03 EC A0.

More Revision Manifest Object Group References: The rest of the revision manifest object group reference elements have been omitted from this example.

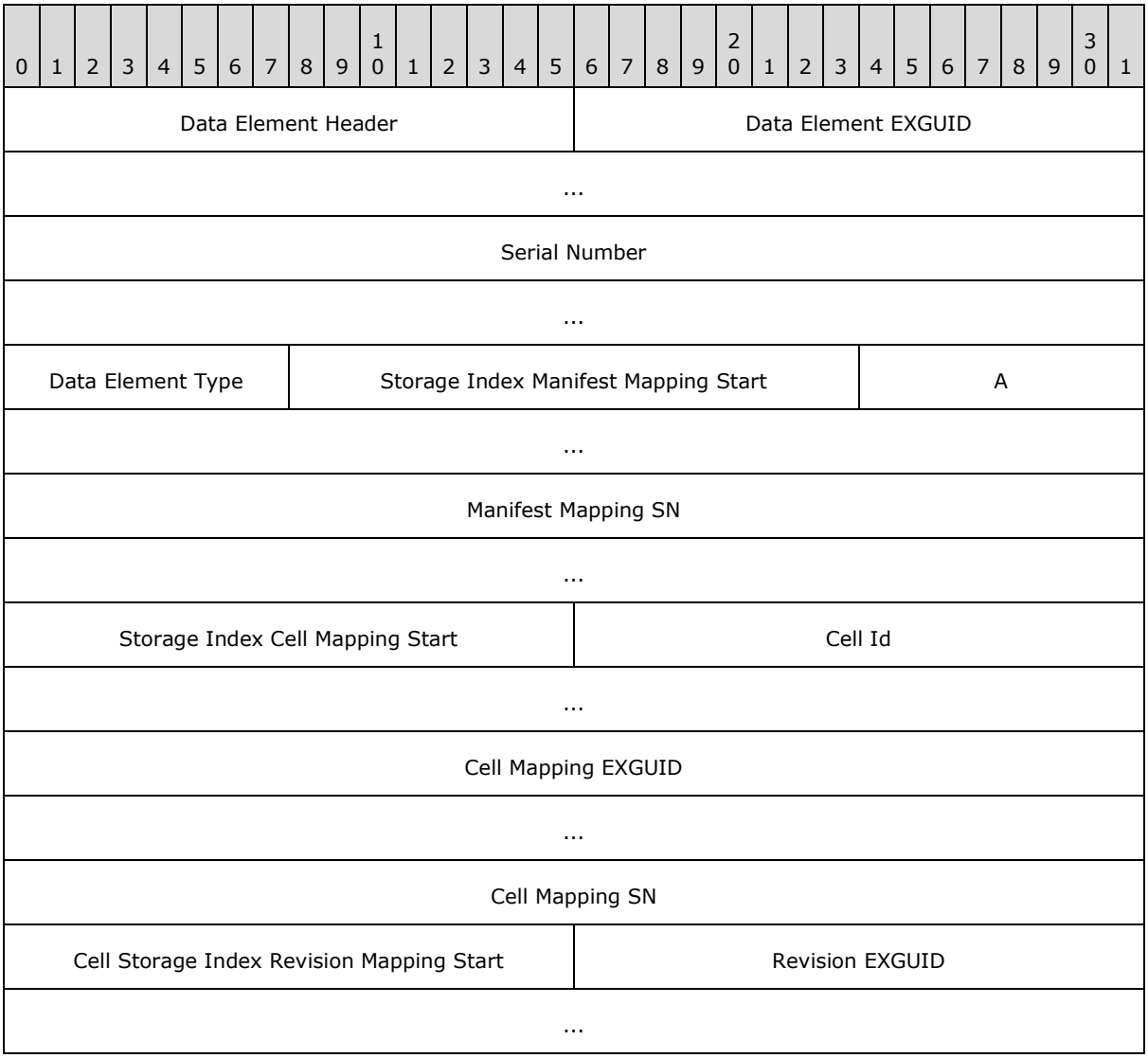
Data Element End (1 byte): 0x05 specifies the **8-bit stream object header** (section 2.2.1.5.3) for data element end.

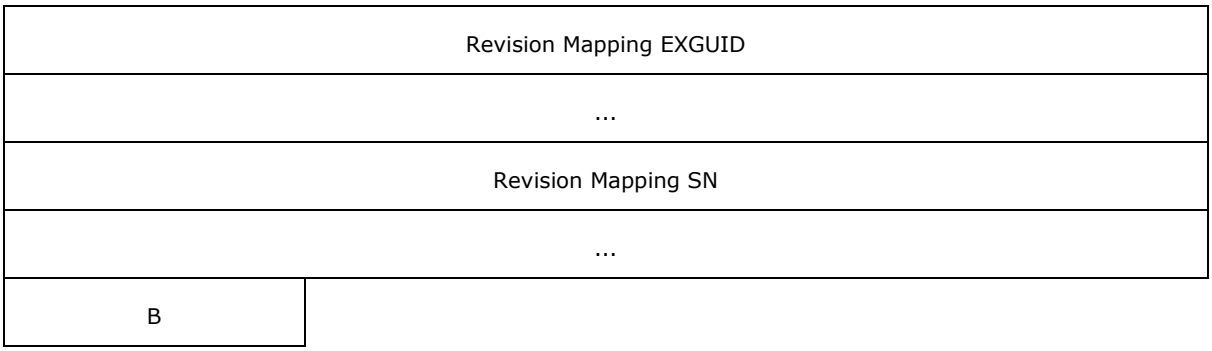
4.3.6 Storage Index

This is the **Storage Index** that is part of the **Put Changes Request** example (section 4.3).

```
0x00006AB0:                                     0C 56 0C 8E
```

```
0x00006AC0: 2E 2E 05 D1 C0 86 48 9C 51 29 D6 61 71 4F 67 80
0x00006AD0: 0A 4E D0 67 25 4F E5 43 91 48 B7 28 D3 AB 89 77
0x00006AE0: 01 00 00 00 00 00 00 00 03 88 54 0C 99 FA 30 D7
0x00006AF0: 2C 12 88 42 B7 22 0A 12 5C FD A7 E5 80 B8 50 CF
0x00006B00: AB 8E 91 64 BF 98 06 70 7E 81 8D C1 02 3E 00 00
0x00006B10: 00 00 00 00 00 70 9A 0C B9 FA DE 84 A3 AA 0D 4A
0x00006B20: A3 A8 52 0C 77 AC 70 73 0C 65 46 2A 6F C8 42 C7
0x00006B30: 46 BA B4 E2 8F DC E1 E3 2B 60 0C 8E FC 0B 2C 04
0x00006B40: 9B 61 4C AB 49 48 45 E6 03 EC A0 80 B8 50 CF AB
0x00006B50: 8E 91 64 BF 98 06 70 7E 81 8D C1 02 40 00 00 00
0x00006B60: 00 00 00 00 68 76 0C 3A FE 28 71 BE DC 01 43 BD
0x00006B70: 84 71 6C 45 6C 80 8A 0C 05 A9 D1 DF 9C 9B 2E 42
0x00006B80: B2 59 81 7A F3 51 14 54 80 B8 50 CF AB 8E 91 64
0x00006B90: BF 98 06 70 7E 81 8D C1 02 3F 00 00 00 00 00 00
0x00006BA0: 00 05 55 03 01
```





Data Element Start (2 bytes): 0x560C specifies the **16-bit stream object header** (section 2.2.1.5.1) for a data element start. Decoded, this has a type of 0x1, length 43, and is compound.

Data Element EXGUID: {"052E2E8E-C0D1-4886-9C51-29D661714F67"} 0x01 specifies the data element extended GUID decoded from 0C 8E 2E 2E 05 D1 C0 86 48 9C 51 29 D6 61 71 4F 67.

SN: 0x80 {"67D04E0A-4F25-43E5-9148-B728D3AB8977"} 0x01 specifies the **serial number** (section 2.2.1.9) decoded from 80 0A 4E D0 67 25 4F E5 43 91 48 B7 28 D3 AB 89 77 01 00 00 00 00 00 00 00.

Data Element Type (1 byte): 0x03 specifies the data element type (section 2.2.1.12.1) as a **compact unsigned 64-bit integer** (section 2.2.1.1). Decoded, this represents a data element type of 0x1.

Storage Index Manifest Mapping Start (2 bytes): 0x5488 specifies the **8-bit stream object header** for storage index cell mapping. Decoded, this represents a type of 0x11, length 42.

A - Manifest Mapping EGUID: {"D730FA99-122C-4288-B722-0A125CFDA7E5"} 0x01 specifies the manifest mapping **extended GUID** (section 2.2.1.7) decoded from 0C 99 FA 30 D7 2C 12 88 42 B7 22 0A 12 5C FD A7 E5.

Manifest Mapping SN: 0x80 {"ABCF50B8-918E-BF64-9806-707E818DC102"} 0x3E specifies the manifest mapping **serial number** decoded from 80 B8 50 CF AB 8E 91 64 BF 98 06 70 7E 81 8D C1 02 3E 00 00 00 00 00 00 00.

Storage Index Cell Mapping Start (2 bytes): 0x9A70 specifies the **8-bit stream object header** for storage index cell mapping. Decoded, this has a type of 0x0E, length 77.

Cell Id: {"84DEFAB9-AAA3-4A0D-A3A8-520C77AC7073"} 0x01, {"6F2A4664-42C8-46C7-BAB4-E28FDCE1E32B"} 0x01 specifies the cell identifier decoded from 0C B9 FA DE 84 A3 AA 0D 4A A3 A8 52 0C 77 AC 70 73 0C 65 46 2A 6F C8 42 C7 46 BA B4 E2 8F DC E1 E3 2B.

Cell Mapping EXGUID: {"2C0BFC8E-9B04-4C61-AB49-4845E603ECA0"} 0x31 specifies the cell mapping **extended GUID** decoded from 60 0C 8E FC 0B 2C 04 9B 61 4C AB 49 48 45 E6 03 EC A0.

Cell Mapping SN: 0x80 {"ABCF50B8-918E-BF64-9806-707E818DC102"} 0x40 specifies the cell mapping **serial number** decoded from 80 B8 50 CF AB 8E 91 64 BF 98 06 70 7E 81 8D C1 02 40 00 00 00 00 00 00 00.

Cell Storage Index Revision Mapping Start (2 bytes): 0x7668 specifies the **16-bit stream object header** for cell storage index revision mapping start. Decoded, this represents a type of 0x0D, length 59.

Revision EXGUID: {"7128FE3A-DCBE-4301-BD84-716C456C808A"} 0x01 specifies the revision **extended GUID** decoded from 0C 3A FE 28 71 BE DC 01 43 BD 84 71 6C 45 6C 80 8A.

Revision Mapping EXGUID: {"DFD1A905-9B9C-422E-B259-817AF3511454"} 0x01 specifies the revision mapping **extended GUID** decoded from 0C 05 A9 D1 DF 9C 9B 2E 42 B2 59 81 7A F3 51 14 54.

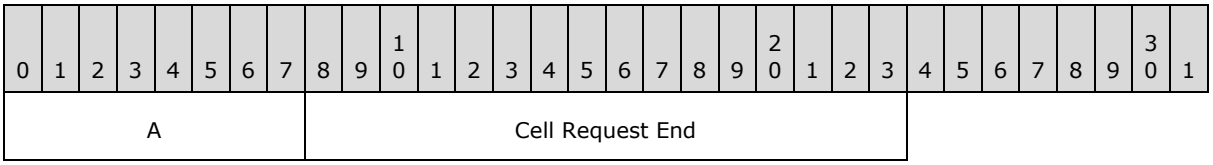
Revision Mapping SN: (0x80 {"8EABCF50-6491-98BF-0670-707E818DC102"} 0x00) specifies the revision mapping **serial number** decoded from 80 B8 50 CF AB 8E 91 64 BF 98 06 70 7E 81 8D C1 02 3F 00 00 00 00 00 00 00.

B - Data Element End (1 byte): 0x05 specifies the **8-bit stream object header** (section 2.2.1.5.3) for data element end.

4.3.7 Request End

This is the **Request End** that is part of the **Put Changes Request** example (section 4.3).

```
0x00006BA0:       55 03 01
```



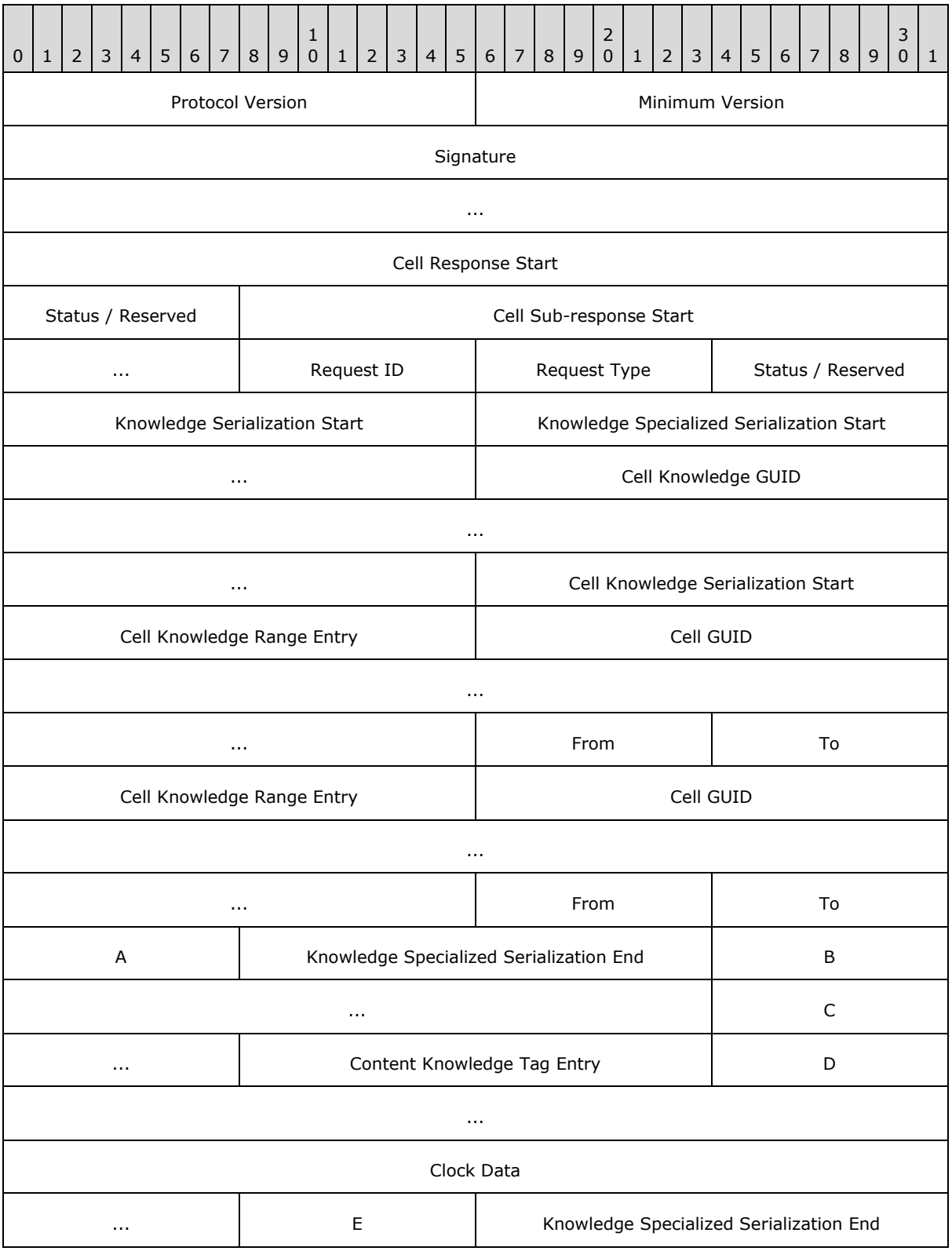
A - Data Element Package End (1 byte): 0x55 specifies the **8-bit stream object header** (section 2.2.1.5.3) for **data element package** (section 2.2.1.12) end. This stream object was started in the **Request Header (**section 4.3.1).

Cell Request End (2 bytes): 0x0103 specifies the **8-bit stream object header** for cell request end. This stream object was started in the **Request Header** section.

4.4 Put Changes Response

This section provides an example of a **Put Changes** sub-response (section 2.2.3.1.3).

```
0x00000000: 0C 00 0B 00 9D CF 29 F3 39 94 06 9B 16 03 02 00
0x00000010: 00 0E 02 06 00 03 0B 00 84 00 26 02 20 00 F6 35
0x00000020: 7A 32 61 07 14 44 96 86 51 E9 00 66 7A 4D A4 00
0x00000030: 78 24 22 92 69 92 46 AD 53 B3 94 89 C2 4F 5A CF
0x00000040: A0 9A 00 E9 78 24 DD 6D 96 6D B9 52 AC 4C 94 89
0x00000050: C2 4F 5A CF A0 9A 00 DF 51 13 01 26 02 20 00 13
0x00000060: 1F 09 10 82 C8 FB 40 98 86 65 33 F9 34 C2 1D 6C
0x00000070: 01 70 2D 0C F9 0B 41 37 6F D1 99 44 A6 C3 27 23
0x00000080: 2E DC A7 11 09 33 00 00 00 B5 13 01 41 07 01 8B
0x00000090: 01
```



F	Cell Sub-response end	Cell Response End
...		

Protocol Version (2 bytes): 0x000C specifies the protocol version of this request.

Minimum Version (2 bytes): 0x000B specifies the minimum version of the protocol schema with which this request is compatible.

Signature: 0x9B069439F329CF9C specifies the signature of this request.

Cell Response Start (4 bytes): 0x00020316 specifies a **32-bit stream object header** (section 2.2.1.5.2) for a cell response start. Decoded, this has a type of 0x062, length 1, and is compound.

Status / Reserved (1 byte): 0x00 represents the status bit and reserved field of the cell response (section 2.2.3).

Cell Sub-response Start (4 bytes): 0x0006020E specifies the **32-bit stream object header** for a cell sub-response start. Decoded, has a type of 0x041, length 3, and is compound.

Request Id (1 byte): 0x03 specifies the request number as a **compact unsigned 64-bit integer** (section 2.2.1.1) for this request. Decoded, this represents a value of 0x1.

Request Type (1 type): 0x0B specifies the **request type** (section 2.2.1.6) as a **compact unsigned 64-bit integer**. Decoded, this represents a value of 0x5.

Status / Reserved (1 byte): 0x00 represents the status bit and reserved field of the cell **sub-response** (section 2.2.3.1).

Knowledge Serialization Start (2 bytes): 0x0084 specifies the **16-bit stream object header** (section 2.2.1.5.1) for a **knowledge** (section 2.2.1.13) serialization start. Decoded, has a type of 0x10, length 0, and is compound.

Knowledge Specialized Serialization Start (4 bytes): 0x00200226 specifies the **32-bit stream object header** for a **knowledge** specialized serialization start. Decoded, this has a type of 0x044, length 16, and is compound.

Cell Knowledge GUID (16 bytes): "327A35F6-0761-4414-9686-51E900667A4D" is the string representation of the **cell knowledge** (section 2.2.1.13.2) GUID.

Cell Knowledge Serialization Start (2 bytes): 0x00A4 specifies the **16-bit stream object header** for **cell knowledge** serialization start. Decoded, this represents a type of 0x14, length 0, and is compound.

Cell Knowledge Range Entry (2 bytes): 0x2478 specifies the **16-bit stream object header** for a **cell knowledge range** entry (section 2.2.1.13.2.1). Decoded, this represents a type of 0x0F, length 18.

Cell GUID (16 bytes): "92699222-AD46-B353-9489-C24F5ACFA09A" is the string representation of the cell GUID.

From (1 byte): 0x00 specifies the beginning of the cell range as a **compact unsigned 64-bit integer**. Decoded, this represents a value of zero.

To (1 byte): 0xE9 specifies the end of the cell range as a **compact unsigned 64-bit integer**. Decoded, this represents a value of 116.

Cell Knowledge Serialization Start: 0x00A4 specifies the **16-bit stream object header** for the **cell knowledge** serialization start. Decoded, this represents a type of 0x14, length 0, and is compound.

Cell Knowledge Range Entry: 0x2478 specifies the **16-bit stream object header** for a **cell knowledge range** entry. Decoded, this represents a type of 0x0F, length 18.

Cell GUID (16 bytes): "6D966DDD-52B9-4CAC-9489-C24F5ACFA09A" is the string representation of the cell GUID.

From (1 byte): 0x00 specifies the beginning of the cell range as a **compact unsigned 64-bit integer**. Decoded, this represents a value of zero.

To (1 byte): 0xDF specifies the end of the cell range as a **compact unsigned 64-bit integer**. Decoded, this represents a value of 111.

A - Cell Knowledge Serialization End (1 byte): 0x51 specifies an **8-bit stream object header** (section 2.2.1.5.3) for a **cell knowledge** serialization end. Decoded, this has a type of 0x14.

Knowledge Specialized Serialization End (2 bytes): 0x0113 specifies a **16-bit stream object header** (section 2.2.1.5.4) for a **knowledge** specialized serialization end. Decoded, this has a type of 0x044.

B - Knowledge Specialized Serialization Start (4 bytes): 0x00200226 specifies a **32-bit stream object header** for **knowledge** specialized serialization start. Decoded, this has a type of 0x044, length of 16, and is compound.

Content Tag Knowledge GUID (16 bytes): "10091F13-C882-40FB-9886-6533F934C21D" is the string representation of the **content tag knowledge** (section 2.2.1.13.5) GUID.

C - Content Tag Knowledge (2 bytes): 0x016C specifies the **16-bit stream object header** for a **content tag knowledge** start. Decoded, this has type 0x2D, length 0, and is compound.

Content Tag Knowledge Entry (2 bytes): 0x2D70 specifies the **16-bit stream object header** for a **content tag knowledge entry**. Decoded, this has type 0x2E and length 22.

D - BLOB Heap Extended GUID (16 bytes): "37410BF9-D16F-4499-A6C3-27232EDCA711" is the string representation of the GUID portion, and 0x01 is the integer portion of the decoded **extended GUID** (section 2.2.1.7).

Clock Data: 0x0000003309 specifies the **binary item** (section 2.2.1.3) in the **content tag knowledge entry** (section 2.2.1.13.5.1).

E - Content Tag Knowledge End (1 byte): 0xB5 specifies the **stream object header** for a **content tag knowledge** end. Decoded, this has type 0x2D.

Knowledge Specialized Serialization End (2 bytes): 0x0113 specifies a **16-bit stream object header** for **knowledge** specialized serialization end. Decoded, this has a type of 0x044.

F - Knowledge Serialization End (1 byte): 0x41 specifies an **8-bit stream object header** for **knowledge** serialization end. Decoded, this has a type of 0x10.

Cell Sub-response end (2 bytes): 0x0107 specifies a **16-bit stream object header** for cell sub-response end. Decoded, this has a value of 0x041.

Cell Response End (2 bytes): 0x018B specifies a **16-bit stream object header** for a cell response end. Decoded, this has a value of 0x62.

5 Security

5.1 Security Considerations for Implementers

This protocol does not introduce any additional security considerations beyond those that apply to its containing protocol [MS-FSSHTTP].

5.2 Index of Security Parameters

None.

6 Appendix A: Full IDL

None.

7 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft Office 2010 suites
- Microsoft SharePoint Server 2010
- Microsoft SharePoint Workspace 2010
- Microsoft Office 2013
- Microsoft SharePoint Server 2013

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product 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 2.2.1.12.6.3: SharePoint Server 2010 and SharePoint Workspace 2010 do not support the **Object Metadata Declaration**.

<2> Section 2.2.2: Data Element Hashing is not supported by Office 2010 and support is configuration dependent for other versions.

<3> Section 2.2.2: SharePoint Server 2010 and SharePoint Workspace 2010 do not support the **Request Hashing Options Declaration** field.

<4> Section 2.2.2.1: SharePoint Server 2010 and SharePoint Workspace 2010 do not support the **Target Partition Id** field.

<5> Section 3.1.4.2: Data Element Hashing is not supported by Office 2010 and support is configuration dependent for other versions.

<6> Section 3.1.4.4: SharePoint Server 2010 and SharePoint Workspace 2010 not support this sub-request.

<7> Section 3.2.4.2: Data Element Hashing is not supported by Office 2010 and support is configuration dependent for other versions.

8 Change Tracking

This section identifies changes that were made to the [MS-FSSHTTPB] protocol document between the February 2013 and July 2013 releases. Changes are classified as New, Major, Minor, Editorial, or No change.

The revision class **New** means that a new document is being released.

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 or functionality.
- An extensive rewrite, addition, or deletion of major portions of content.
- The removal of a document from the documentation set.
- Changes made for template compliance.

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 **Editorial** means that the language and formatting in the technical content was changed. Editorial changes apply to grammatical, formatting, and style issues.

The revision class **No change** means that no new technical or language changes were introduced. The technical content of the document is identical to the last released version, but minor editorial and formatting changes, as well as updates to the header and footer information, and to the revision summary, may have been made.

Major and minor changes can be described further using the following change types:

- New content added.
- Content updated.
- Content removed.
- New product behavior note added.
- Product behavior note updated.
- Product behavior note removed.
- New protocol syntax added.
- Protocol syntax updated.
- Protocol syntax removed.
- New content added due to protocol revision.
- Content updated due to protocol revision.
- Content removed due to protocol revision.
- New protocol syntax added due to protocol revision.

- Protocol syntax updated due to protocol revision.
- Protocol syntax removed due to protocol revision.
- New content added for template compliance.
- Content updated for template compliance.
- Content removed for template compliance.
- Obsolete document removed.

Editorial changes are always classified with the change type **Editorially updated.**

Some important terms used in the change type descriptions are defined as follows:

- **Protocol syntax** refers to data elements (such as packets, structures, enumerations, and methods) as well as interfaces.
- **Protocol revision** refers to changes made to a protocol that affect the bits that are sent over the wire.

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

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
2.2.1.5.1 16-bit Stream Object Header Start	Updated description for "Length".	N	Content updated.
2.2.1.5.2 32-bit Stream Object Header Start	Added the "Error String Supplemental Info" object type.	N	Content updated.
2.2.1.5.2 32-bit Stream Object Header Start	Updated the compound value for "Target Partition Id" from 0 to 1.	N	Content updated.
2.2.1.5.2 32-bit Stream Object Header Start	Updated description for "Length".	N	Content updated.
2.2.1.12.6.3 Object Metadata Declaration	Added one product behavior note for "Object Metadata Declaration".	N	New product behavior note added.
2.2.2.1.3 Query Changes	Updated the description for "Max Data Elements".	N	Content updated.
2.2.2.1.4 Put Changes	Updated the description for "E - Abort Remaining Put Changes on Failure".	N	Content updated.
2.2.3.1 Sub-Responses	Removed "Data Element Package (variable)".	N	Content removed.
2.2.3.1.4	Updated the description for the "Integer	N	Content updated.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
Allocate ExtendedGuid Range	Range Max (variable)" element.		
2.2.3.2 Response Error	Added "Error String Supplemental Info Start" and "Error String Supplemental Info".	N	Content updated.

9 Index

N

O

P

Q

R

S

T

V