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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
7/13/2009	0.1	Major	Initial Availability
8/28/2009	0.2	Editorial	Revised and edited the technical content
11/6/2009	0.3	Editorial	Revised and edited the technical content
2/19/2010	1.0	Major	Updated and revised the technical content
3/31/2010	1.01	Editorial	Revised and edited the technical content
4/30/2010	1.02	Editorial	Revised and edited the technical content
6/7/2010	1.03	Editorial	Revised and edited the technical content
6/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
7/23/2010	1.05	Minor	Clarified the meaning of the technical content.
9/27/2010	1.06	Editorial	Changed language and formatting in the technical content.
11/15/2010	1.06	None	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.
3/18/2011	1.07	None	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	1.07	None	No changes to the meaning, language, or formatting of the technical content.
1/20/2012	2.0	Major	Significantly changed the technical content.
4/11/2012	2.0	None	No changes to the meaning, language, or formatting of the technical content.
7/16/2012	2.1	Minor	Clarified the meaning of the technical content.
9/12/2012	2.1	None	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	3.0	Major	Significantly changed the technical content.
2/11/2013	4.0	Major	Significantly changed the technical content.
7/30/2013	4.1	Minor	Clarified the meaning of the technical content.
11/18/2013	4.2	Minor	Clarified the meaning of the technical content.
2/10/2014	4.2	None	No changes to the meaning, language, or formatting of the technical content.
4/30/2014	5.0	Major	Significantly changed the technical content.
7/31/2014	6.0	Major	Significantly changed the technical content.
10/30/2014	6.1	Minor	Clarified the meaning of the technical content.
3/16/2015	7.0	Major	Significantly changed the technical content.

Date	Revision History	Revision Class	Comments
6/30/2015	8.0	Major	Significantly changed the technical content.
2/26/2016	9.0	Major	Significantly changed the technical content.
4/14/2016	10.0	Major	Significantly changed the technical content.
7/15/2016	10.0	None	No changes to the meaning, language, or formatting of the technical content.
9/14/2016	10.1	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 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

binary large object (BLOB): A discrete packet of data that is stored in a database and is treated as a sequence of uninterpreted bytes.

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

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

Hypertext Transfer Protocol (HTTP): An application-level protocol for distributed, collaborative, hypermedia information systems (text, graphic images, sound, video, and other multimedia files) on the World Wide Web.

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

Session Initiation Protocol (SIP) address: A URI that does not include a "sip:" prefix and is used to establish multimedia communications sessions between two or more users over an IP network, as described in [RFC3261].

SOAP: A lightweight protocol for exchanging structured information in a decentralized, distributed environment. **SOAP** uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation-specific semantics. SOAP 1.2 supersedes SOAP 1.1. See [SOAP1.2-1/2003].

UTF-16: A standard for encoding Unicode characters, defined in the Unicode standard, in which the most commonly used characters are defined as double-byte characters. Unless specified otherwise, this term refers to the UTF-16 encoding form specified in [UNICODE5.0.0/2007] section 3.9.

UTF-8: A byte-oriented standard for encoding Unicode characters, defined in the Unicode standard. Unless specified otherwise, this term refers to the UTF-8 encoding form specified in [UNICODE5.0.0/2007] section 3.9.

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

1.2 References

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

1.2.1 Normative References

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

[MS-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

[RFC2822] Resnick, P., Ed., "Internet Message Format", RFC 2822, April 2001, http://www.ietf.org/rfc/rfc2822.txt

1.2.2 Informative References

[MS-FSSHTTPD] Microsoft Corporation, "Binary Data Format for File Synchronization via SOAP".

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 **Simple Object Access Protocol (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.

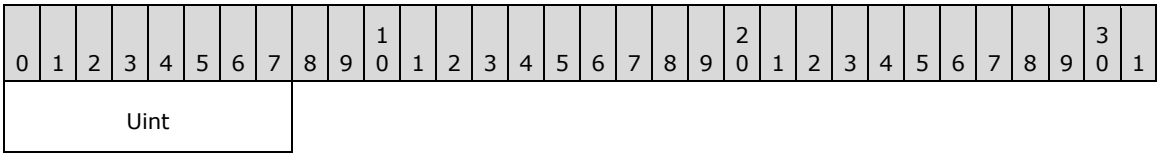
2.2.1 Basic Types

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

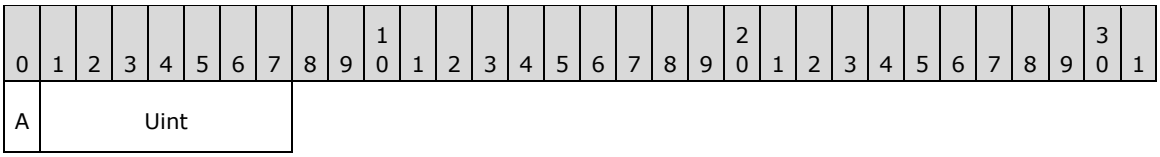
A 1-byte encoding of the value zero.



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.

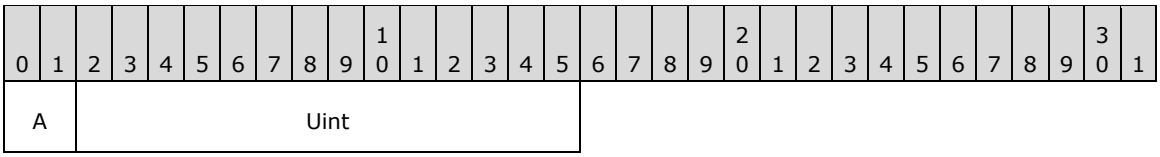


A – Type (1 bit): A flag that specifies this format from all other formats of a compact unsigned 64-bit integer (section 2.2.1.1), and MUST be one.

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.

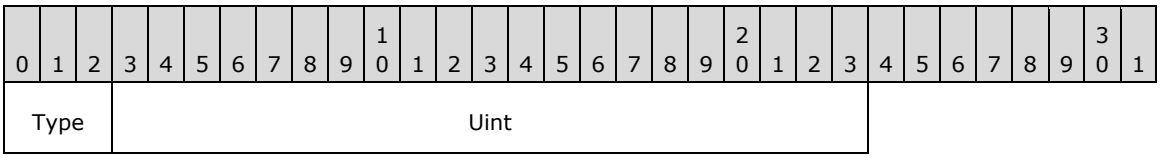


A – Type (2 bits): A flag that specifies this format from all other formats of a compact unsigned 64-bit integer (section 2.2.1.1), and MUST be two.

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.

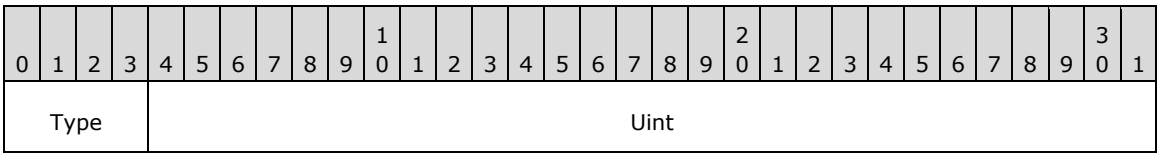


Type (3 bits): A flag that specifies this format from all other formats of a compact unsigned 64-bit integer (section 2.2.1.1), and MUST be four.

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.

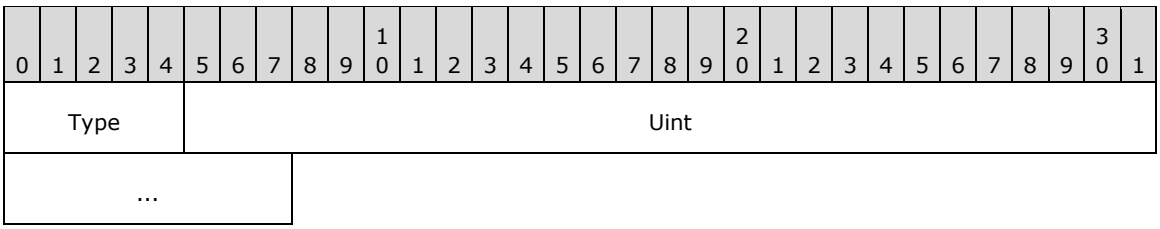


Type (4 bits): A flag that specifies this format from all other formats of a compact unsigned 64-bit integer (section 2.2.1.1), and MUST be eight.

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.

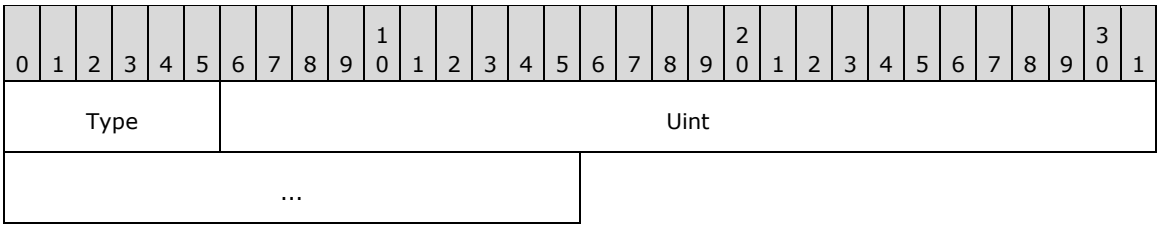


Type (5 bits): A flag that specifies this format from all other formats of a compact unsigned 64-bit integer (section 2.2.1.1), and MUST be 16.

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.

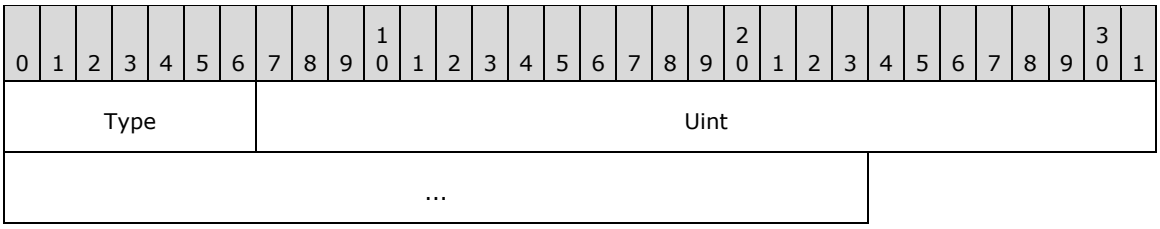


Type (6 bits): A flag that specifies this format from all other formats of a compact unsigned 64-bit integer (section 2.2.1.1), and MUST be 32.

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.

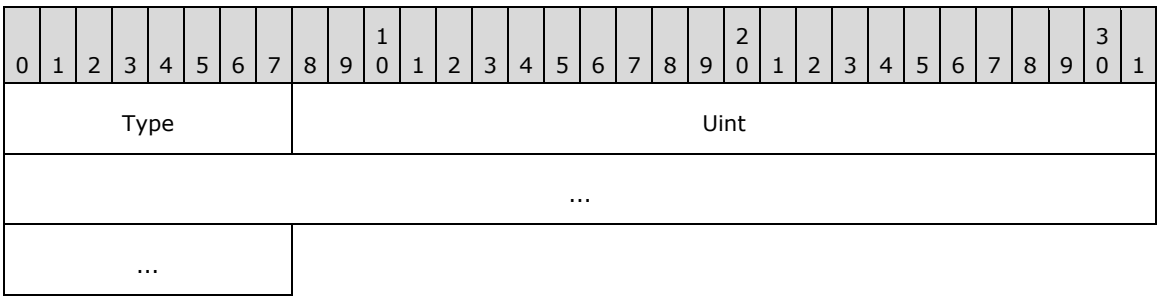


Type (7 bits): A flag that specifies this format from all other formats of a compact unsigned 64-bit integer (section 2.2.1.1), and MUST be 64.

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.

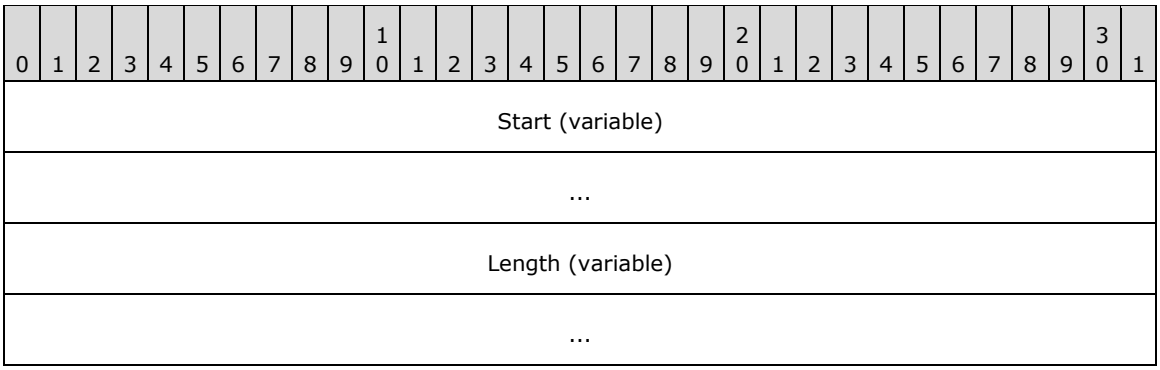


Type (8 bits): A flag that specifies this format from all other formats of a compact unsigned 64-bit integer (section 2.2.1.1), and MUST be 128.

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.

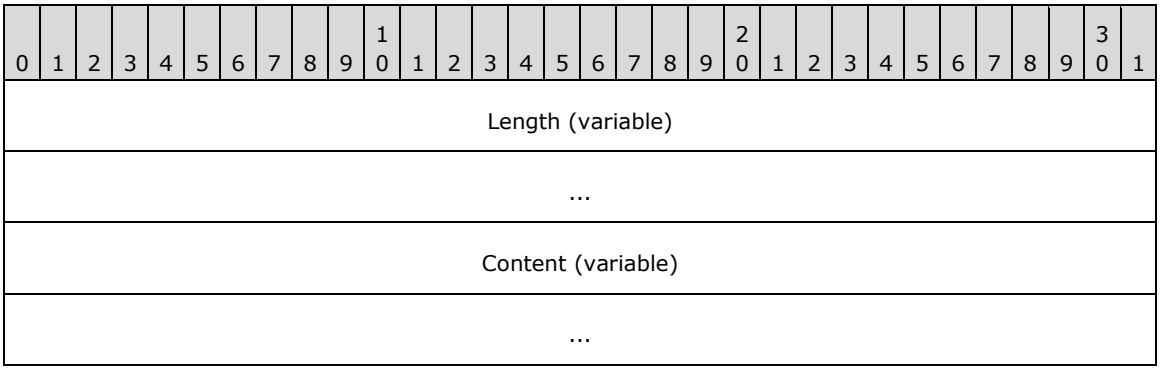


Start (variable): A compact unsigned 64-bit integer (section 2.2.1.1) that specifies the byte-offset within the file of the beginning of the file chunk.

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.

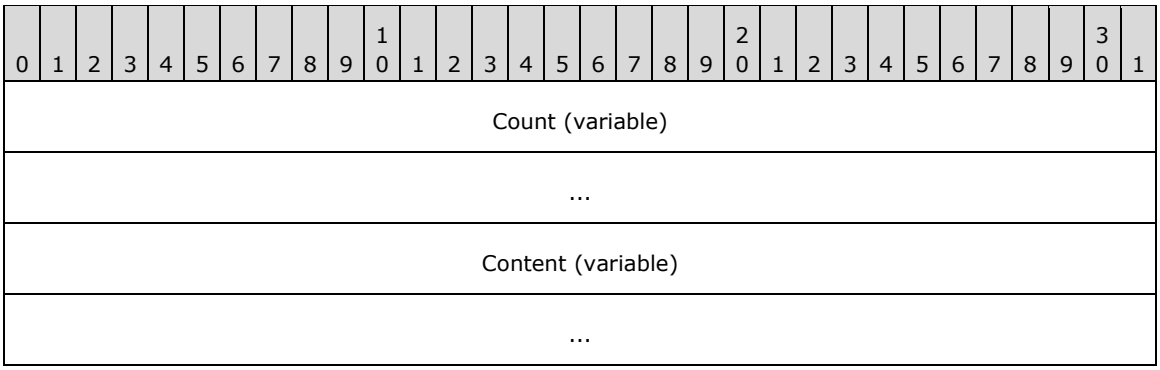


Length (variable): A compact unsigned 64-bit integer (section 2.2.1.1) that specifies the count of bytes of **Content** of the item.

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.



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

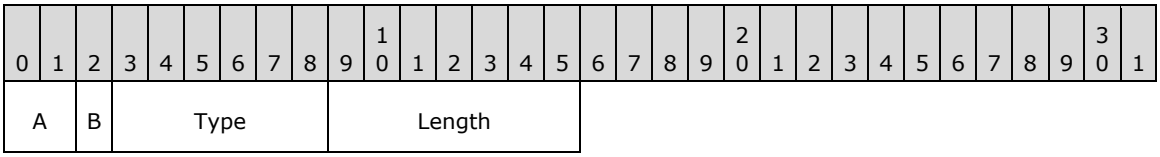
Content (variable): An array of **UTF-16** characters that specify the string. It MUST NOT be null-terminated.

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.

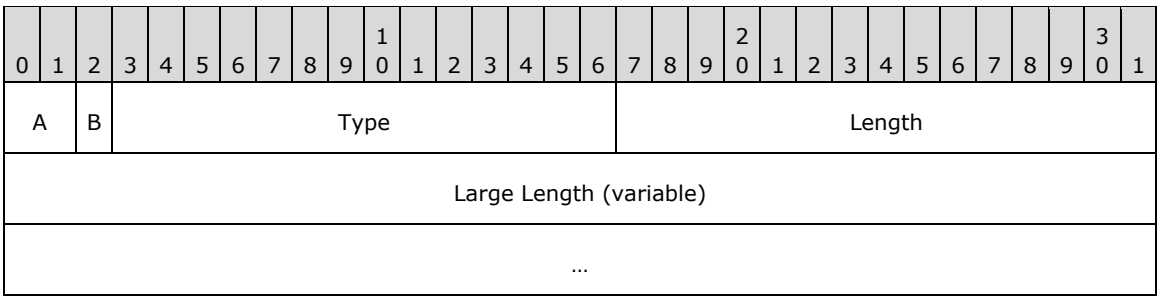
The following table lists the possible stream object types, and the corresponding **Compound** value:

Stream object type	Value	Compound
Data Element	0x01	1
Object Data BLOB	0x02	0
Object Group Object Excluded Data	0x03	0
Waterline Knowledge Entry (section 2.2.1.13.4.1)	0x04	0
Object Group Object Data BLOB Declaration	0x05	0
Data Element Hash	0x06	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

Stream object type	Value	Compound
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.



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 the 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.

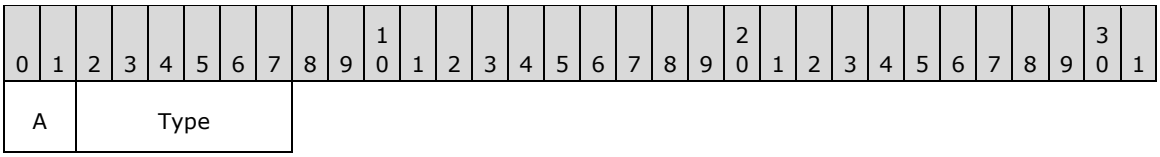
The following table lists the possible stream object types, and the corresponding **Compound** value.

Stream object type	Value	Compound
Request	0x040	1
Sub-response	0x041	1
Sub-request	0x042	1
Read access response	0x043	1

Stream object type	Value	Compound
Specialized Knowledge	0x044	1
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
Error HRESULT	0x052	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 Extended GUID Range request (section 2.2.2.1.5)	0x080	0
Allocate Extended GUID Range response (section 2.2.3.1.4)	0x081	0
Target Partition Id (section 2.2.2.1.1)	0x83	0
Put Changes Lock Id (section 2.2.2.1.4.2)	0x85	0
Additional Flags (section 2.2.2.1.4.1)	0x86	0
Put Changes Response	0x87	0
Request hashing options	0x88	0
Diagnostic Request Option Output (section 2.2.3.1.3.1)	0x89	0
Diagnostic Request Option Input (section 2.2.2.1.4.3)	0x8A	0
User Agent Client and Platform	0x8B	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.

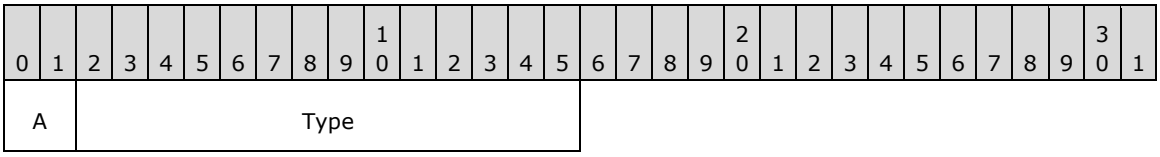
Type (6-bits): A 6-bit unsigned integer that specifies the stream object type.

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
Read access response	0x043
Specialized	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

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.

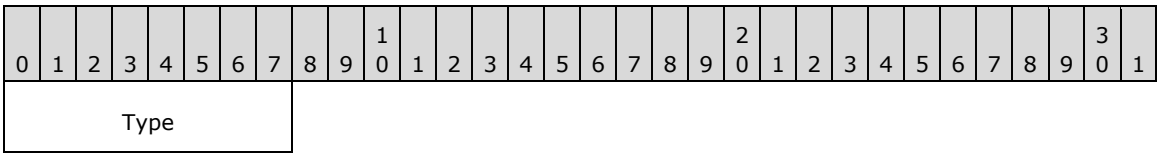
Value	Meaning
1	Query Access
2	Query Changes
5	Put Changes
11	Allocate Extended GUID Range

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

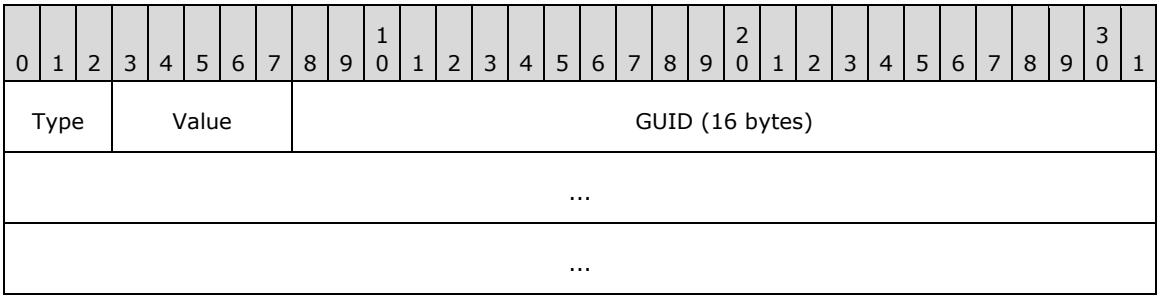
A 1-byte encoding of the **Extended GUID** when the GUID part is {00000000-0000-0000-0000-000000000000} and the unsigned integer part is zero.



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.



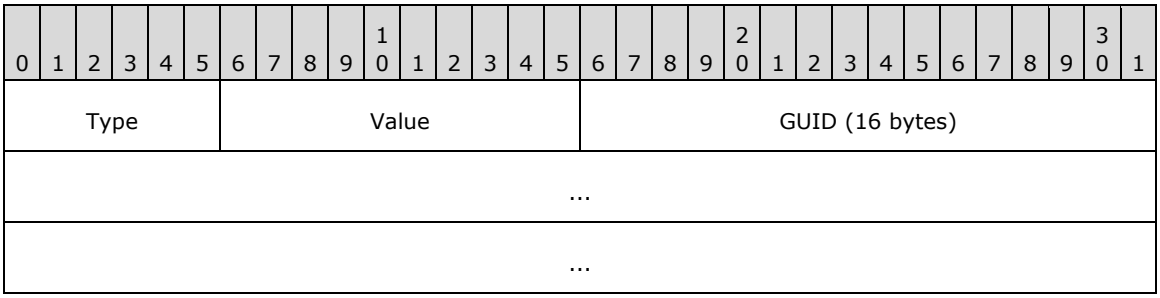
Type (3 bits): An unsigned integer that specifies the type that MUST be 4.

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.



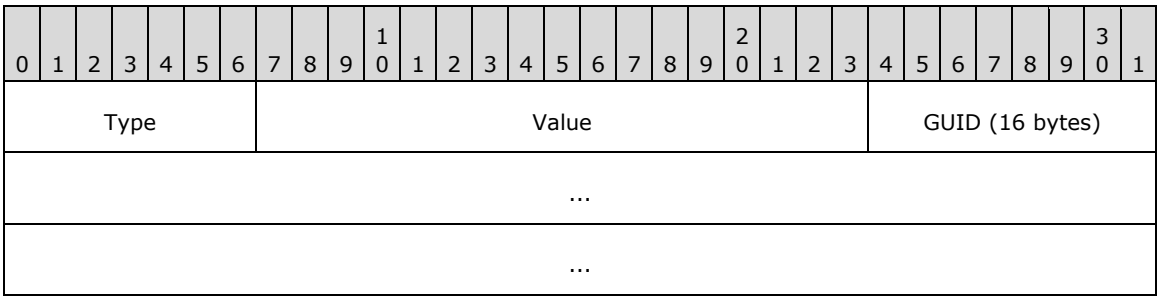
Type (6 bits): An unsigned integer that specifies the type that MUST be 32.

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.



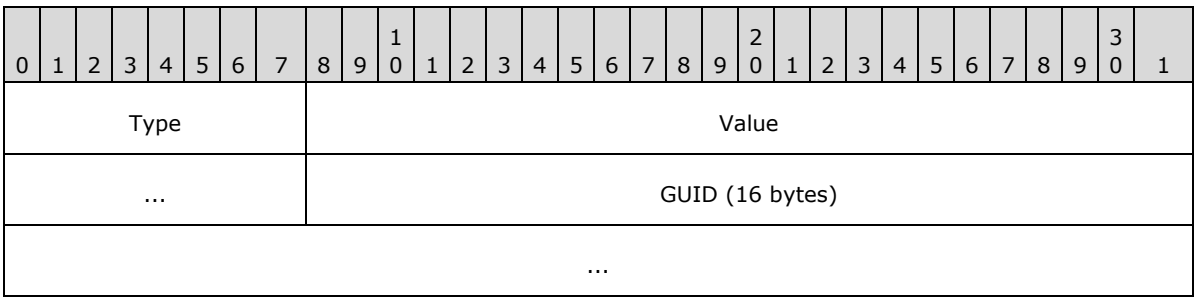
Type (7 bits): An unsigned integer that specifies the type that MUST be 64.

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.



...

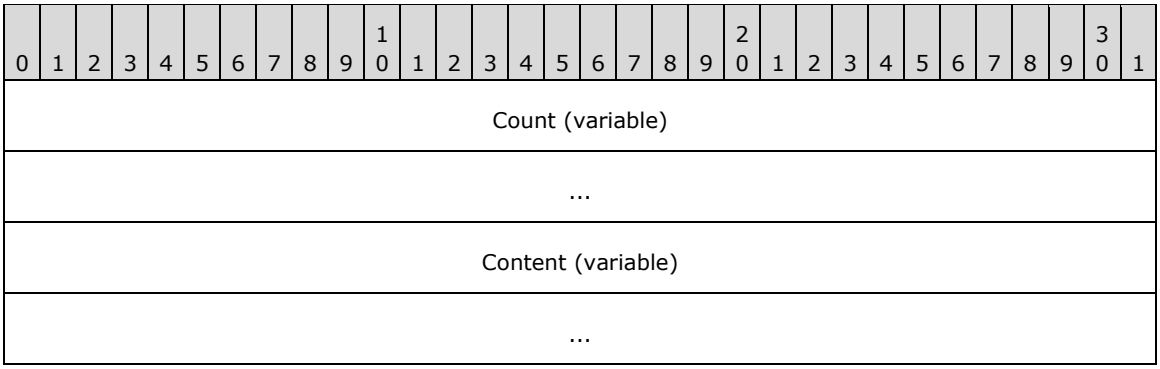
Type (1 byte): An unsigned integer that specifies the type that MUST be 128.

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.

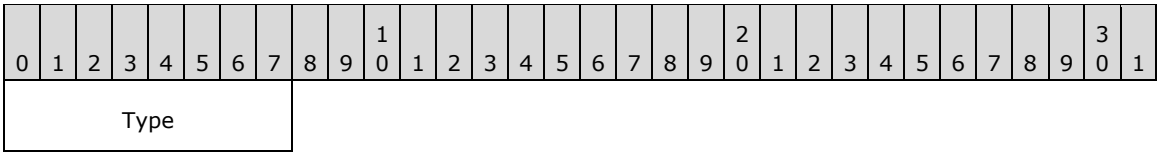
Content (variable): An **Extended GUID Array** that specifies an array of items.

2.2.1.9 Serial Number

A variable-width encoding that specifies a combination of a GUID and an unsigned 64-bit integer. The **Serial Number** of a data element can be created by the creator of the data element, but the server is authoritative and can replace a serial number that is created by the client. The server will return a **Cell Knowledge Range** that specifies the range of serial numbers, as specified in section 2.2.1.13.2.1. The following formats are used to encode non-overlapping ranges of **Serial Numbers**.

2.2.1.9.1 Serial Number Null Value

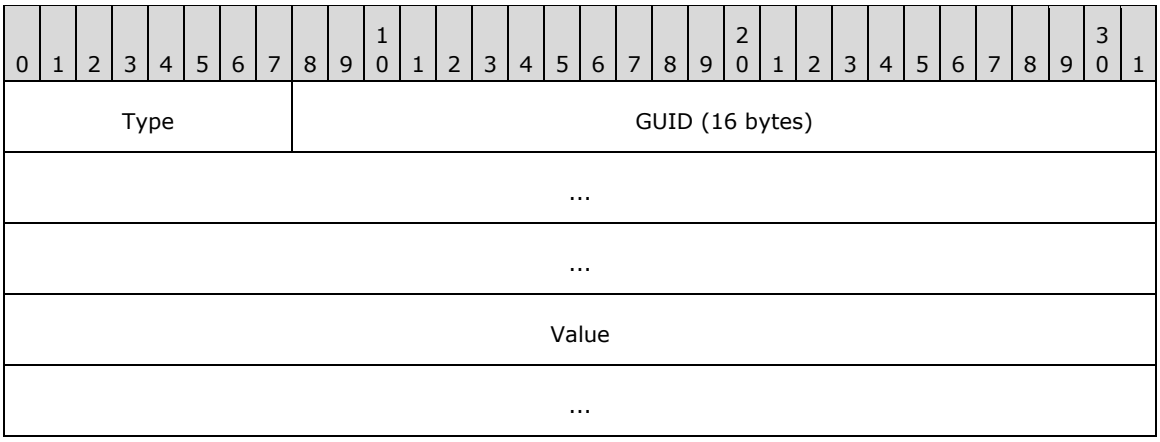
A 1-byte encoding of the **Serial Number** when the GUID part is {00000000-0000-0000-0000-000000000000} and the unsigned integer part is zero.



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**.



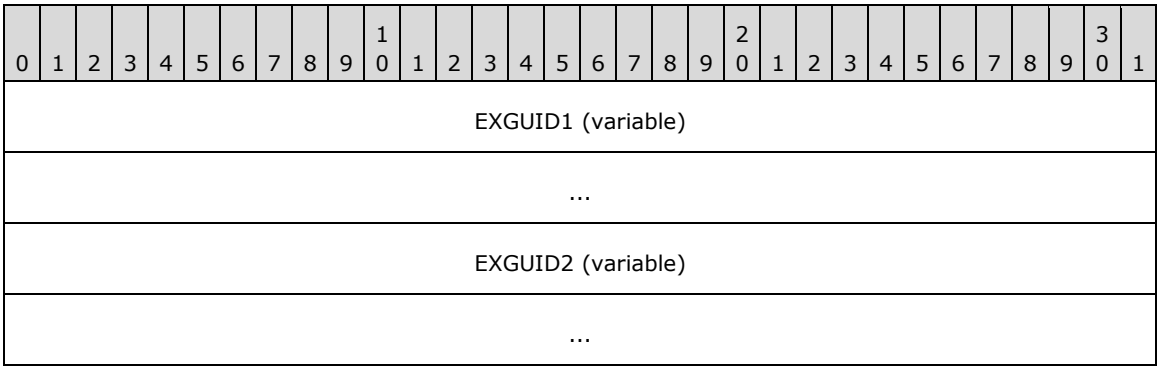
Type (1 byte): An unsigned integer that specifies the type, and MUST be 128.

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.

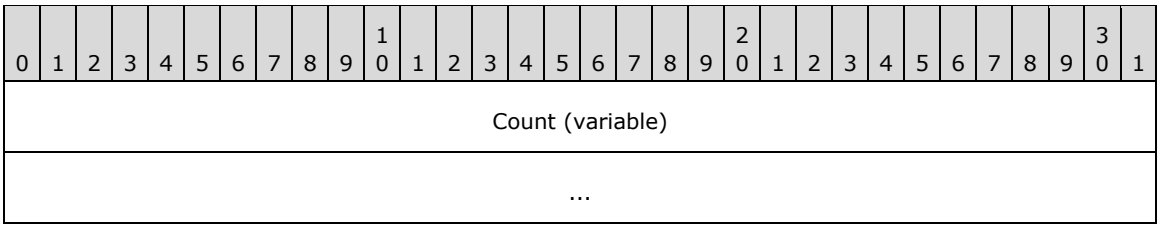


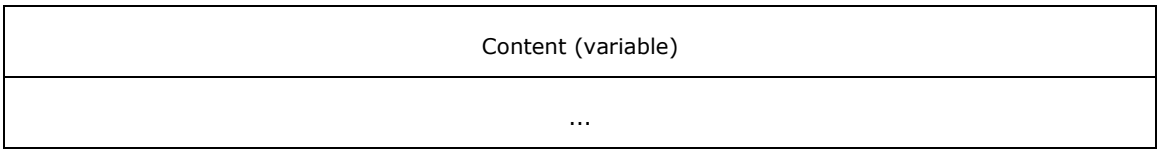
EXGUID1 (variable): An **Extended GUID** that specifies the first cell identifier.

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.



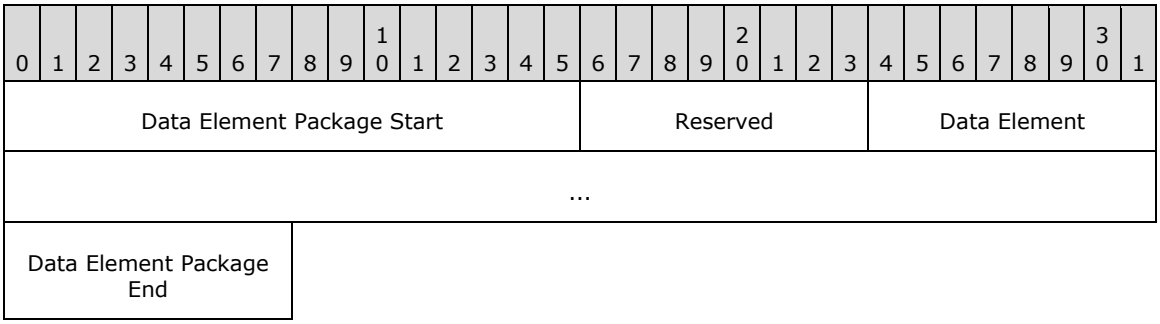


Count (variable): A compact unsigned 64-bit integer (section 2.2.1.1) that specifies the count of **Cell IDs** in the array.

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 that contains the serialized file data elements. If the client doesn't have any data elements, this field MUST NOT be present. All data elements are immutable. The types of data elements are listed in section 2.2.1.12.1.

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. All data elements are immutable.

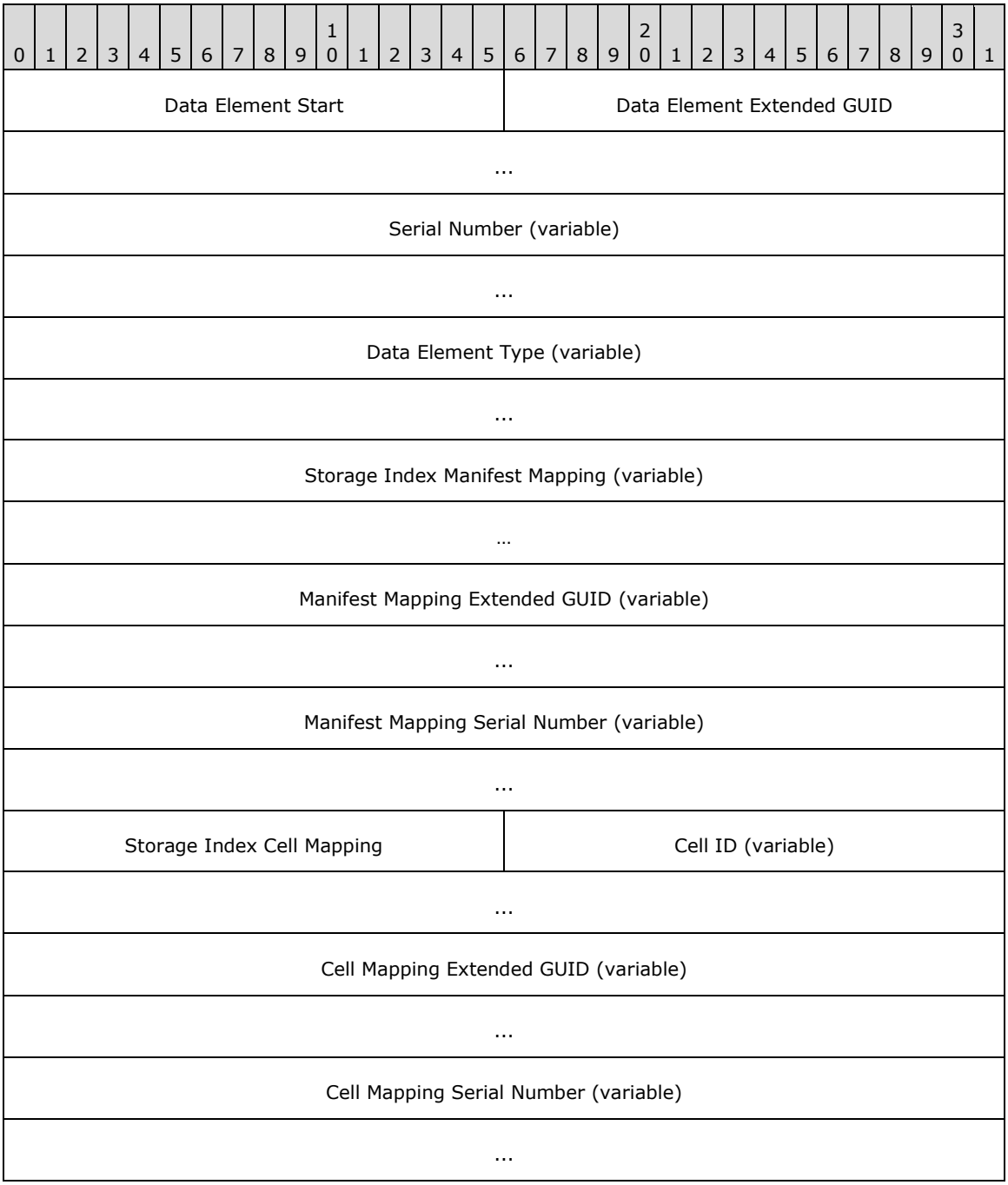
Data element type	Value
Storage Index (section 2.2.1.12.2)	0x01
Storage Manifest (section 2.2.1.12.3)	0x02
Cell Manifest (section 2.2.1.12.4)	0x03
Revision Manifest (section 2.2.1.12.5)	0x04
Object Group (section 2.2.1.12.6)	0x05
Data Element Fragment (section 2.2.1.12.7)	0x06
Object Data BLOB (section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

Additionally, the Storage Index contains a set of mappings, and each mapping is assigned a serial number that is unique to that mapping. The client SHOULD add these mapping serial numbers to its serial number knowledge, just as it SHOULD add data element serial numbers to the serial number knowledge.



Storage Index Revision Mapping	Revision Mapping Extended GUID
Revision Mapping Extended GUID (variable)	
...	
Revision Mapping Serial Number (variable)	
...	
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 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 Headers** that specify 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.

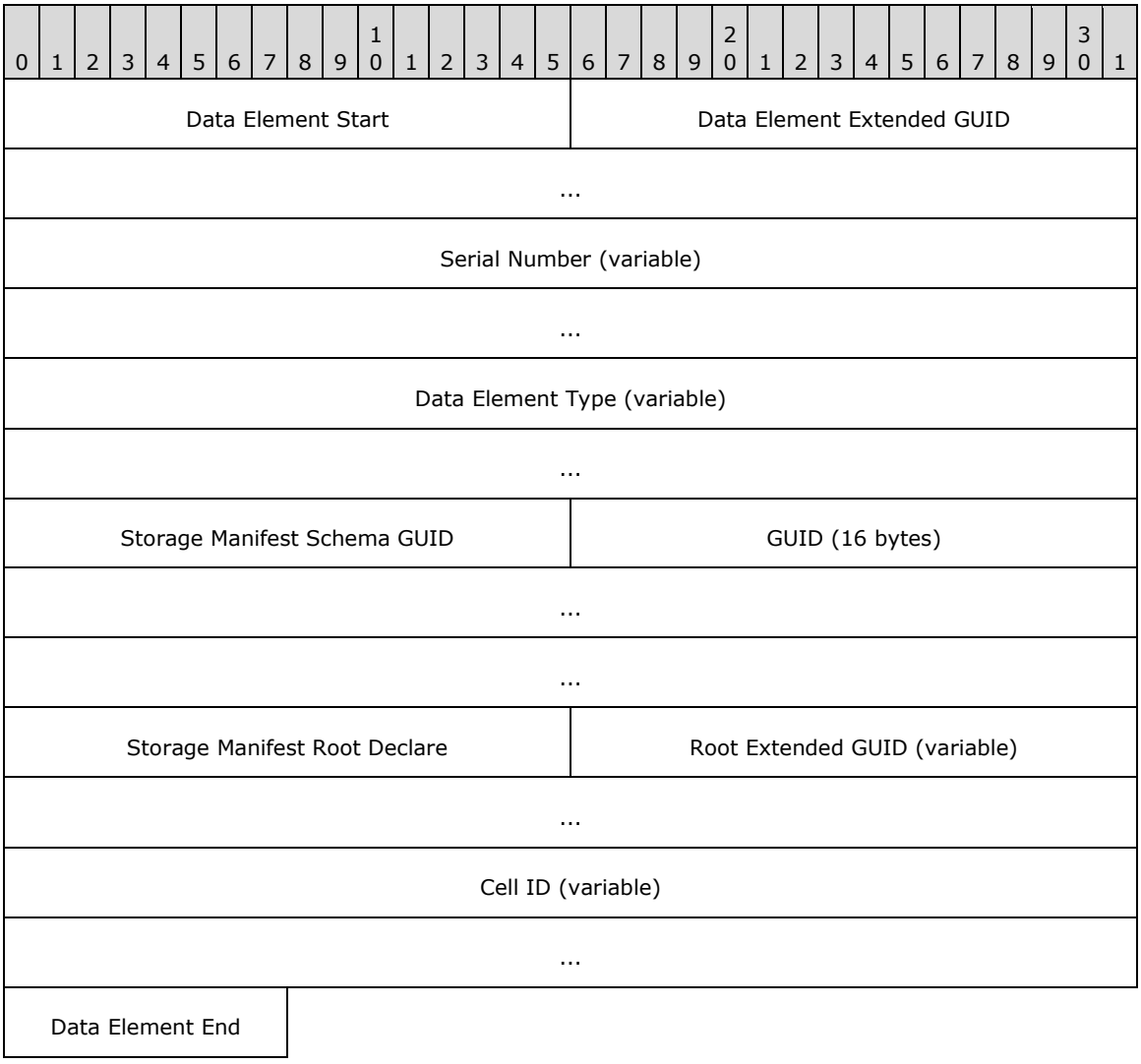
Revision Mapping Extended GUID (variable): An **Extended GUID** that specifies the Revision Mapping.

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 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.

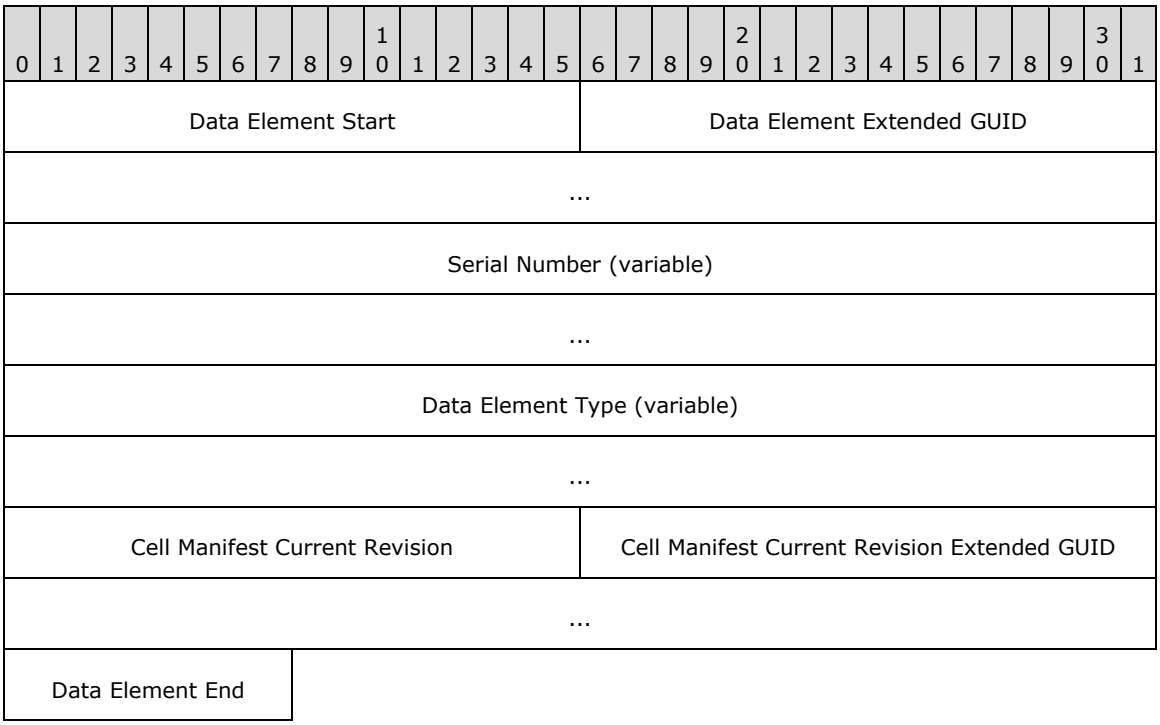
Root Extended GUID (variable): An **Extended GUID** that specifies the root Storage Manifest.

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 Cell Manifest 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 Cell Manifest data element type.

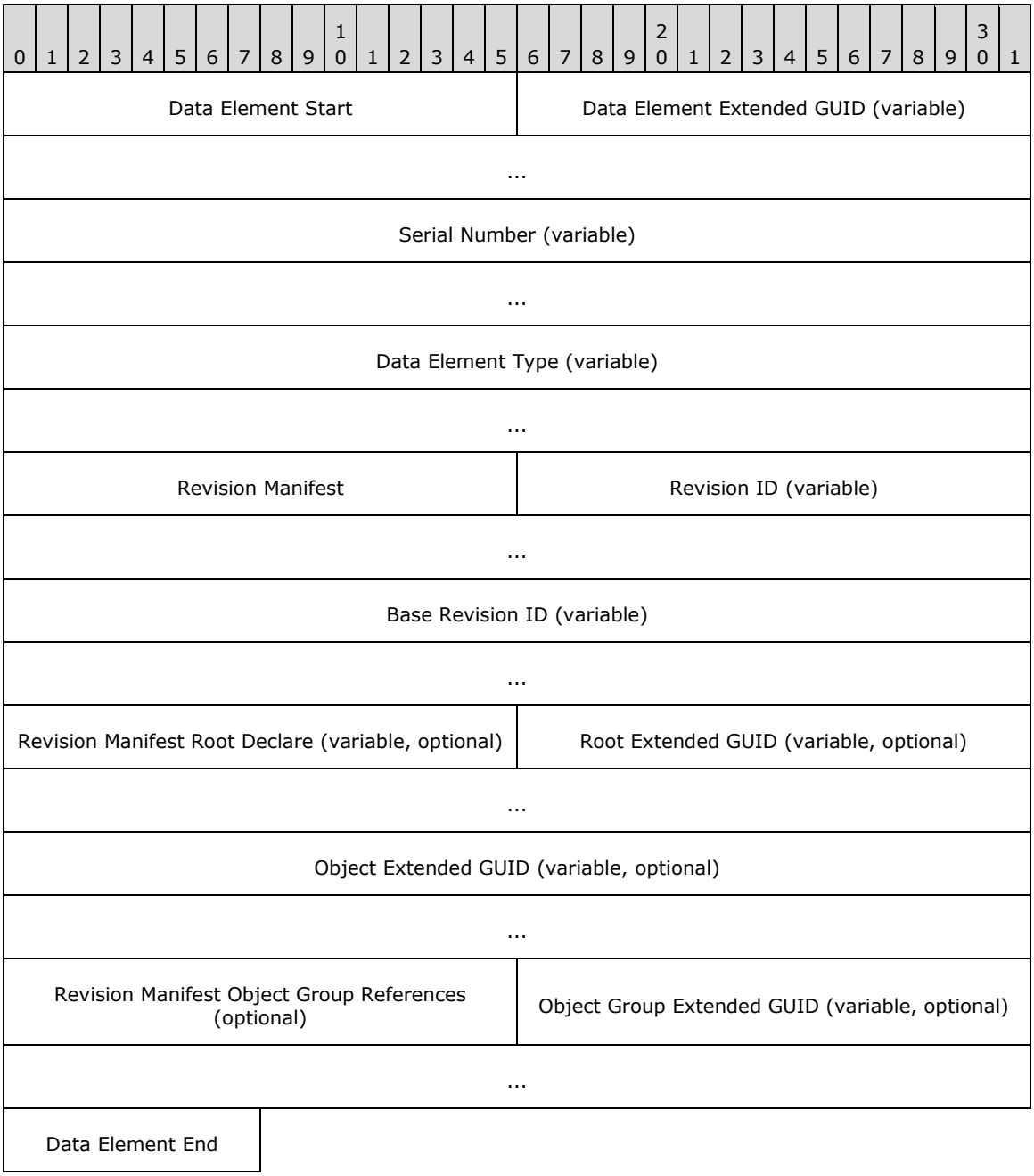
Cell Manifest Current Revision (2 bytes): A 16-bit **Stream Object Header** that specifies a Cell Manifest current revision.

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.



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**.

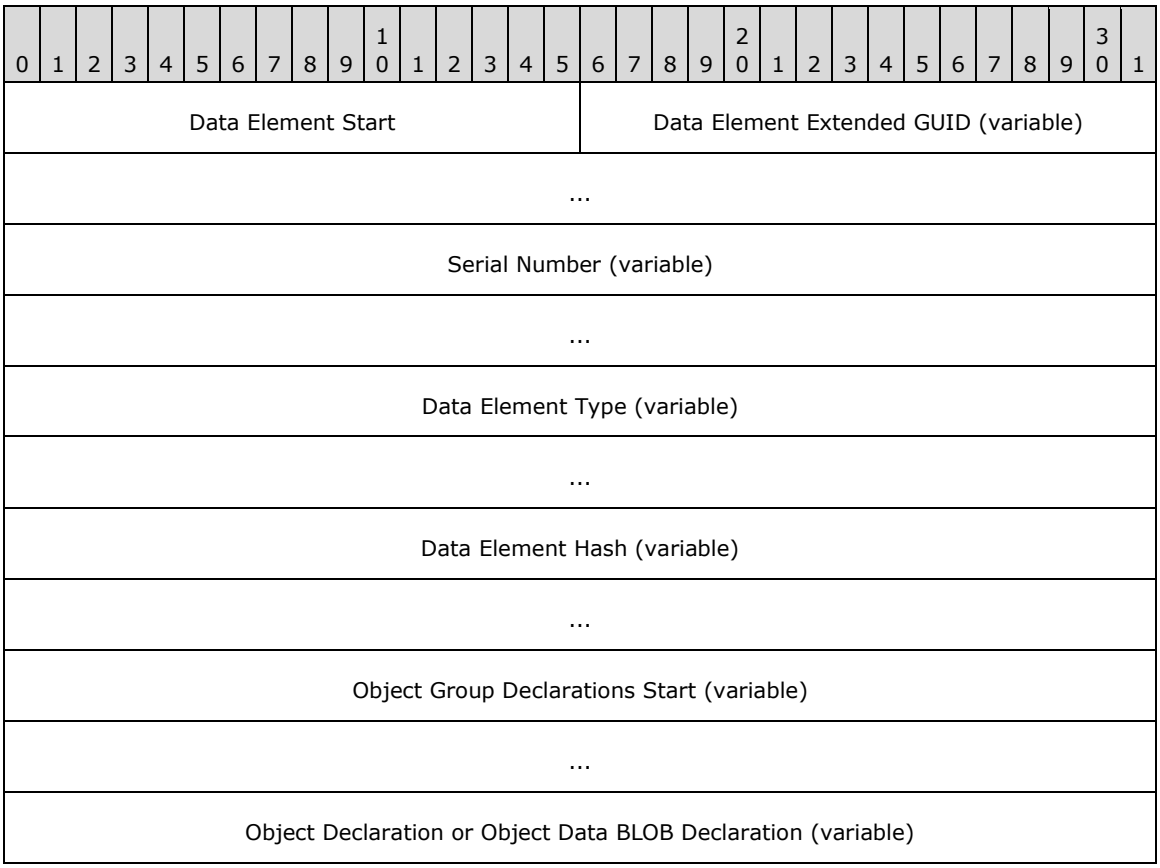
Revision Manifest Object Group References (2 bytes, optional): Zero or more 16-bit **Stream Object Header** that specify Revision Manifest Object Group references, each followed by Object Group **Extended GUIDs**.

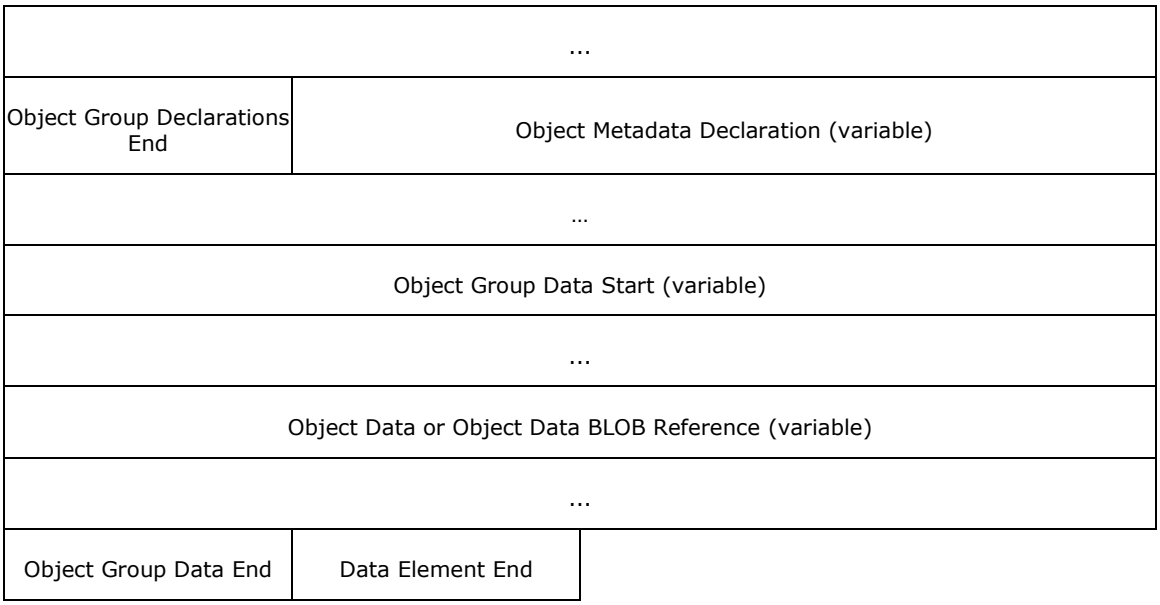
Object Group Extended GUID (variable): An **Extended GUID** that specifies the Object Group for each **Revision Manifest Object Group Reference**.

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** (section 2.2.1.12.6.6) that specifies the value to be used when fetching/injecting Object Data (section 2.2.1.12.6.4) 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 reference (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.

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 Declaration (variable)																															
...																															
Object Extended GUID (variable)																															
...																															
Object Partition ID (variable)																															
...																															
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.

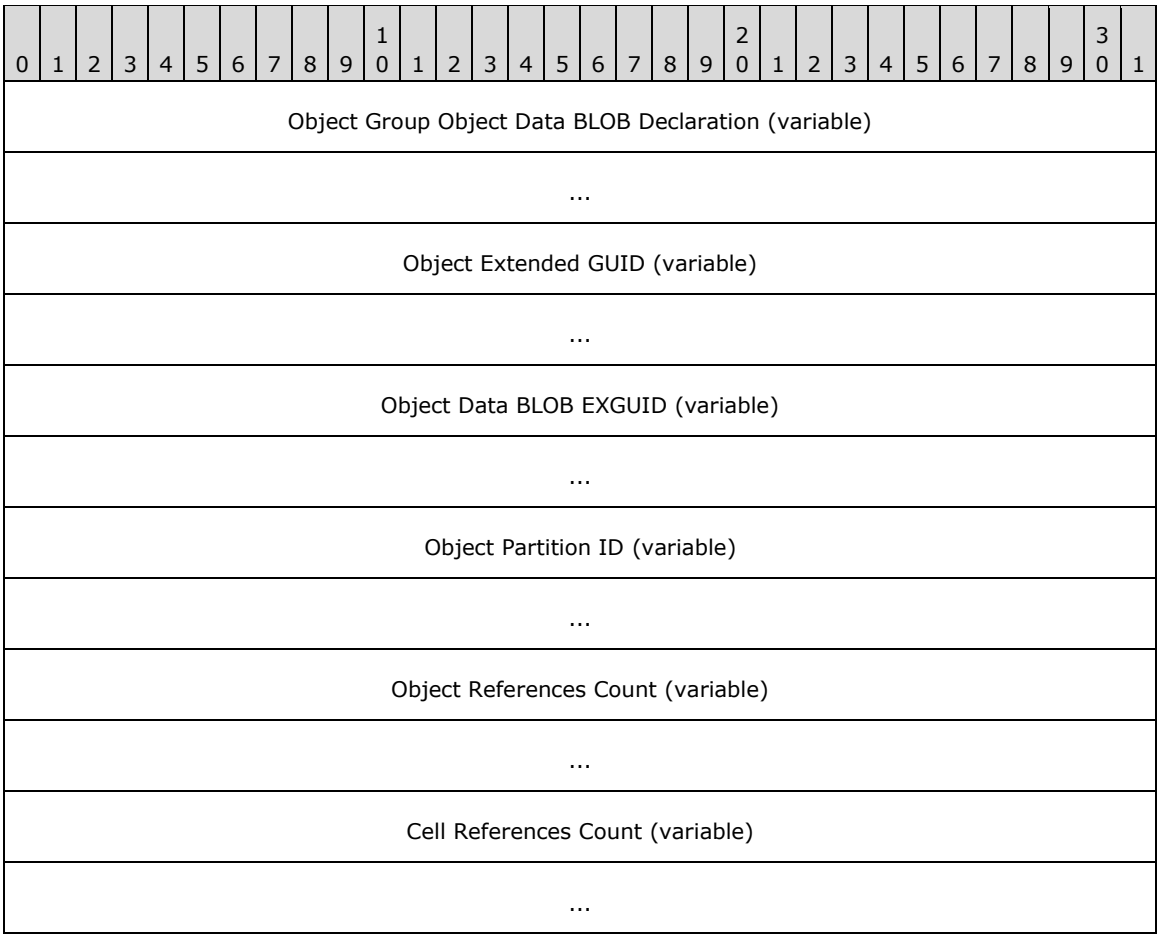
Object Data Size (variable): A compact unsigned 64-bit integer that specifies the size in bytes of the binary data opaque to this protocol for the declared object. This SHOULD match the size of the **Binary Item** (section 2.2.1.3) in the corresponding **Object Data** (section 2.2.1.12.6.4) for this object.<1>

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.



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 (section 2.2.1.12.6.2).

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**.

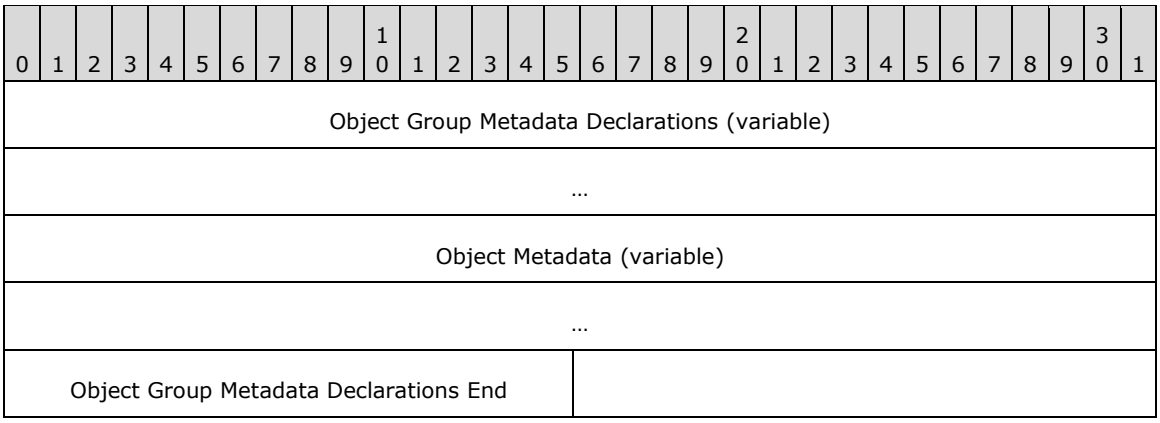
Object Partition ID (variable): A compact unsigned 64-bit integer (section 2.2.1.1) that specifies the object partition of the object.

Object 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.<2>



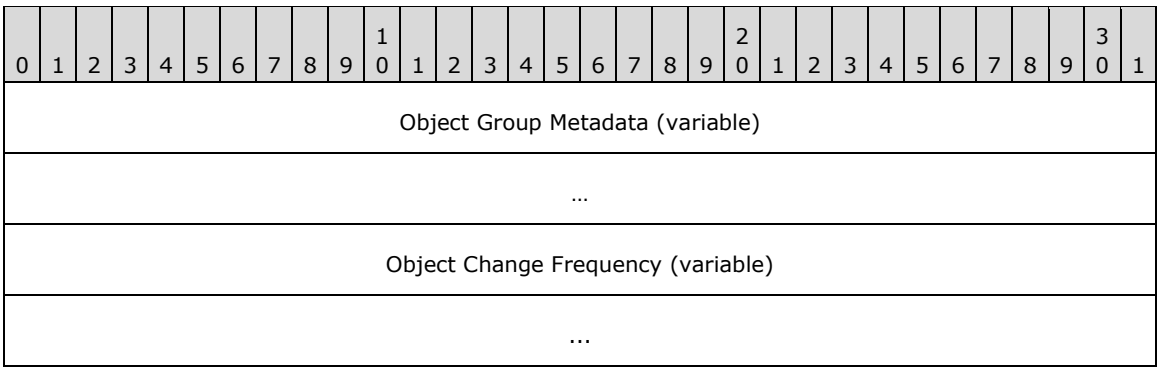
Object Group Metadata Declarations (variable): A 32-bit **Stream Object Header** section 2.2.1.5.2) that specifies an Object Group metadata declarations.

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.



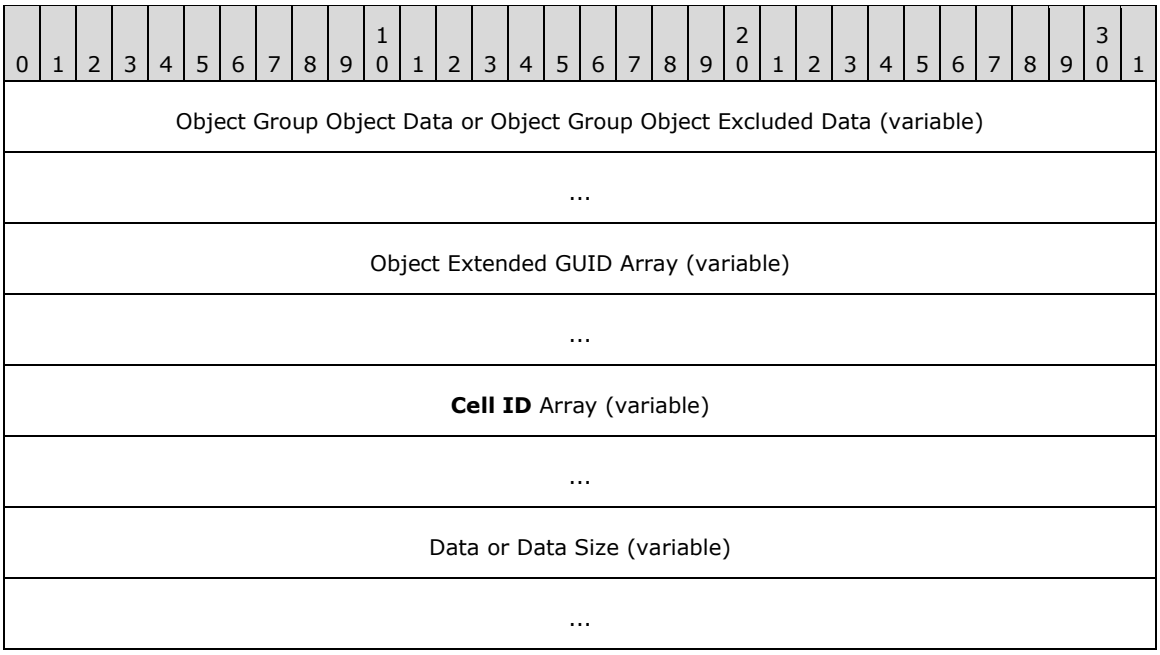
Object Group Metadata (variable): A 32-bit **Stream Object Header** (section 2.2.1.5.2) that specifies an Object Group metadata.

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.



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.

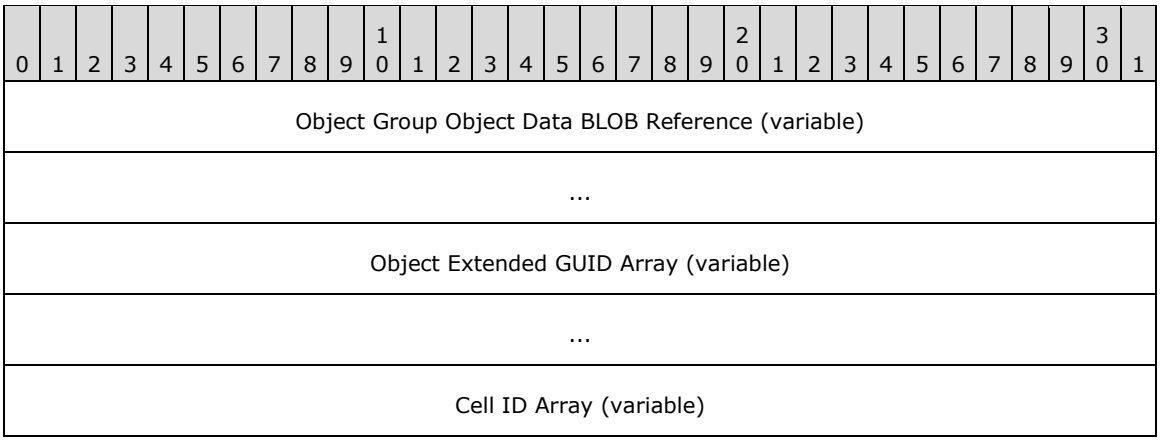
Object Extended GUID Array (variable): An **Extended GUID Array** (section 2.2.1.8) that specifies the Object Group.

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.



...
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).

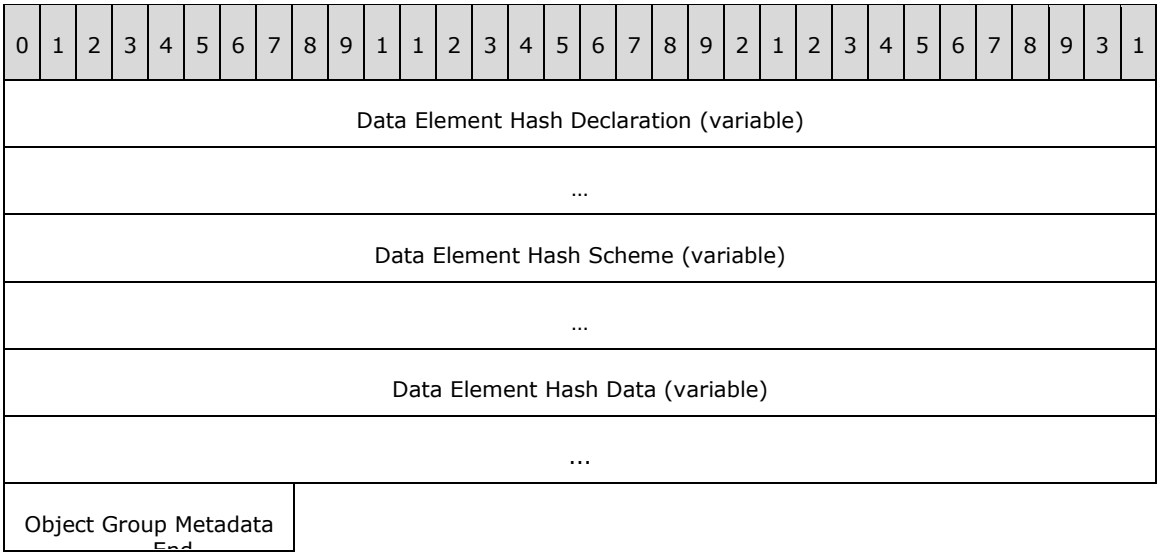
Object Extended GUID Array (variable): An **Extended GUID Array** (section 2.2.1.8) that specifies the object references.

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

A **Data Element Hash** has the following structure.



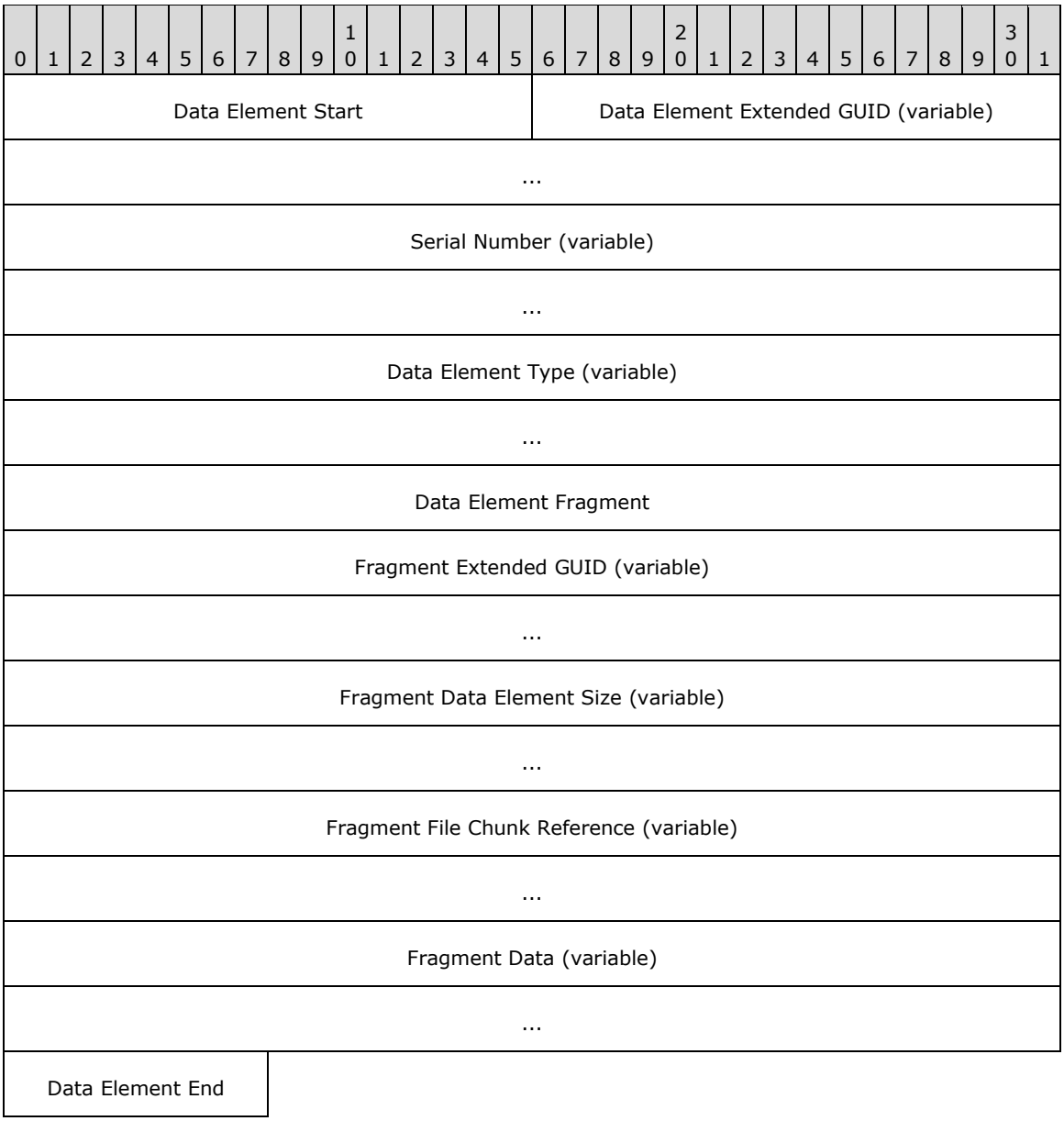
Data Element Hash Declaration (variable): A 16-bit (section 2.2.1.5.1) or 32-bit **Stream Object Header** section 2.2.1.5.2) that specifies a data element hash declaration.

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.

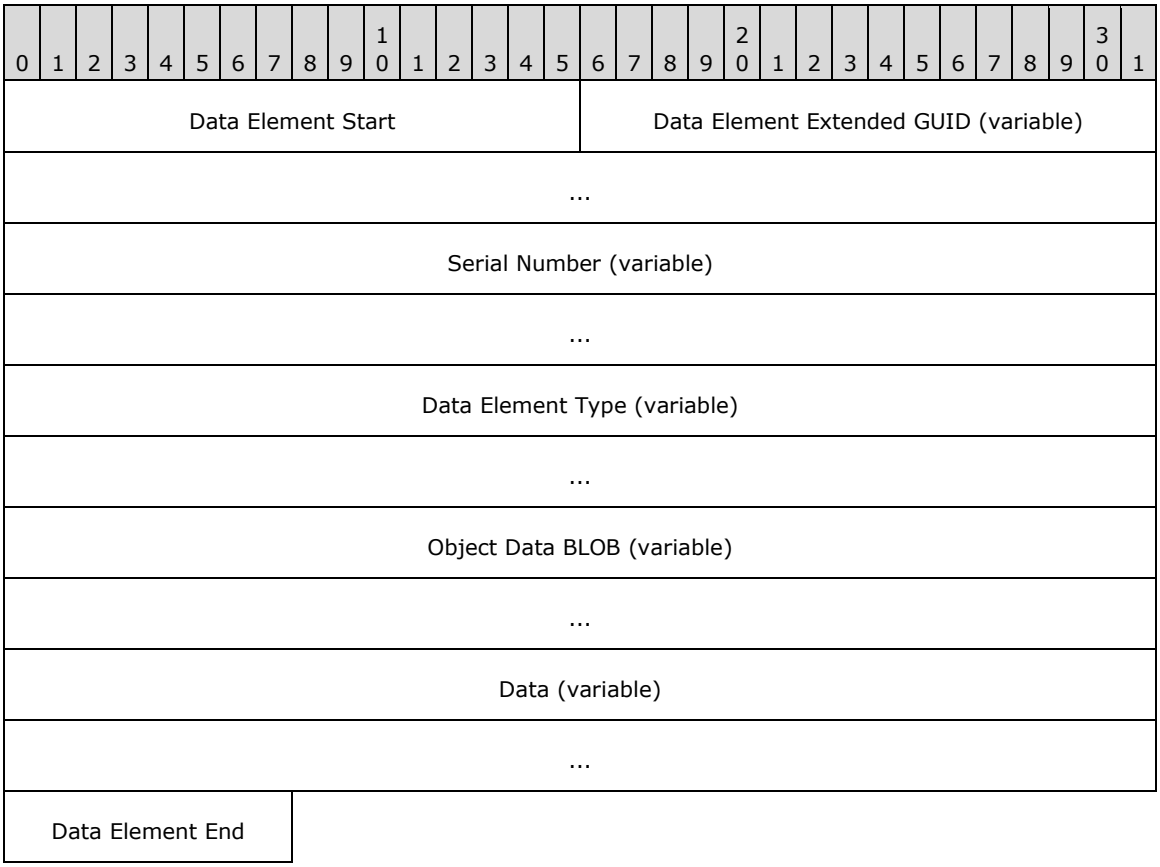
Fragment File Chunk Reference (variable): A **File Chunk Reference** (section 2.2.1.2) that specifies the Data Element Fragment.

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.

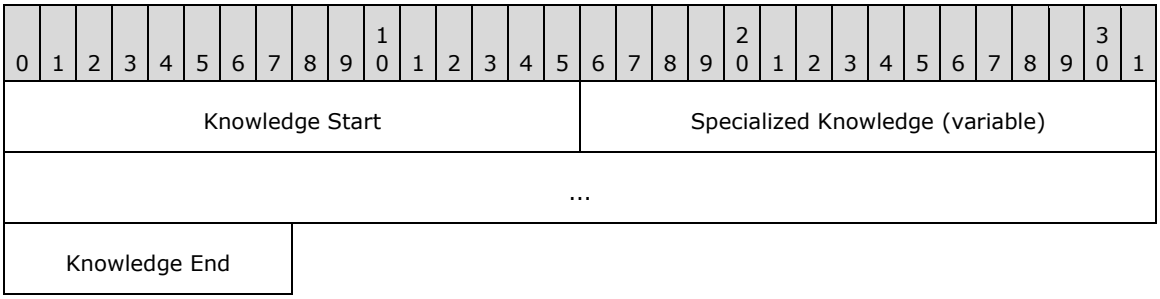
Object Data BLOB (variable): A 16-bit or 32-bit **Stream Object Header** that specifies an Object Data BLOB.

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.



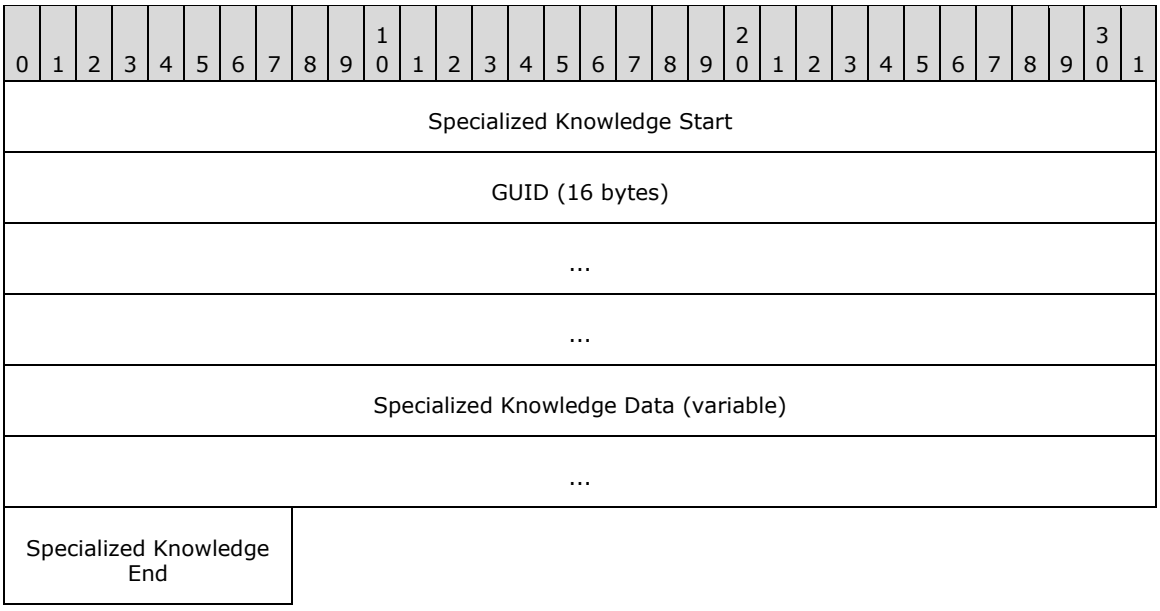
Knowledge Start (2 bytes): A 16-bit **Stream Object Header** (section 2.2.1.5.1) that specifies a **Knowledge** (section 2.2.1.13) start.

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:

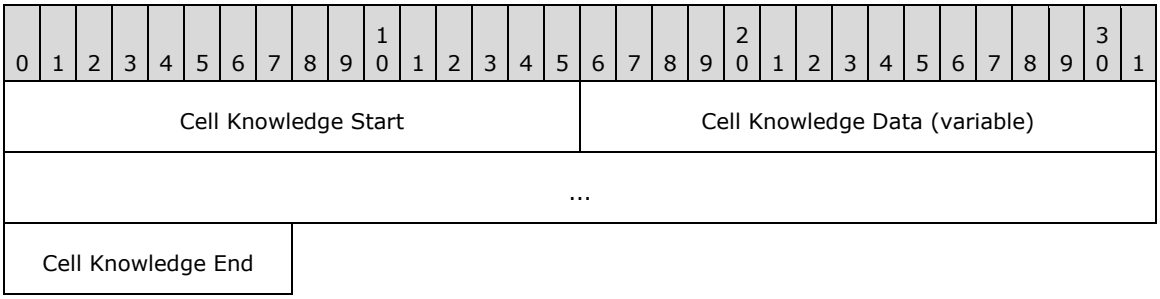
GUID (string representation)	Knowledge type
{327A35F6-0761-4414-9686-51E900667A4D}	Cell Knowledge (section 2.2.1.13.2)
{3A76E90E-8032-4D0C-B9DD-F3C65029433E}	Waterline Knowledge (section 2.2.1.13.4)
{0ABE4F35-01DF-4134-A24A-7C79F0859844}	Fragment Knowledge (section 2.2.1.13.3)
{10091F13-C882-40FB-9886-6533F934C21D}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



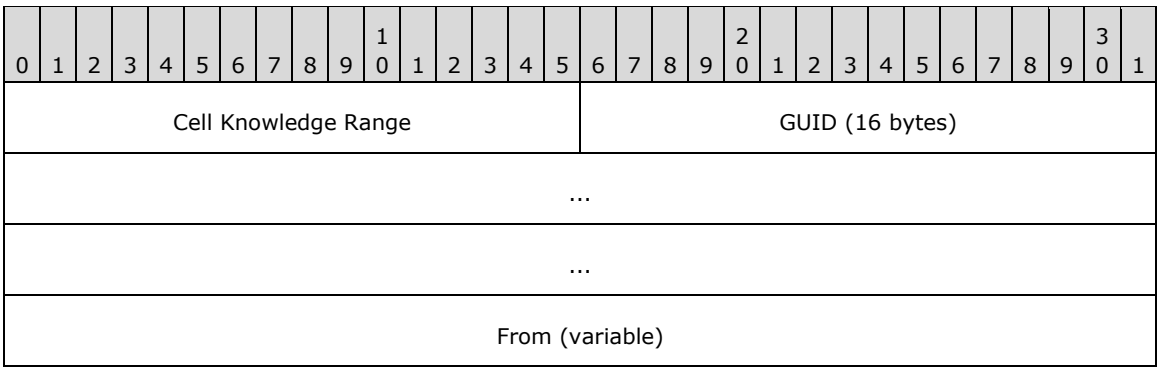
Cell Knowledge Start (2 bytes): A 16-bit **Stream Object Header** (section 2.2.1.5.1) that specifies a **Cell Knowledge** start.

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.



...
To (variable)
...

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.

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
Cell Knowledge Entry																Serial Number (variable)															
...																															

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.

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
Fragment Knowledge Start																															
Fragment Knowledge Entries (variable)																															
...																															
Fragment Knowledge End																															

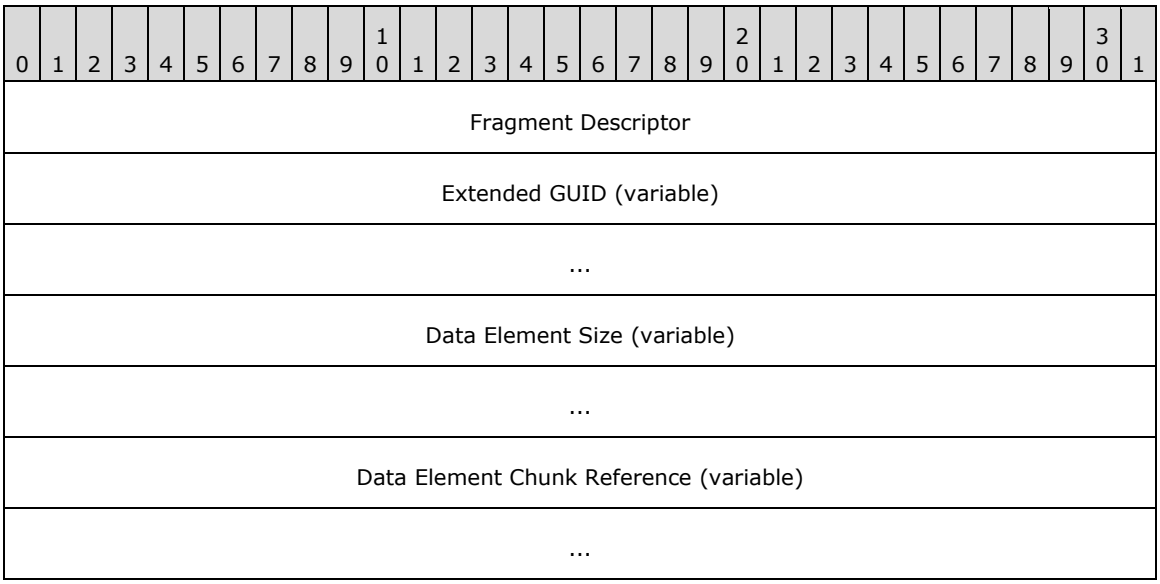
Fragment Knowledge Start (4 bytes): A 32-bit **Stream Object Header** (section 2.2.1.5.2) that specifies a **Fragment Knowledge** start.

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.

Waterline Knowledge is intended to be used for server-specific optimizations. Clients MUST NOT interpret the values as they are not defined by this protocol.

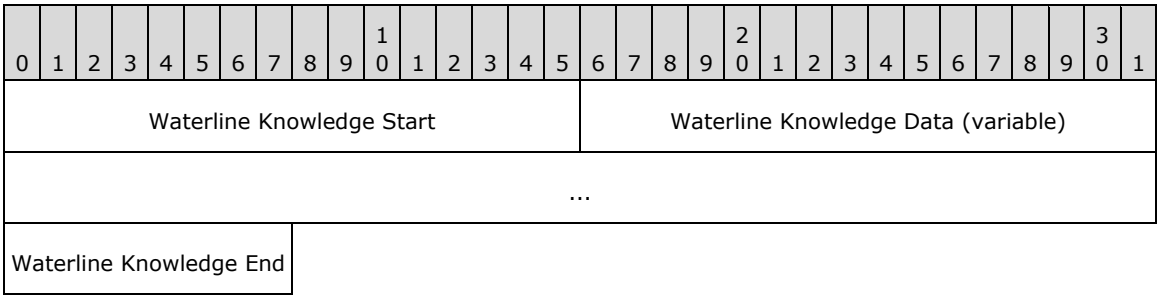
Client considerations:

- That these properties MUST NOT be modified by the client.
- These values SHOULD be considered opaque from the client's perspective and the client SHOULD NOT attempt to interpret the values.
- They MUST be sent back to the server unmodified.

Server considerations.

- Server implementations do not need to replicate the same logic and behavior as SharePoint when using these properties to decide which cells to send to the client.
- Because the values are opaque to the client, a server implementation can use these values as needed.

It has the following format.



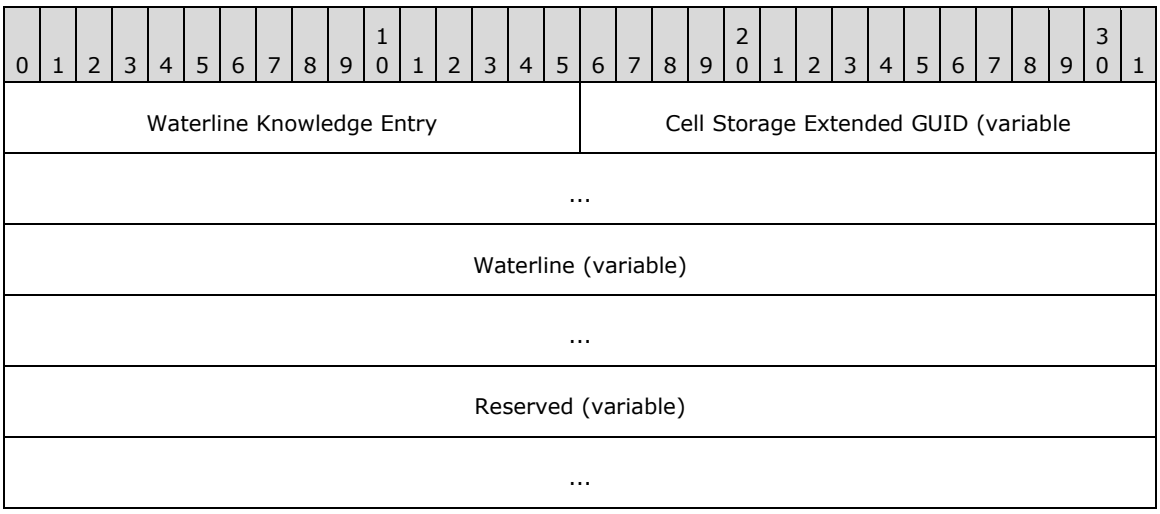
Waterline Knowledge Start (2 bytes): A 16-bit **Stream Object Header** (section 2.2.1.5.1) that specifies a **Waterline Knowledge** start.

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.



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 changes to implementer-defined content that is stored on the server. The **Content Tag Knowledge** has the following format.

Content Tag Knowledge is intended to be used for server-specific optimizations. Clients MUST NOT interpret the values as they are not defined by this protocol.

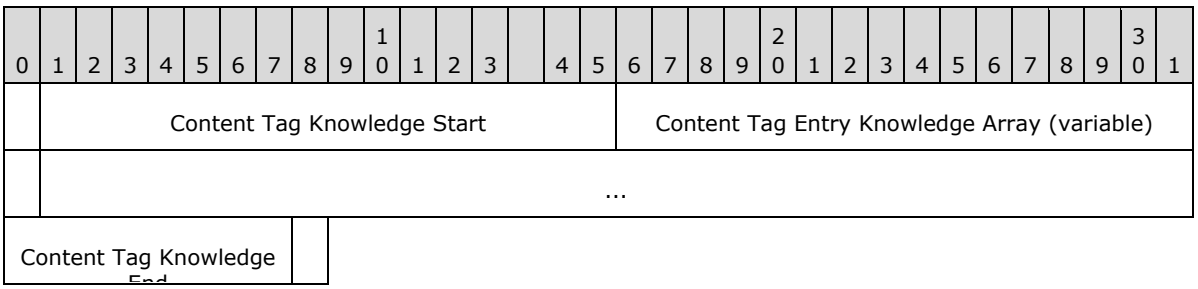
Client considerations:

- That these properties MUST NOT be modified by the client.
- These values SHOULD be considered opaque from the client's perspective and the client SHOULD NOT attempt to interpret the values.
- They MUST be sent back to the server unmodified.

Server considerations.

- Server implementations do not need to replicate the same logic and behavior as SharePoint when using these properties to decide which cells to send to the client.
- Because the values are opaque to the client, a server implementation can use these values as needed.

It has the following format.



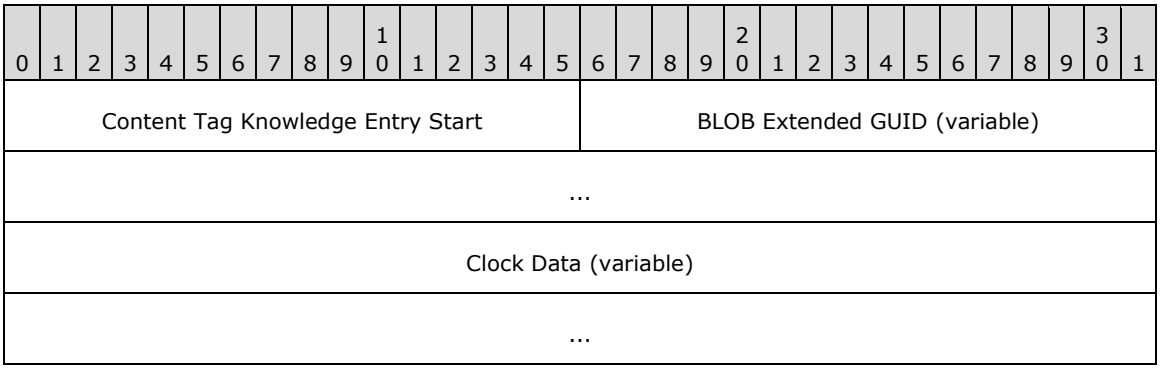
Content Tag Knowledge Start (2 bytes): A 16-bit **Stream Object Header** (section 2.2.1.5.1) that specifies the **Content Tag Knowledge** start.

Content Tag Knowledge Entry Array (variable): An array of **Content Tag Knowledge Entry** structures (section 2.2.1.13.5.1), each of which specifies changes for a BLOB.

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** specifies changes for a BLOB.



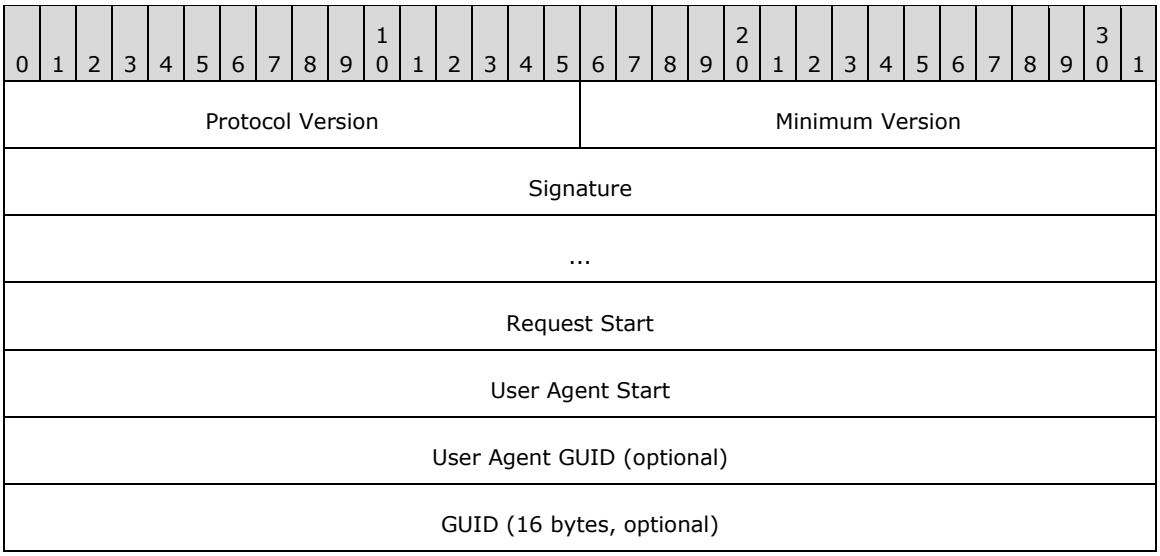
Content Tag Knowledge Entry Start (2 bytes): A 16-bit **Stream Object Header** (section 2.2.1.5.1) that specifies the start of a **Content Tag Knowledge Entry**.

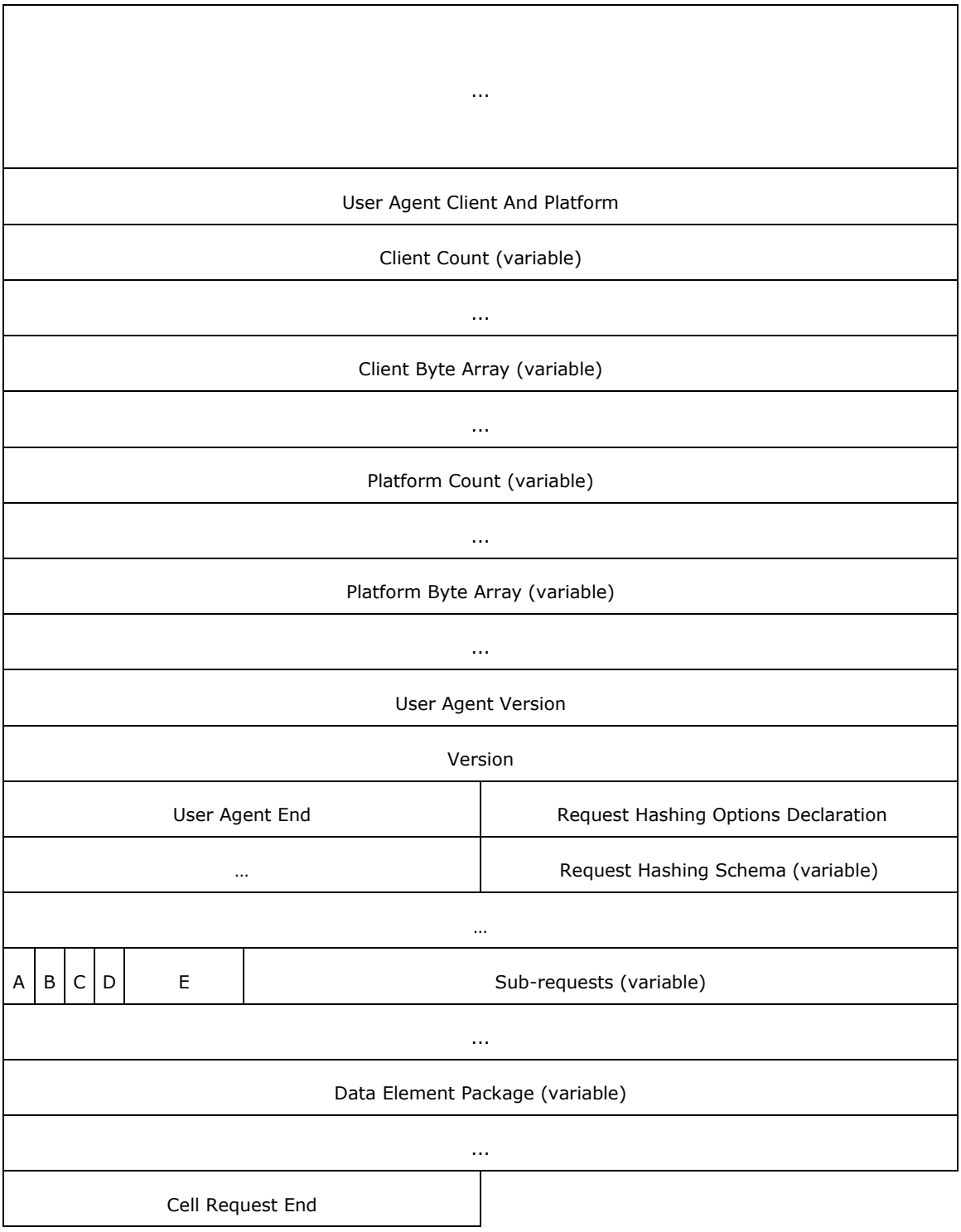
BLOB Extended GUID (variable): An **Extended GUID** (section 2.2.1.7) that specifies the BLOB this content tag is for.

Clock Data (variable): A **binary item** (section 2.2.1.3) that specifies changes for a BLOB on the server. This is implementation-specific on the server side; the client MUST set this to the same value as the server when it is included in a subsequent **Query Changes** (section 2.2.2.1.3) sub-request.

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.<3>





Protocol Version (2bytes): An unsigned integer that specifies the protocol schema version number used in this request. The valid values for this field are 12, 13 and 14.<4>

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, optional): A 32-bit **Stream Object Header** that specifies a user agent GUID.

GUID (16 bytes, optional): A GUID that specifies the user agent. This GUID **MUST** be specified if and only if the **User Agent GUID** is specified.

User Agent Client and Platform (4 bytes): A 32-bit **Stream Object Header** that specifies a user agent client and platform. This object replaces the **User Agent GUID**. If this is specified, the **User Agent GUID** SHOULD NOT be specified.

Client Count (variable): A compact unsigned 64-bit integer (section 2.2.1.1) that specifies the count of bytes in the **Client Byte Array**.

Client Byte Array (variable): An array of bytes that is a UTF-8 encoded string specifying the client name for the user agent.

Platform Count (variable): A compact unsigned 64-bit integer (section 2.2.1.1) that specifies the count of bytes in the **Platform Byte Array**.

Platform Byte Array (variable): An array of bytes that is a UTF-8 encoded string specifying the platform name for 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.

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: An optional 32-bit **Stream Object Header** that specifies a request hashing options declaration.<5>

Request Hashing Schema: An optional 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, optional): A reserved bit that MUST be set to zero and MUST be ignored.

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

C – Request Data Element Hashes Instead of Data (1 bit, optional): 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, optional): If set, a bit that specifies to include data element hashes (if available) when returning data elements; otherwise data element hashes will 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, optional): A 4-bit reserved field that MUST be set to zero and MUST be ignored.

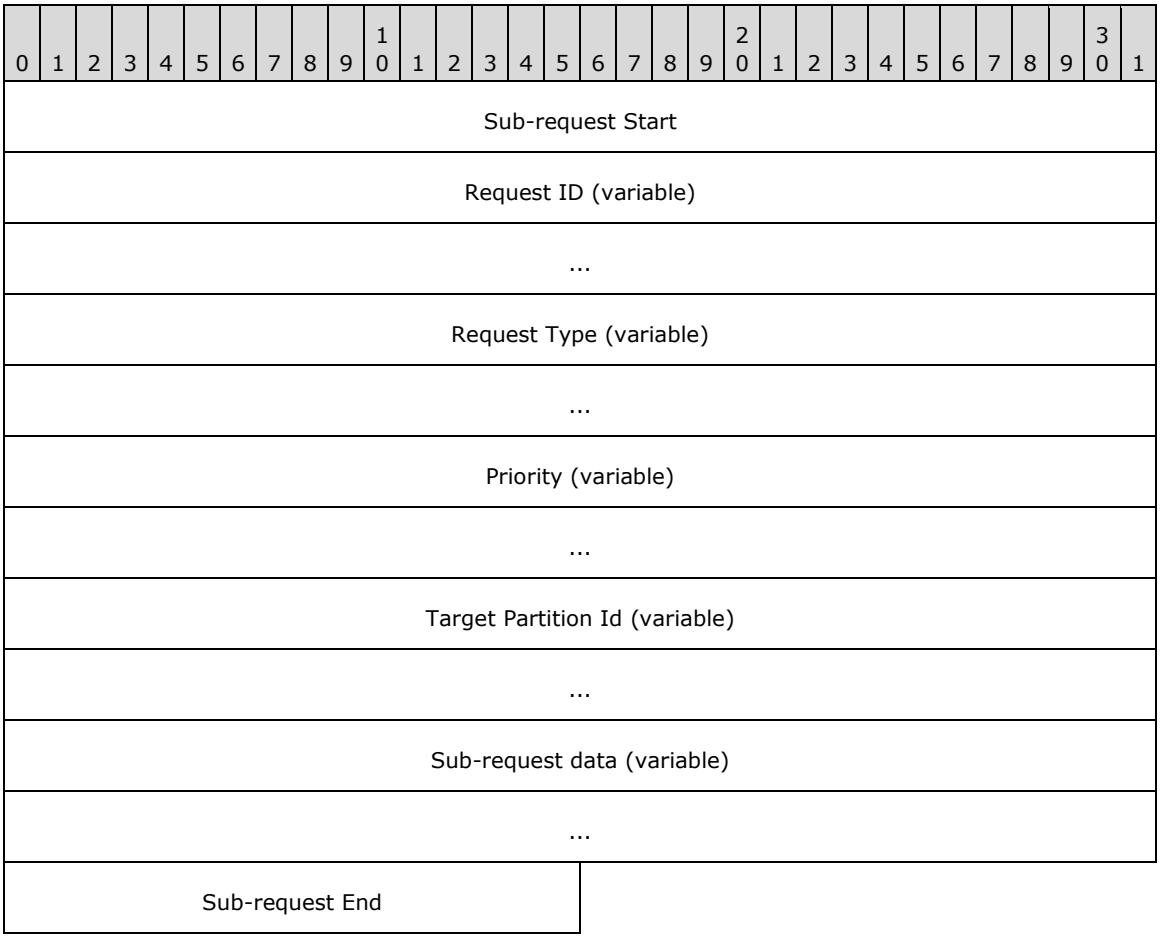
Sub-requests (variable): An array of sub-requests (section 2.2.2.1), that specifies request information to execute on the server.

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.<6>

Target Partition Id (variable): An optional **Target Partition Id** (section 2.2.2.1.1) that specifies the target partition for this sub-request.<7>

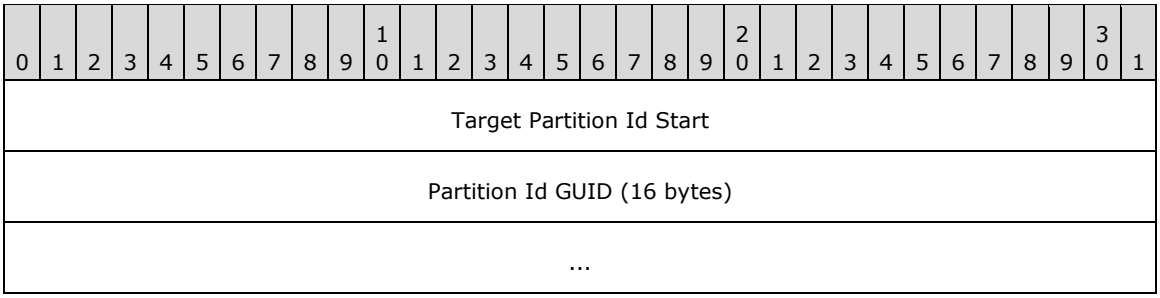
Sub-request data (variable): A structure that specifies additional data for this sub-request. The structure used depends on the value of the **Request Type**, as specified by the following table.

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**.

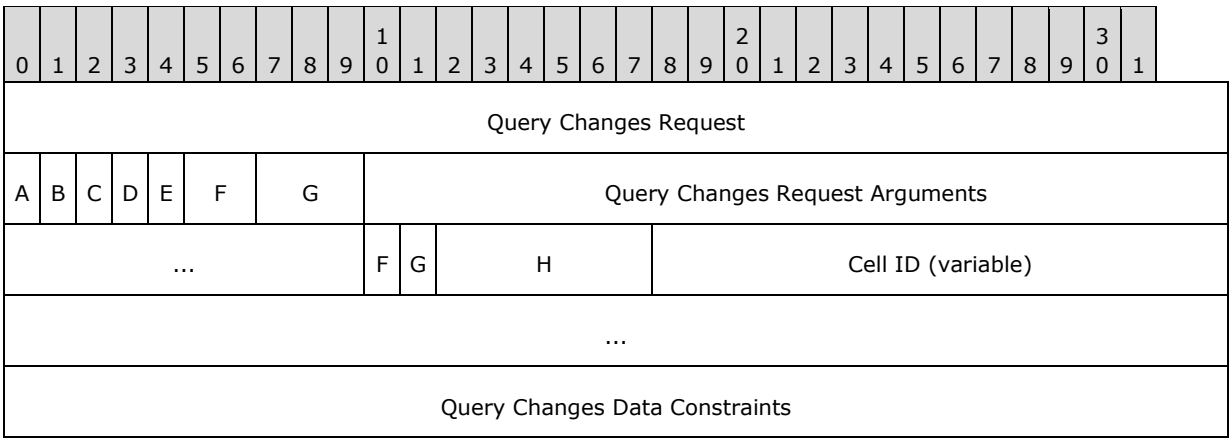
Partition Id GUID (16 bytes): A GUID that specifies the partition.

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.



Max Data Elements (variable)
...
Query Changes Versioning
Major Version Number (variable)
Minor Version Number (variable)
Query Changes Filters (variable)
...
Knowledge (variable)
...

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 ignored.

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

C – Exclude Object Data (1 bit): This attribute will be ignored by the server.

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** (section 2.2.1.13); otherwise, the **Serial Numbers** of filtered out data elements are not included in the response **Knowledge**.<8>

E – Allow Fragments 2 (1 bit): If set, a bit that specifies to allow fragments; otherwise, it does not allow fragments, unless the bit specified in B is set.

F – Round Knowledge to Whole Cell Changes (1 bit): If set, a bit that specifies that the knowledge specified in the request MUST be modified, prior to change enumeration, such that any changes under a cell node, as implied by the knowledge, cause the knowledge to be modified such that all changes in that cell are returned.

G – Reserved1 (2 bits): A 4-bit reserved field that MUST be set to zero and MUST be ignored.

Query Changes Request Arguments (4 bytes): An optional 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<9>. This field is only sent if **Query Changes Request Arguments** is specified.

G – Include Cell Changes (1 bit): If set, a bit that specifies to include cell changes; otherwise, cell changes are not included<10>. This field is only sent if **Query Changes Request Arguments** is specified.

H - Reserved2 (6 bits): A 6-bit reserved field that MUST be set to zero and MUST be ignored. This field is only sent if **Query Changes Request Arguments** is specified.

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 **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, in bytes, the limit of data elements 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.

Major Version Number (variable): An optional **compact unsigned 32-bit integer** (section 2.2.1.1) that specifies the major version number of the version of the file to query. A **Major Version Number** of 0 and a **Minor Version Number** of 0 is always the most recent version of the document.

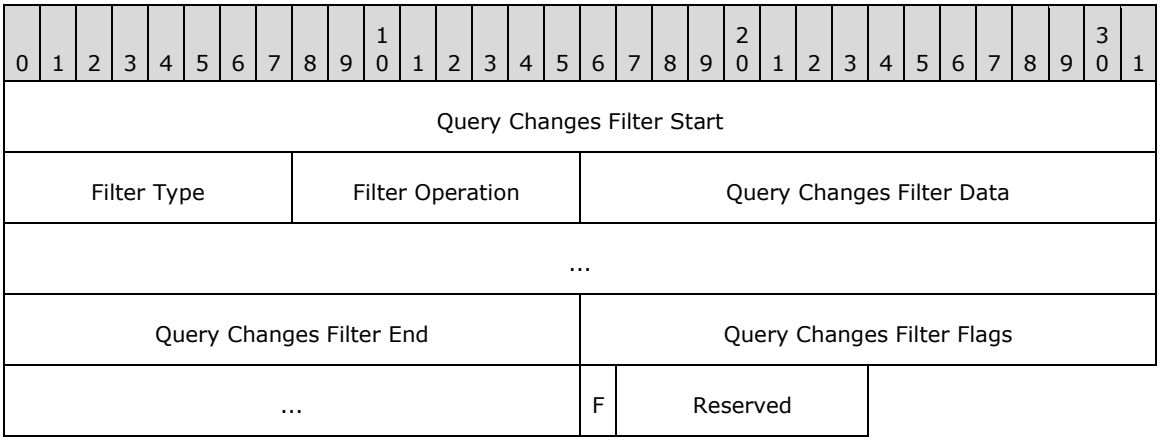
Minor Version Number (variable): An optional **compact unsigned 32-bit integer** (section 2.2.1.1) that specifies the minor version number of the version of the file to query.

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).

Value	Type
6	Data element IDs filter (section 2.2.2.1.3.1.6).
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.

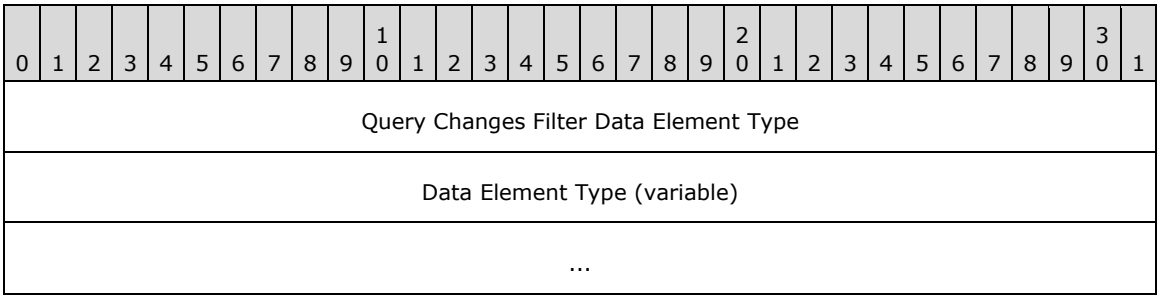
Reserved (7 bits): A 7-bit reserved field that MUST be set to zero, and MUST be ignored if **Query Changes** filter flags are specified.

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, according to the following table.

Value	Data element type
0	None
1	Storage Index
2	Storage Manifest
3	Cell Manifest

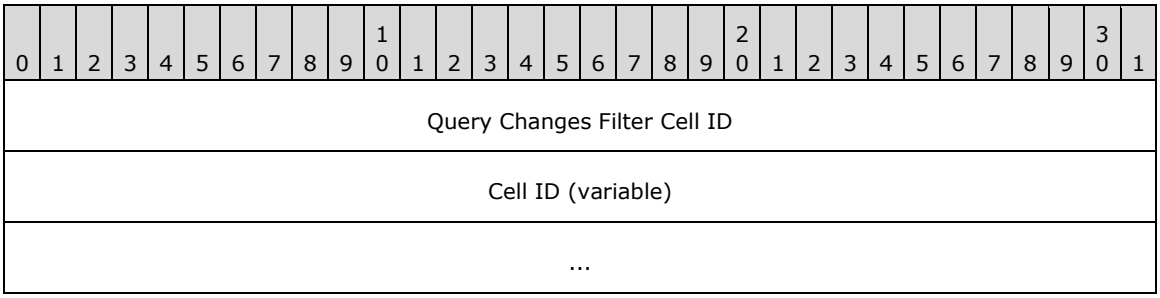
Value	Data element type
4	Revision Manifest
5	Object Group
6	Data Element Fragment
10	Object Data BLOB

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.

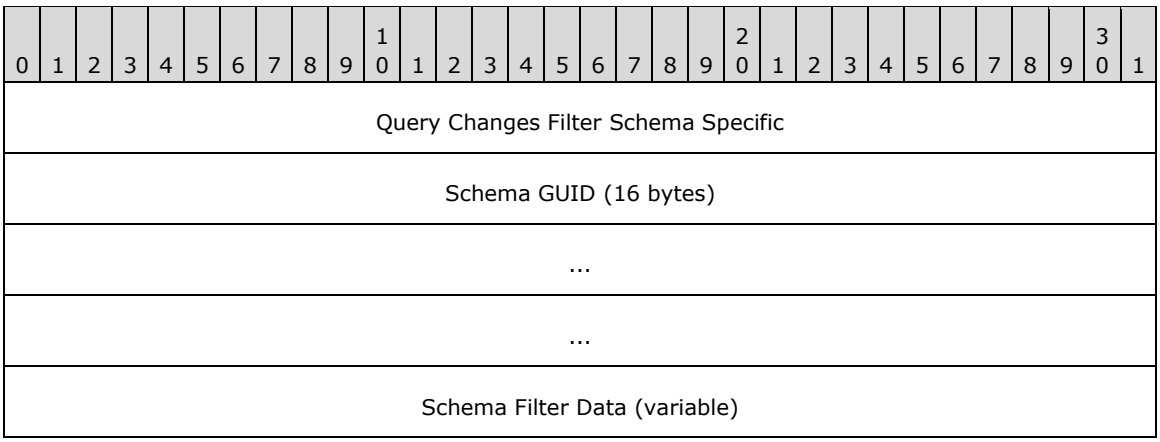


Query Changes Filter Cell ID (4 bytes): A 32-bit **Stream Object Header** (section 2.2.1.5.2) that specifies a **Query Changes** filter cell identifier.

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.



...

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 whose data element identifiers are contained in a specified list 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
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	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Query Changes Filter Hierarchy																															
Depth								Count (variable)																							
...																															
Root Index Key Byte Array (variable)																															
...																															

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.

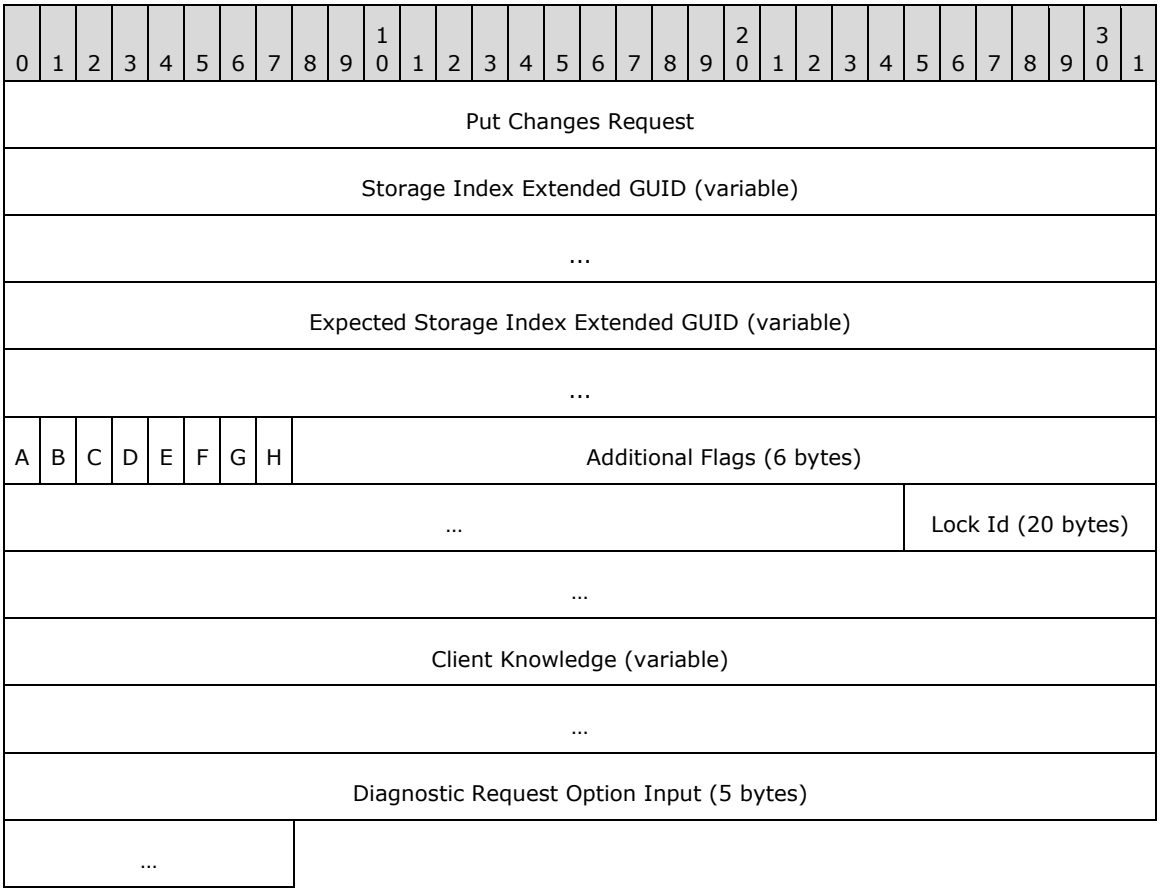
Value	Meaning
0	Index values corresponding to the specified keys only.
1	First data elements referenced by the Storage Index values corresponding to the specified keys only.
2	Single level. All data elements under the sub-graphs rooted by the specified keys stopping at any Storage Index entries.
3	Deep.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

Count (variable): A compact unsigned 64-bit integer (section 2.2.1.1) that specifies the count of bytes in the **Root Index Key Byte Array**.

Root Index Key Byte Array (variable): A byte array that specifies the root index key. The root index key must be a 40-byte array representing two **Extended GUIDs** that specify a Cell Manifest identifier, a 20-byte array representing an **Extended GUID** that specifies a Revision Manifest identifier or an **Extended GUID** that specifies the root Storage Manifest.

2.2.2.1.4 Put Changes

The **Put Changes** sub-request 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.

If the **Extended GUID** does not have the corresponding data element in the **Data Element Package** of the request, the protocol server MUST return a **Cell Error** failure value of 16 indicating the referenced data element not found failure, as specified in section 2.2.3.2.1. 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. When the flag is set to **true**, the **Storage Index Extended GUID (variable)** MUST not be specified.

C - Partial Last (1 bit): A bit that specifies if this is the last **Put Changes** in a partial set of changes. When the flag is set to **true**, the **Storage Index Extended GUID (variable)** MUST be specified.

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.<11>

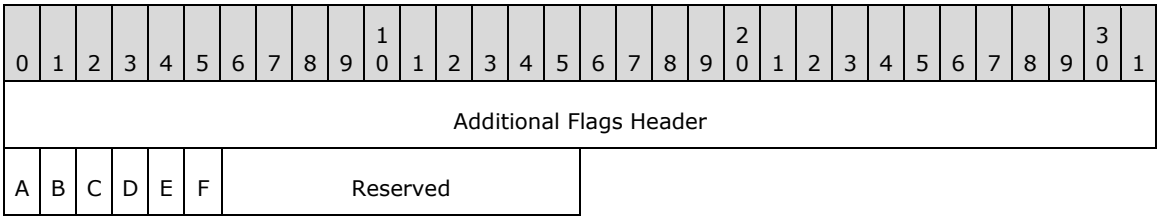
Lock Id (20 bytes): A **Lock Id** structure<12> (section 2.2.2.1.4.2) that specifies the lock ID of the file on the server.

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 the **Resultant Knowledge** (section 2.2.3) in the **Put Changes** response.

Diagnostic Request Option Input (5 bytes): An optional **Diagnostic Request Option Input** structure (section 2.2.2.1.4.3) that specifies desired diagnostic behavior for the **Put Changes** request.<13>

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 returned 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 will 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.

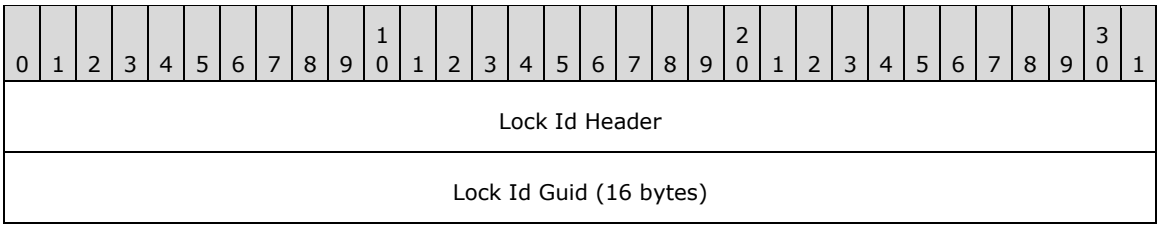
E – Full File Replace Put (1 bit): A bit that specifies that the **Put Changes** request will be treated as a full file save with no dependencies on any pre-existing state. This flag is not required for a full file save, but if specified, the server MAY bypass checks that would otherwise be unnecessary.

F – Require Storage Mappings Rooted (1 bit): A bit that specifies that the **Put Changes** request should fail coherency if any of the supplied Storage Indexes are unrooted.

Reserved (10 bits): A 10-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.

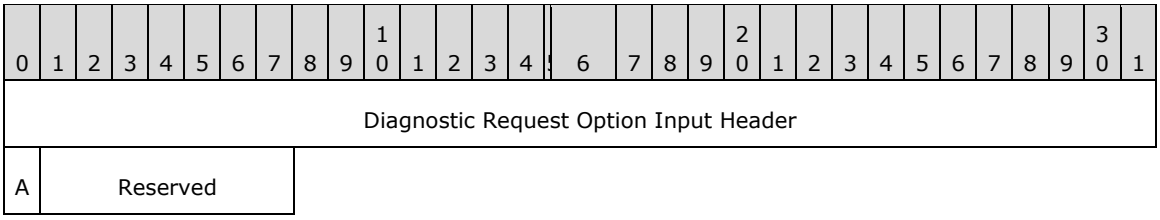


Lock Id Header (4 bytes): A 32-bit **Stream Object Header** (section 2.2.1.5.2) that specifies a **Lock Id** start.

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.

2.2.2.1.4.3 Diagnostic Request Option Input

Diagnostic Request Option Input has the following format.



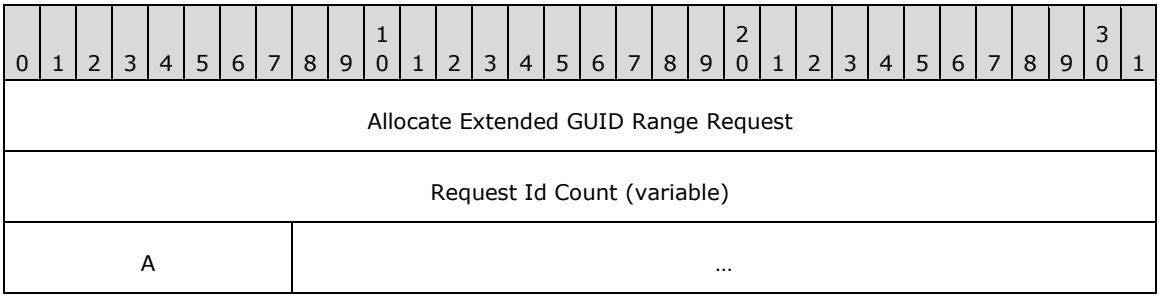
Diagnostic Request Option Input Header (4 bytes): A 32-bit **Stream Object Header** (section 2.2.1.5.2) that specifies a **Diagnostic Request Option Input**.

A – Force Revision Chain Optimization (1 bit): A bit that, if set, specifies that the server should optimize the chain of revisions by refactoring them as part of the **Put Changes** request.

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

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 Extended GUID 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. The valid values for this field are 12, 13 and 14.<14>

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.

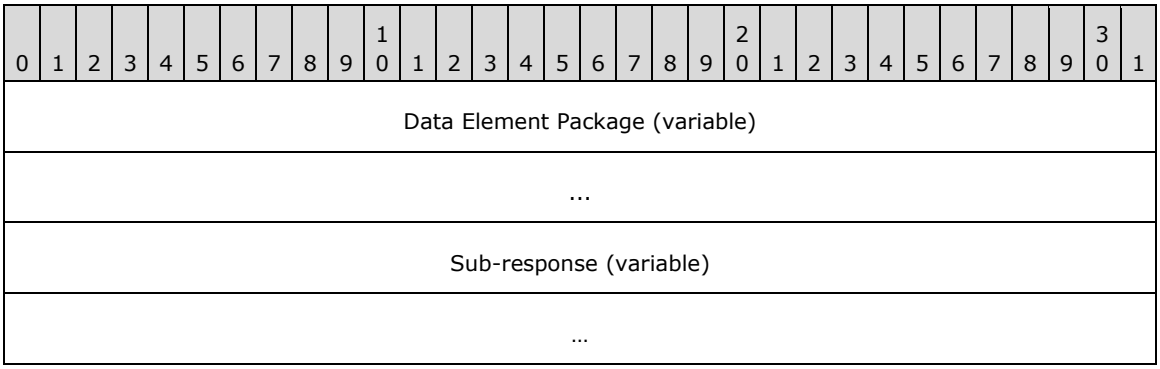
Signature (8 bytes): An unsigned integer that specifies a constant signature, to identify this as a response. This MUST be set to 0x9B069439F329CF9D.

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.



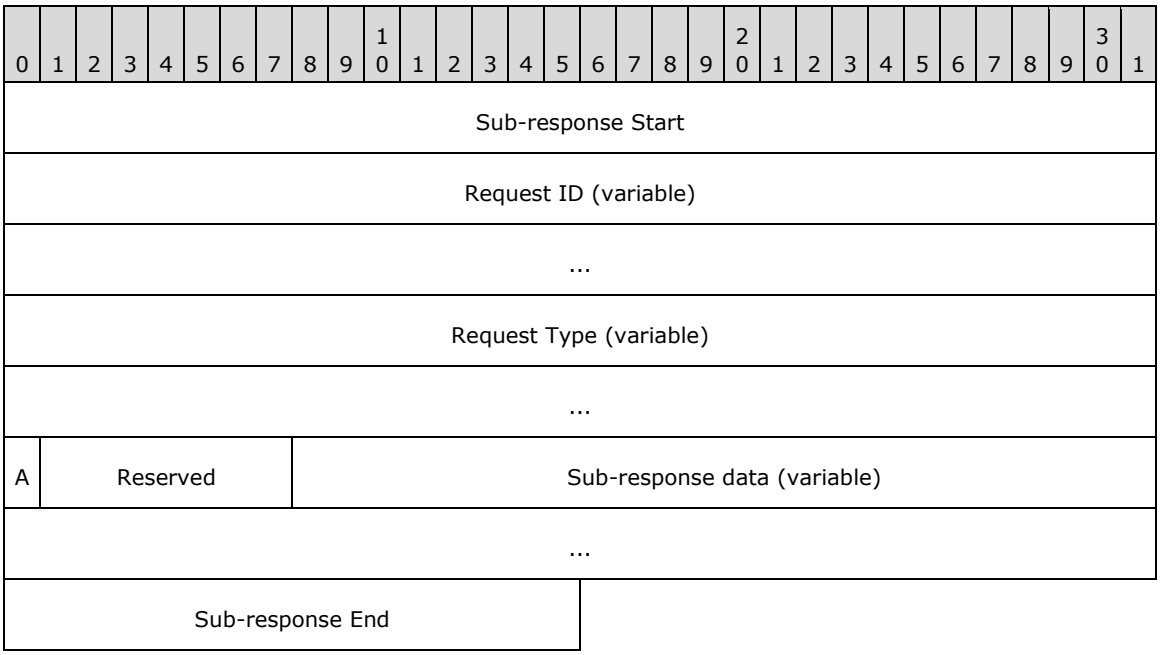
Data Element Package (variable): An optional **Data Element Package** structure (section 2.2.1.12) that specifies data elements corresponding to the sub-responses (section 2.2.3.1). If the sub-responses do not reference data elements or no data elements are available for the sub-response, this structure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

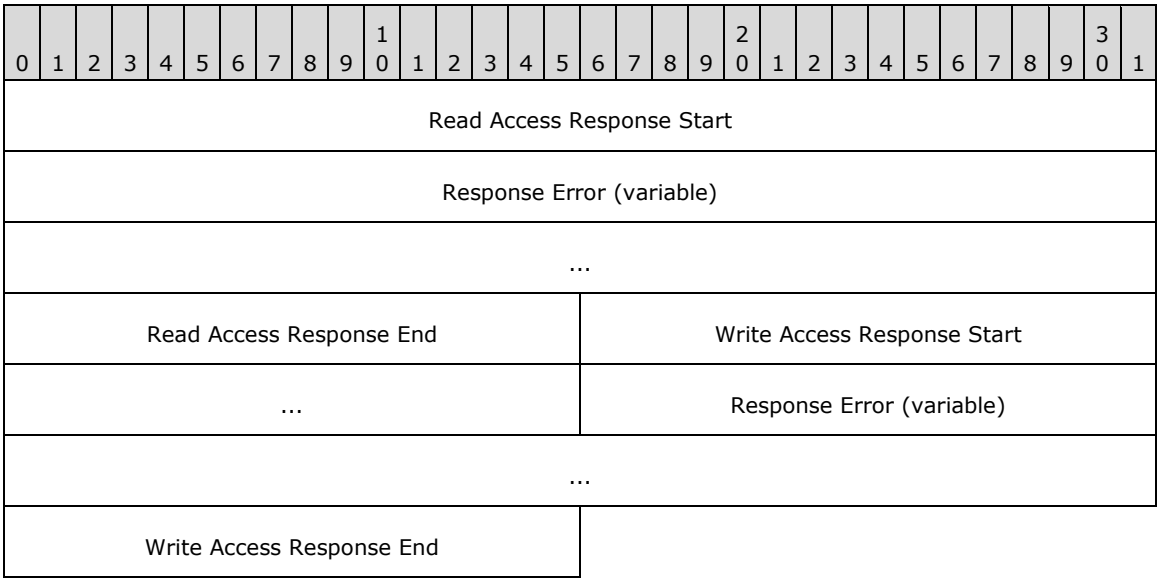
Sub-response data (variable): A **Response Error** that specifies the error information about failure of the sub-request, or depending on the **Request Type** (section 2.2.1.6), specifies additional data. See the following table for details:

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** sub-response 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.

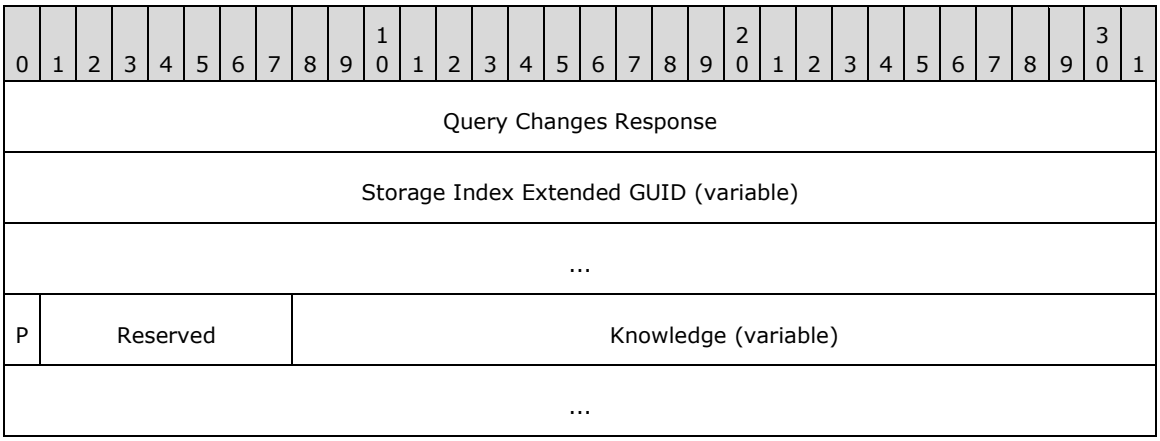
Write Access Response Start (4 bytes): A 32-bit **Stream Object Header** that specifies a write access response start.

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** sub-response returns the set of changes the server has for the data elements in the following format.



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.

2.2.3.1.2.1 Query Changes to Editors Table Partition

If the **Partition Id GUID** of the **Target Partition Id** of the sub-request (section 2.2.2.1.1) of the associated **QueryChangesRequest** is the value of the Guid "7808F4DD-2385-49d6-B7CE-37ACA5E43602", the protocol server will interpret the **QueryChangesRequest** as a download of the editors table specified in [MS-FSSHTTP] section 3.1.4.8.

The editors table is an XML fragment of type **EditorsTable** element (section 2.2.3.1.2.1.2). The protocol server MUST return the editors table in the associated data element collection using the following process:

1. Construct a byte array representation of the editors table using **UTF-8** encoding.
2. Compress the byte array using the **DEFLATE** compression algorithm.
3. Prepend the compressed byte array with the **Editors Table Zip Stream Header** (section 2.2.3.1.2.1.1).
4. Treat the byte array as a file stream and chunk it using the schema described in [MS-FSSHTTPD].
5. Sync the resulting cell using this protocol as you would any other cell.

2.2.3.1.2.1.1 Editors Table Zip Stream Header

A header that is prepended to the compressed **EditorsTable** byte array. This header is an array of eight bytes. The value and order of these bytes MUST be as follows: 0x1A, 0x5A, 0x3A, 0x30, 0x00, 0x00, 0x00, 0x00.

2.2.3.1.2.1.2 EditorsTable

The **EditorsTable** element describes data about active editors in a file.

```
<xs:element name="EditorsTable">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="tns:EditorElement" name="Editor" minOccurs="0" maxOccurs="unbounded"
/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

2.2.3.1.2.1.3 EditorElement

The **EditorElement** describes data about an editor in a file.

```
<xs:element name="EditorElement">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="CacheID" minOccurs="1" maxOccurs="1" type="xs:string"/>
      <xs:element name="FriendlyName" minOccurs="0" maxOccurs="1" type="
type="tns:UserNameType" />
      <xs:element name="LoginName" minOccurs="0" maxOccurs="1" type="tns:UserLoginType" />
      <xs:element name="SIPAddress" minOccurs="0" maxOccurs="1" type="xs:string" />
      <xs:element name="EmailAddress" minOccurs="0" maxOccurs="1" type="xs:string" />
      <xs:element name="HasEditorPermission" minOccurs="0" maxOccurs="1" type="xs:boolean" />
      <xs:element name="Timeout" minOccurs="1" maxOccurs="1" type="xs:positiveInteger" />
      <xs:element name="Metadata" minOccurs="0" maxOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:any minOccurs="0" maxOccurs="unbounded" type="xs:binary" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

CacheID: A string that serves to uniquely identify each client that has access to an editors table on a coauthorable file. This MUST be the same as the **ClientID** attribute of the **EditorsTableSubRequestDataType** ([MS-FSSHTTP] section 2.3.1.23), the **CoauthSubRequestDataType** [MS-FSSHTTP] section 2.3.1.5), the **ExclusiveLockSubRequestDataType** ([MS-FSSHTTP] section 2.3.1.9) or the **SchemaLockSubRequestDataType** ([MS-FSSHTTP] section 2.3.1.13).

FriendlyName: A **UserNameType** that specifies the user name for the client. **UserNameType** is defined in [MS-FSSHTTP] section 2.3.2.7.

LoginName: A **UserLoginType** that specifies the user login alias of the client. **UserLoginType** is defined in [MS-FSSHTTP] section 2.3.2.6.

SIPAddress: A string that specifies the **Session Initiation Protocol (SIP) address** associated with the client.

EmailAddress: A string that specifies the email address associated with the client. The format of the email address MUST be as specified in [RFC2822] section 3.4.1.

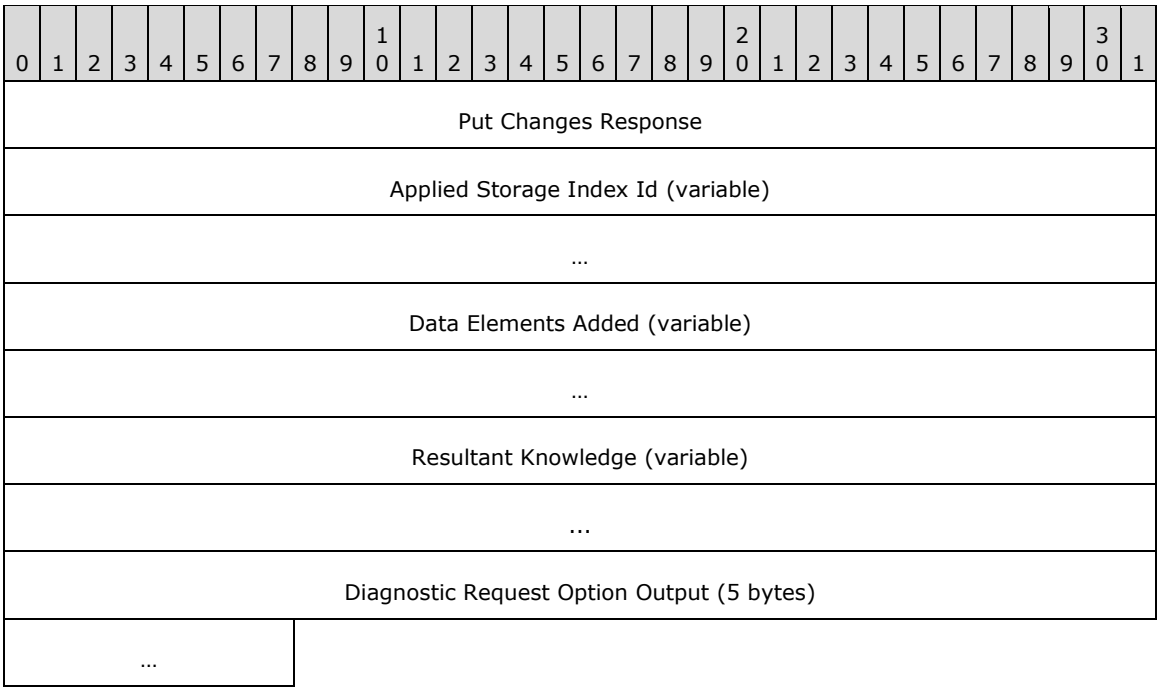
HasEditorPermission: A string that specifies if the editor has permission to make edits to the file.

Timeout: A positive integer that specifies the time when the editor's entry will expire and the editor will no longer be considered an active editor, which is expressed as a tick count. A single tick represents 100 nanoseconds, or one ten-millionth of a second. **ServerTime** ([MS-FSSHTTP] section 3.1.4.7) specifies the number of 100-nanosecond intervals that have elapsed since 00:00:00 on January 1, 1601, which MUST be **Coordinated Universal Time (UTC)**.

Metadata: An element that specifies any arbitrary key-value pairs that the protocol client has provided. Each contained element represents one such key-value pair. The name of the element is the key. The binary content is the value.

2.2.3.1.3 Put Changes

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



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 identifier.<15>

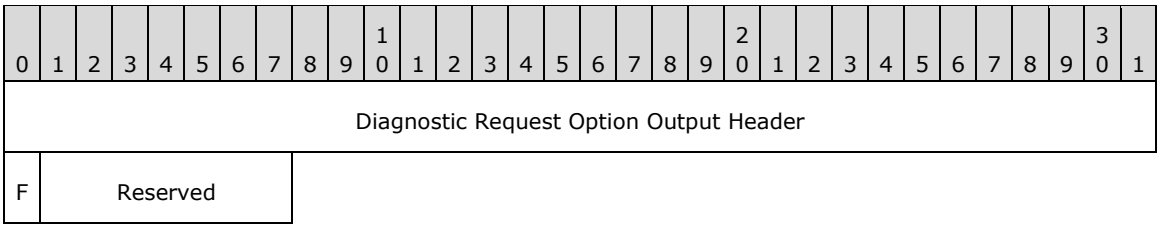
Data Elements Added (variable): An **Extended GUID Array** (section 2.2.1.8) that specifies the data element identifiers of the added data elements.<16>

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.

Diagnostic Request Option Output (5 bytes): An optional **Diagnostic Request Option Output** structure (section 2.2.3.1.3.1) that specifies the result of any diagnostic options performed for a request.<17>

2.2.3.1.3.1 Diagnostic Request Option Output

The **Diagnostic Request Option Output** has the following format.



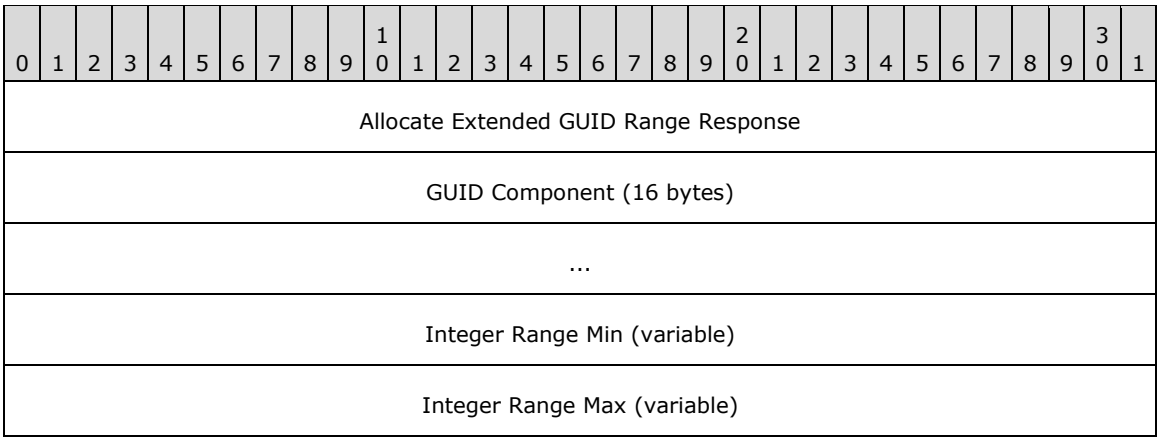
Diagnostic Request Option Output Header (4 bytes): A 32-bit **Stream Object Header** (section 2.2.1.5.2) that specifies a **Diagnostic Request Option Output**.

F – Forced (1 bit): A bit that specifies whether a forced Revision Chain optimization occurred.

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

2.2.3.1.4 Allocate Extended GUID Range

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



Allocate Extended GUID Range Response (4 bytes): A **Stream Object Header** (section 2.2.1.5) that specifies an **Allocate Extended GUID Range** response.

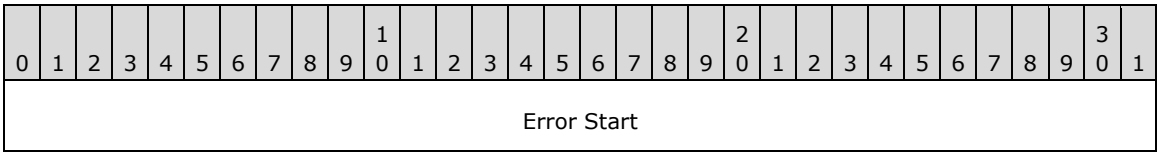
GUID Component (16 bytes): A GUID that specifies the GUID portion of the reserved **Extended GUIDs** (section 2.2.1.7).

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 1,000 and maximum allowed value of 100,000 that 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 Type GUID (16 bytes)
...
...
Error Data (variable)
...
Error String Supplemental Info Start (4 bytes, optional)
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	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
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.

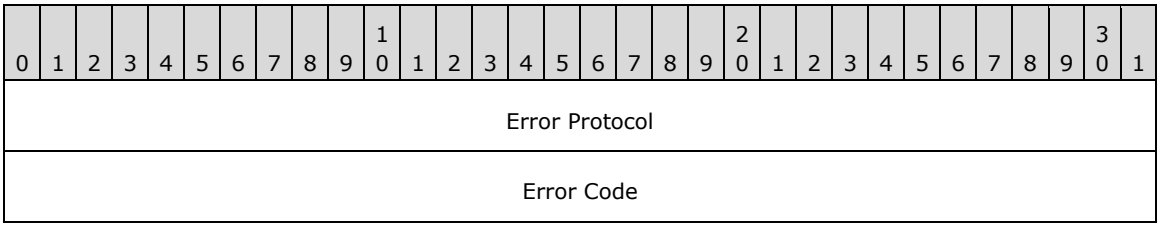
Error Code (4 bytes): An unsigned integer that specifies the error code. The following table contains the possible error codes.

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.
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.

Error code	Error
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 GUID**s.
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.



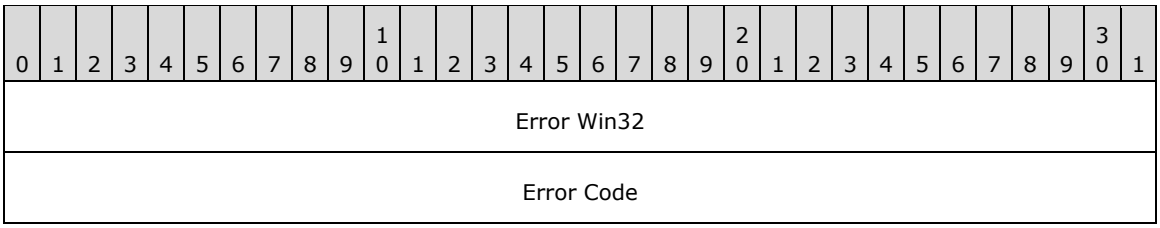
Error Protocol (4 bytes): A 32-bit **Stream Object Header** (section 2.2.1.5.2) that specifies an error protocol.

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.

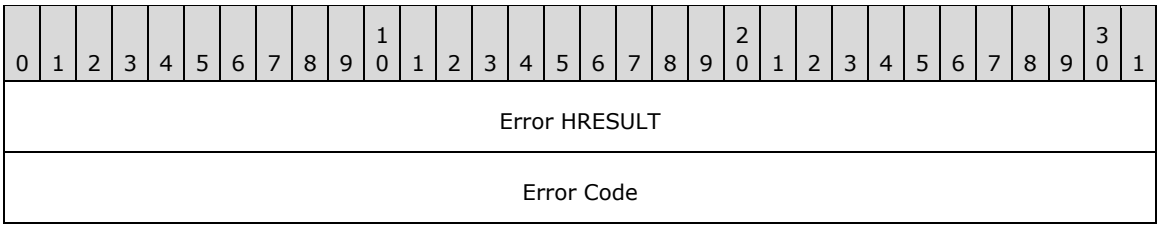


Error Win32 (4 bytes): A 32-bit **Stream Object Header** (section 2.2.1.5.2) that specifies an error win32.

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.



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.

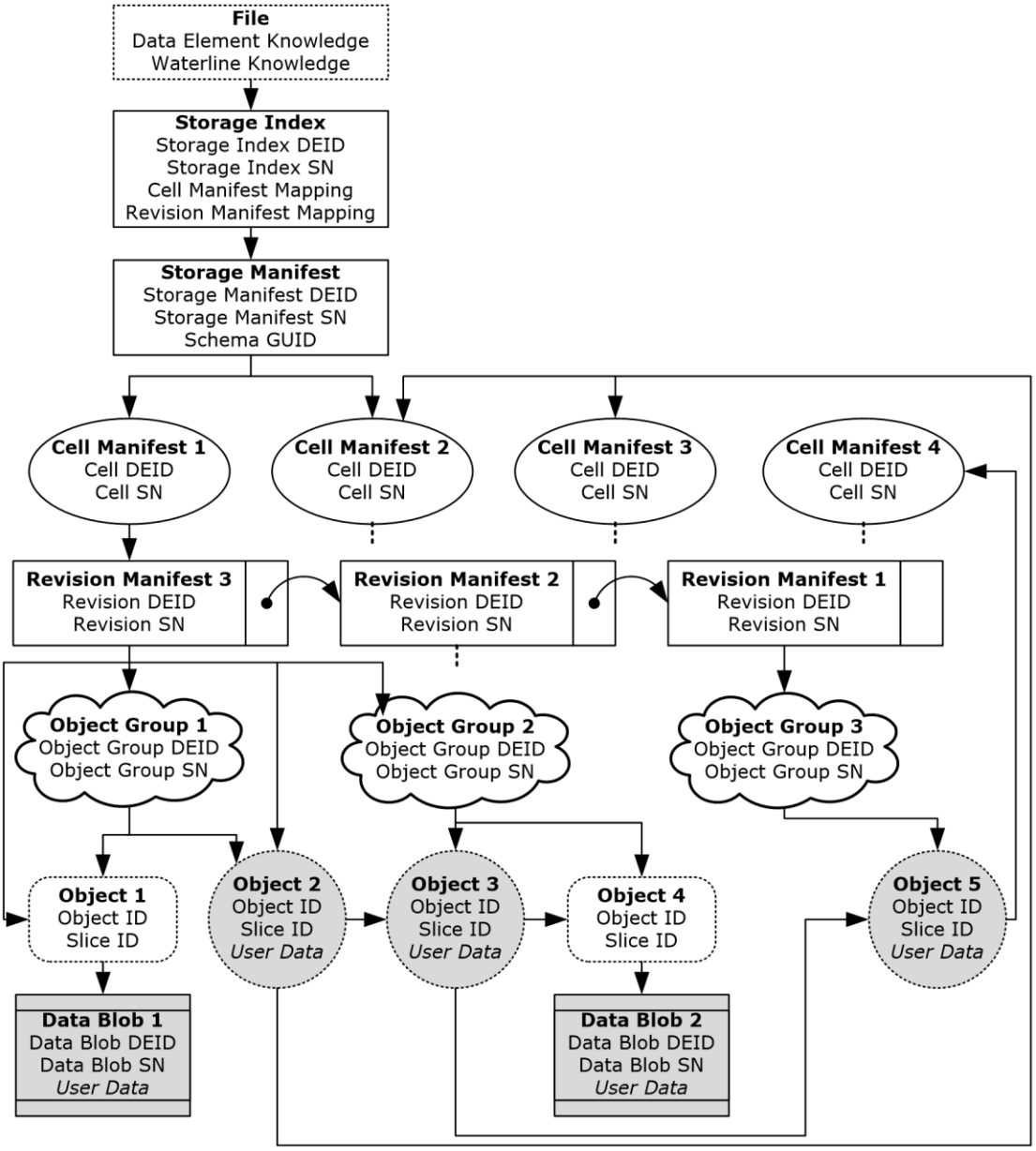


Figure 1: Abstract Data Model

The figure shows components as follows:

- Similar shapes represent similar objects.
- Shaded elements are leaf nodes.
- Shaded boxes are objects that store file data.
- Shaded circles are objects that contain pointers to other objects.

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. All data elements are immutable.

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 (section 2.2.1.12.2).

- **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. There is only one Storage Index data element for a file.

- **Storage Index DEID:** The **Extended GUID** (section 2.2.1.7) of the Storage Index data element.
- **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. There is only one Storage Manifest data element for a file.

- **Storage Manifest DEID:** The **Extended GUID** of the Storage Manifest data element.
- **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 Manifest DEID:** The **Extended GUID** of the Cell Manifest data element for this cell.
- **Cell Manifest 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 Manifest DEID:** The **Extended GUID** of the Revision Manifest data element for this revision.
- **Revision Manifest 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 (section 2.2.1.12.6) 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 **Extended GUID** of this Object Group data element.
- **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 the 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 the 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 (section 2.2.1.12.8) node that contains the user data for the 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 **Extended GUID** of this Data Blob data element.
- **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.
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.
Allocate Extended GUID Range	Client requests a unique range of **Extended GUID** (section 2.2.1.7) values.

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.<18>

If **Request Data Element Hashes** (section 2.2.2) is 1 or **Request Data Element Hashes Instead of Data** (section 2.2.2) is 1, the server MUST return **Data Element Hashes** (section 2.2.2) in the schema specified by **Request Hashing Schema** (section 2.2.2) for Object Group data elements (section 2.2.1.12.6) that 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 two objects 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.

<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="32">Data 1</td></tr>
<tr><td colspan="16">Data 2</td><td colspan="16">Data 3</td></tr>
<tr><td colspan="8">Data 4</td><td colspan="8">Data 5</td><td colspan="8">Data 6</td><td colspan="8">Data 7</td></tr>
<tr><td colspan="8">Data 8</td><td colspan="8">Data 9</td><td colspan="8">Data 10</td><td colspan="8">Data 11</td></tr>
</table>

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 following table specifies the compare method for two objects, Object A and Object B.

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. Stop.	Continue to step 9.	A is greater than 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 Extended GUID 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 **Extended GUID** values (section 2.2.1.7).<19> The server MUST reply back to the client with an **Allocate Extended GUID 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.<20>

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** specified in 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 are able to 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 Extended GUID 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 **Extended GUIDs** in subsequent **Put Changes** sub-requests (section 2.2.2.1.4), a server compliant with this specification can store data more efficiently than one operating with client-allocated **Extended GUIDs**.

3.2.5 Timer Events

None.

3.2.6 Other Local Events

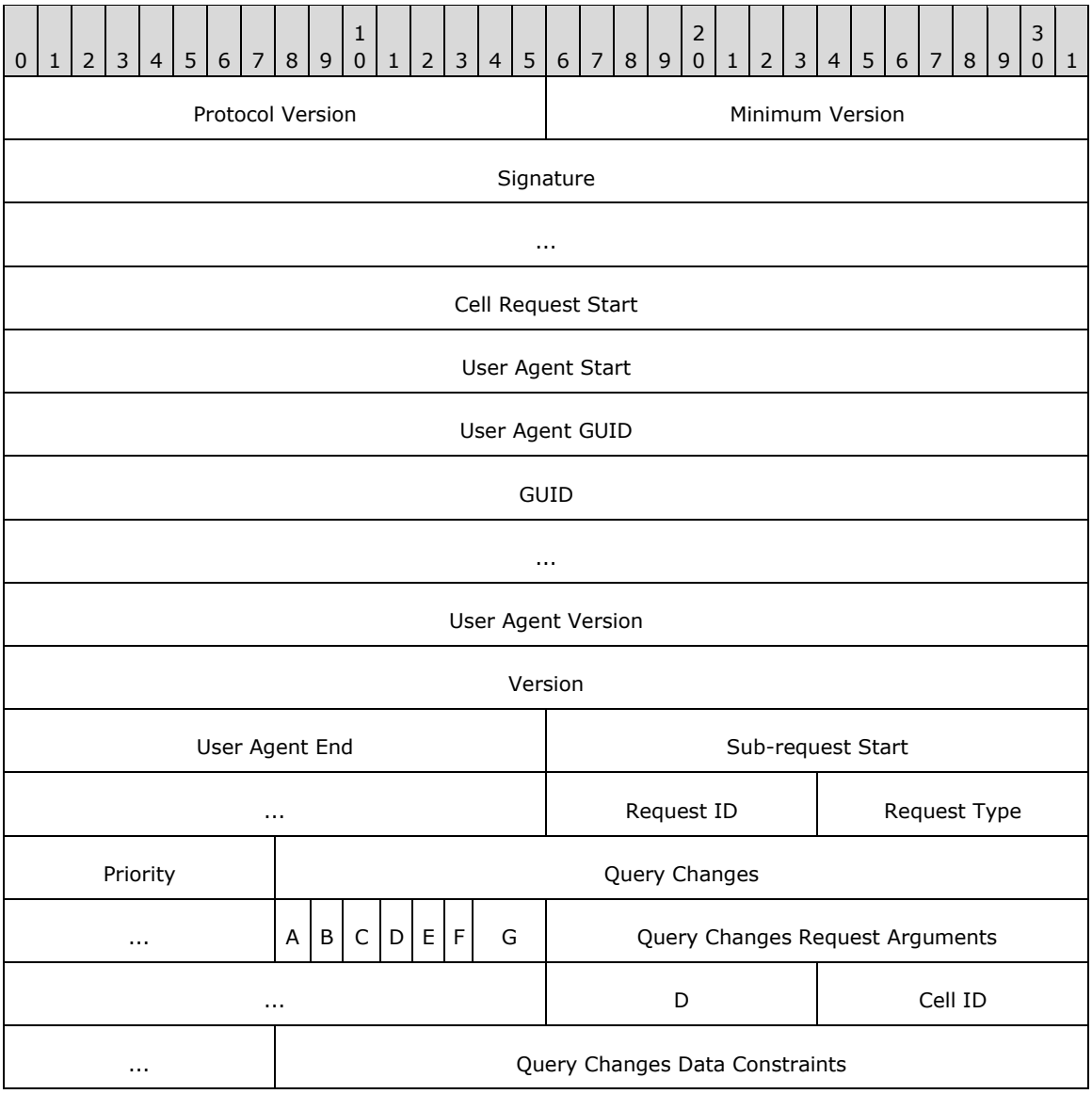
None.

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
```



...	Maximum Data Elements	
...	Knowledge Start	Knowledge End
Sub-request End	Data Element Package Start	
Reserved	J	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 a **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 - Exclude Object Data (1 bit): zero, ignored by the server.

C - Exclude Object Data/Include Filtered Out Data Elements In Knowledge/ Reserved1 (6 bits): Zero specifies that the **serial numbers** of filtered out data elements are not included in the response **knowledge**.

Query Changes Request Arguments (4 bytes): zero specifies that fragments are not allowed.

F - Round Knowledge to Whole Cell Changes (1 bit): zero specifies that knowledge is not rounded to whole cell changes.

G - Reserved (2 bits): Zero specifies reserved bits.

H - 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.

D - Include Storage Manifest/Include Cell Changes/ Reserved2 (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 a 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.

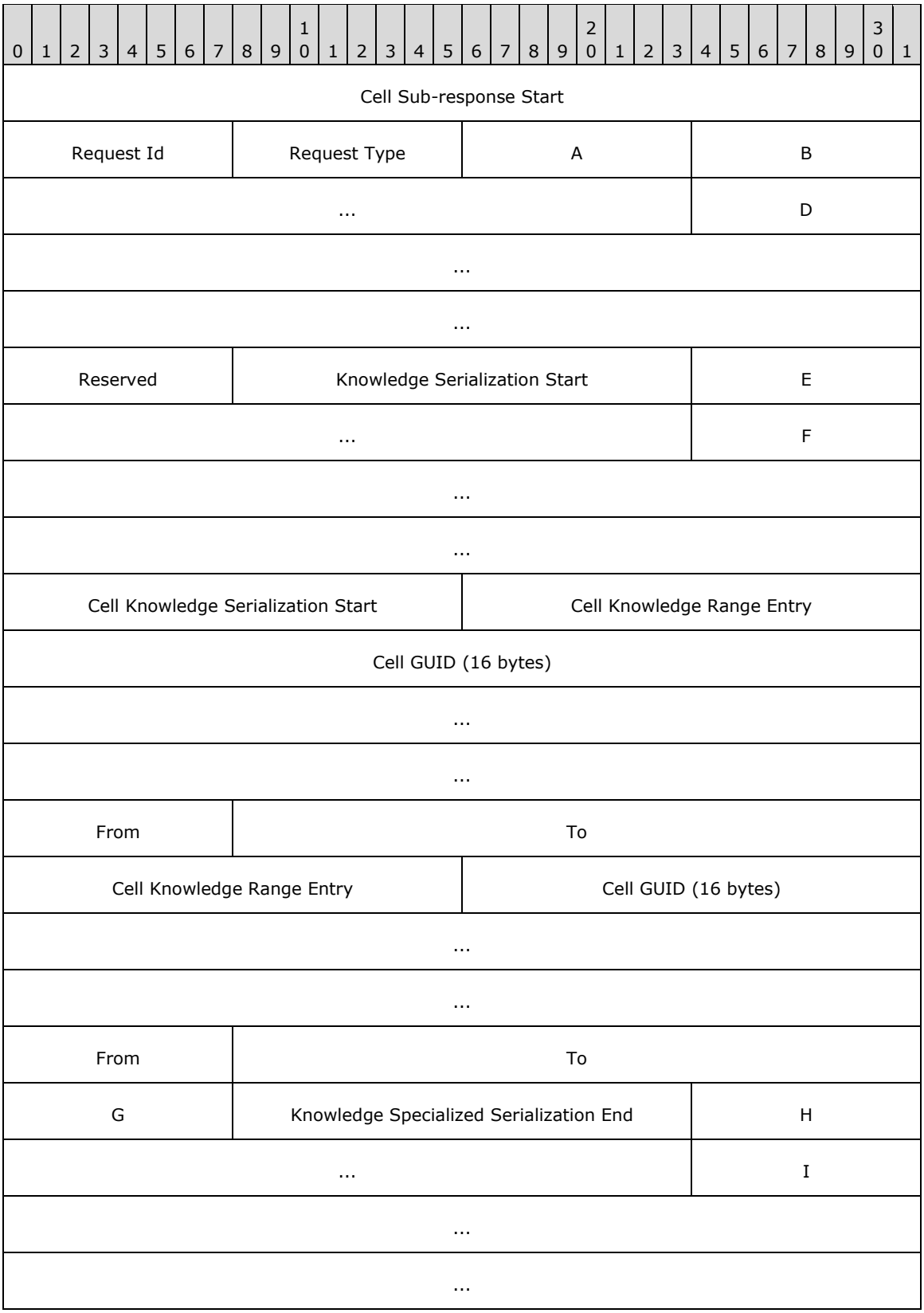
J - Data Element Package End (1 byte): 0x55 specifies the 8-bit **Stream Object Header** for the **Data Element Package** end. This stream object was started in the **Request Header** (section 4.3.1).

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**.

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
```



Reserved	Waterline Knowledge Start	J
...	Cell Storage Extended GUID (16 bytes)	
...		
...		
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.

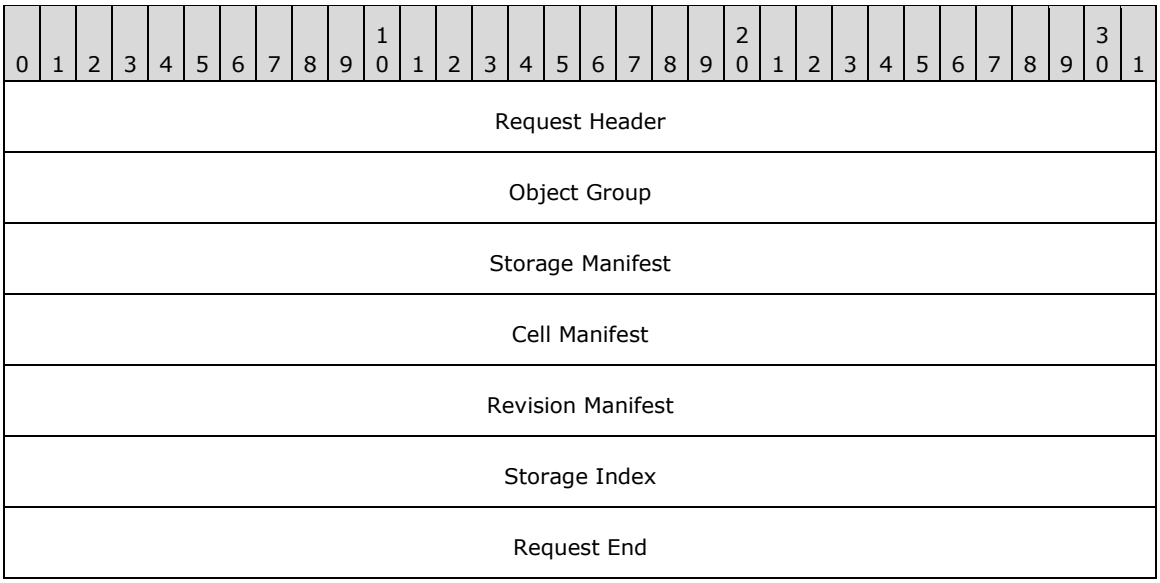
Cell Sub response End (2 bytes): 0x0701 specifies a 16-bit **Stream Object Header** (section 2.2.1.5.4) for cell sub-response end. Decoded, this has a value of 0x041.

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).

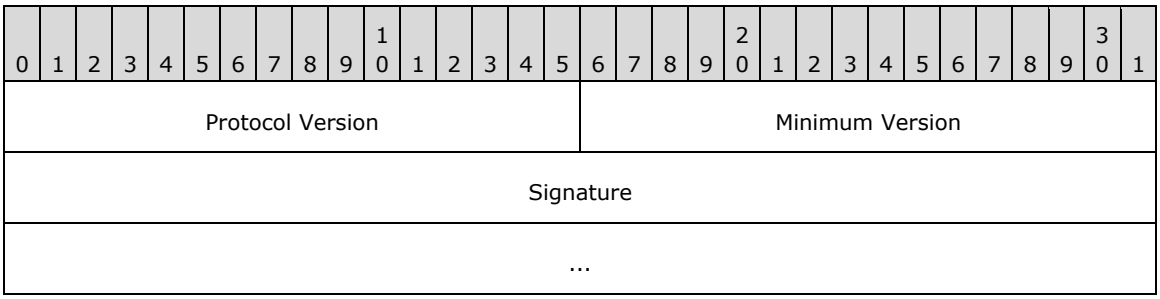
Storage Index (4 bytes): Contents of **Storage Index** (section 4.3.6).

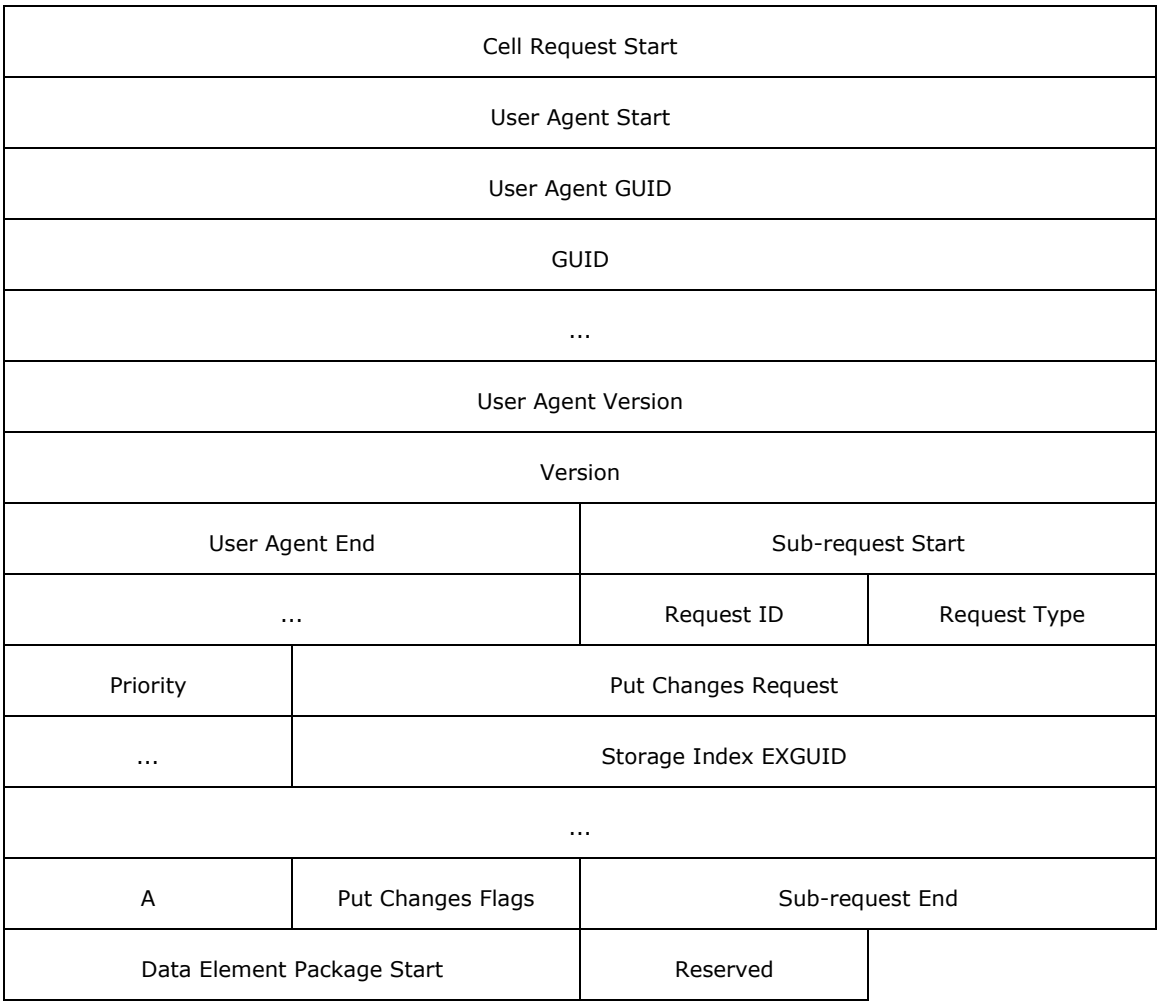
Request End (4 bytes): Contents of **Request End** (section 4.3.7).

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.

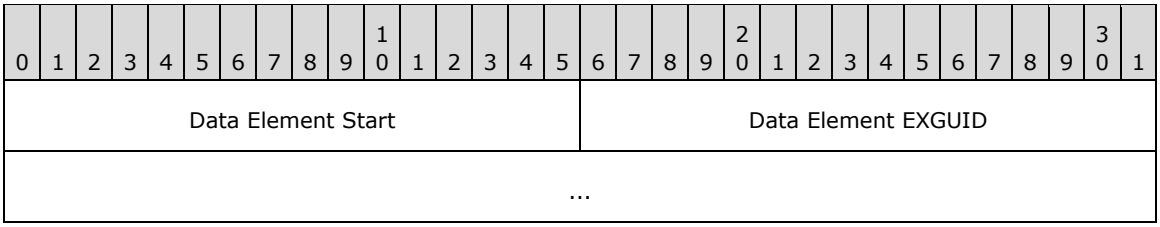
Data Element Package Start (2 bytes): 0x02AC specifies the 16-bit **Stream Object Header** for a **Data Element Package** (section 2.2.1.12) start. Decoded, this has a type of 0x15, length 1, and is compound.

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
```



SN			
Data Element Type	Object Group Declarations Start		Object Declaration
...	Object EXGUID		
...			
Object Partition ID	Object Data Size	A	Cell References Count
B	C	Reserved	D
...			E
...			
Object Group Data End	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: {"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.

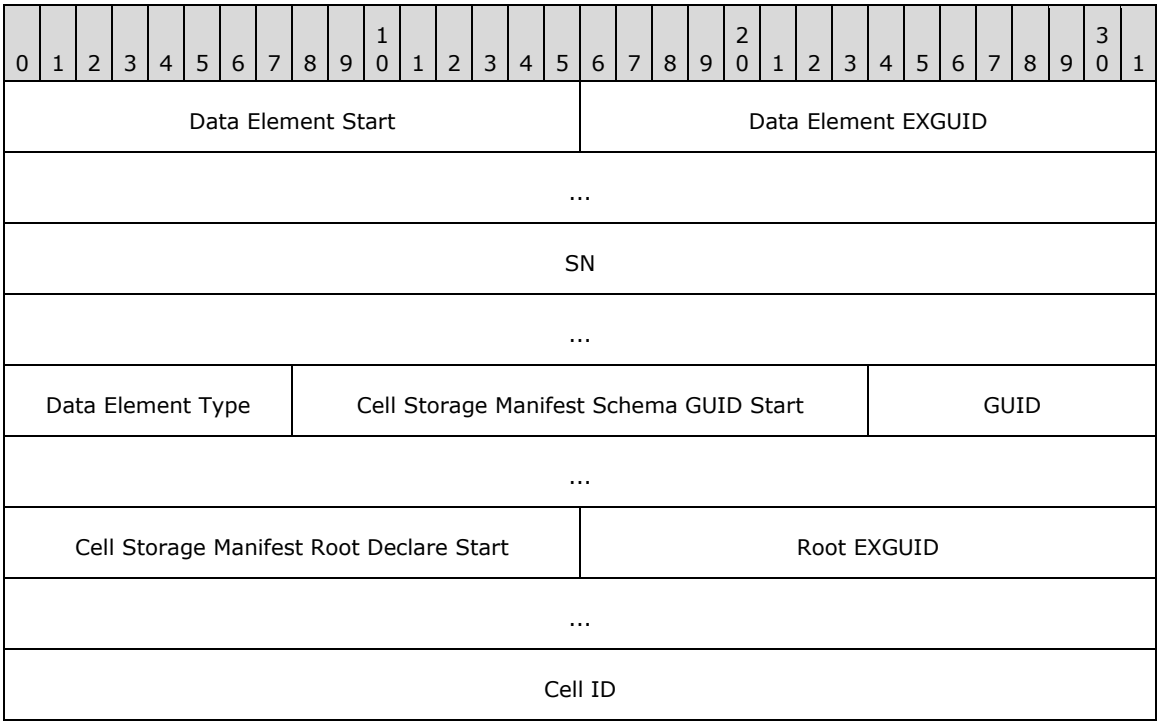
Object Group Data End (1 byte): 0x79 specifies the 8-bit **Stream Object Header** for Object Group data end.

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 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: {"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
```

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
Data Element Start																Data Element EXGUID															

...		
SN		
...		
Data Element Type	Cell Manifest Current Revision Start	A
...		
Data Element End		

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.

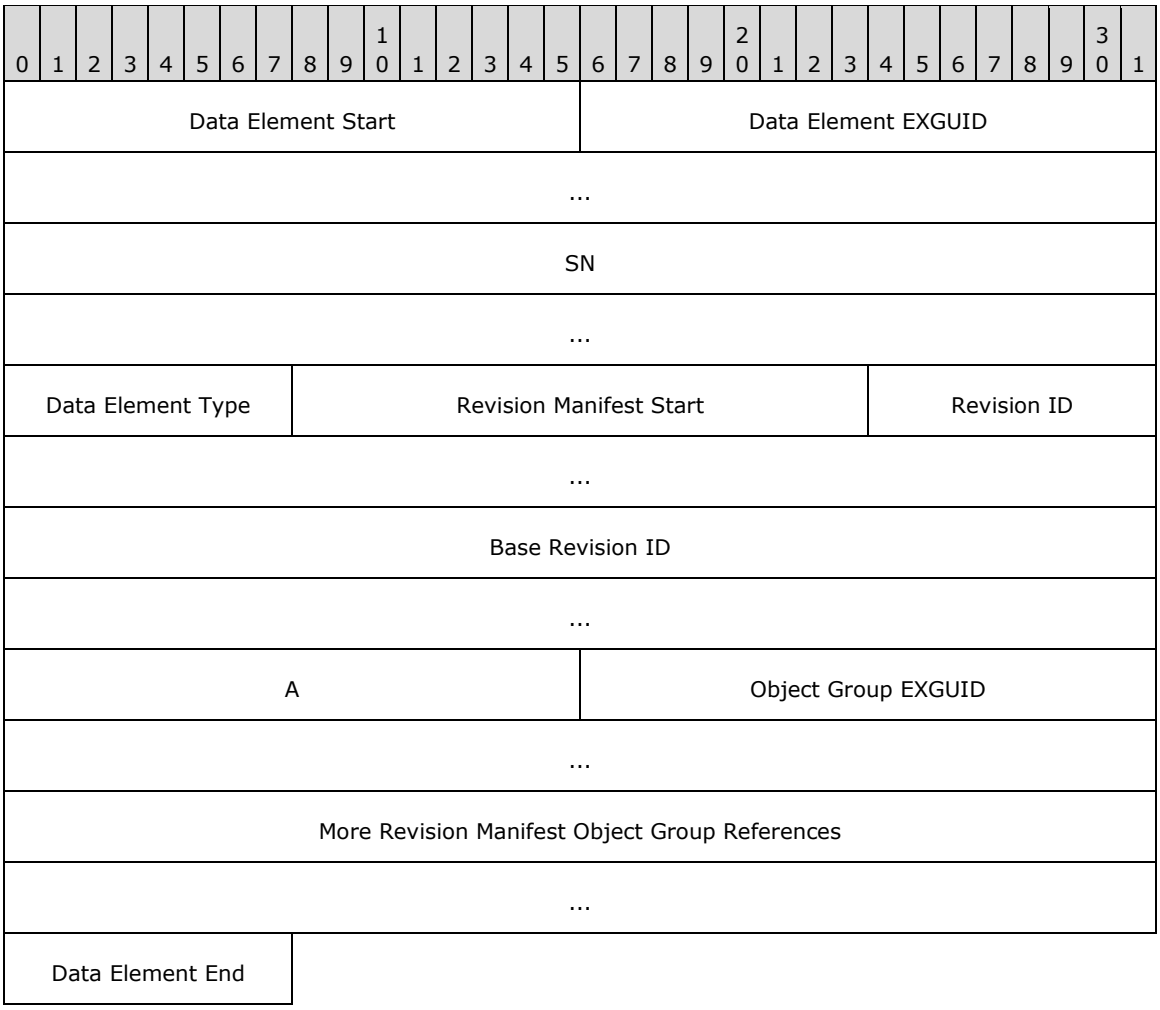
A - Cell Manifest Current Revision EXGUID: {"7128FE2A-DCBE-4301-BD84-716C456C808A"} 0x01 specifies a string representation of the **Current Revision Extended GUID** (section 2.2.1.7) decoded from 0C 3A FE 28 71 BE DC 01 43 BD 84 71 6C 45 6C 80 8A.

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
```



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.

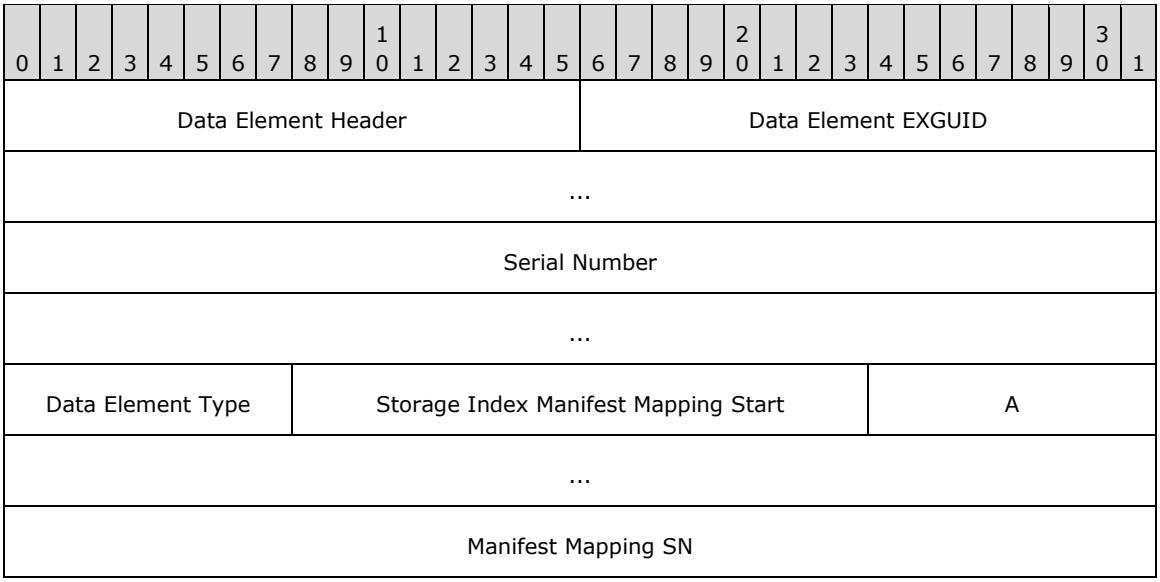
More Revision Manifest Object Group References: The rest of the Revision Manifest Object Group reference elements have been omitted from this example.

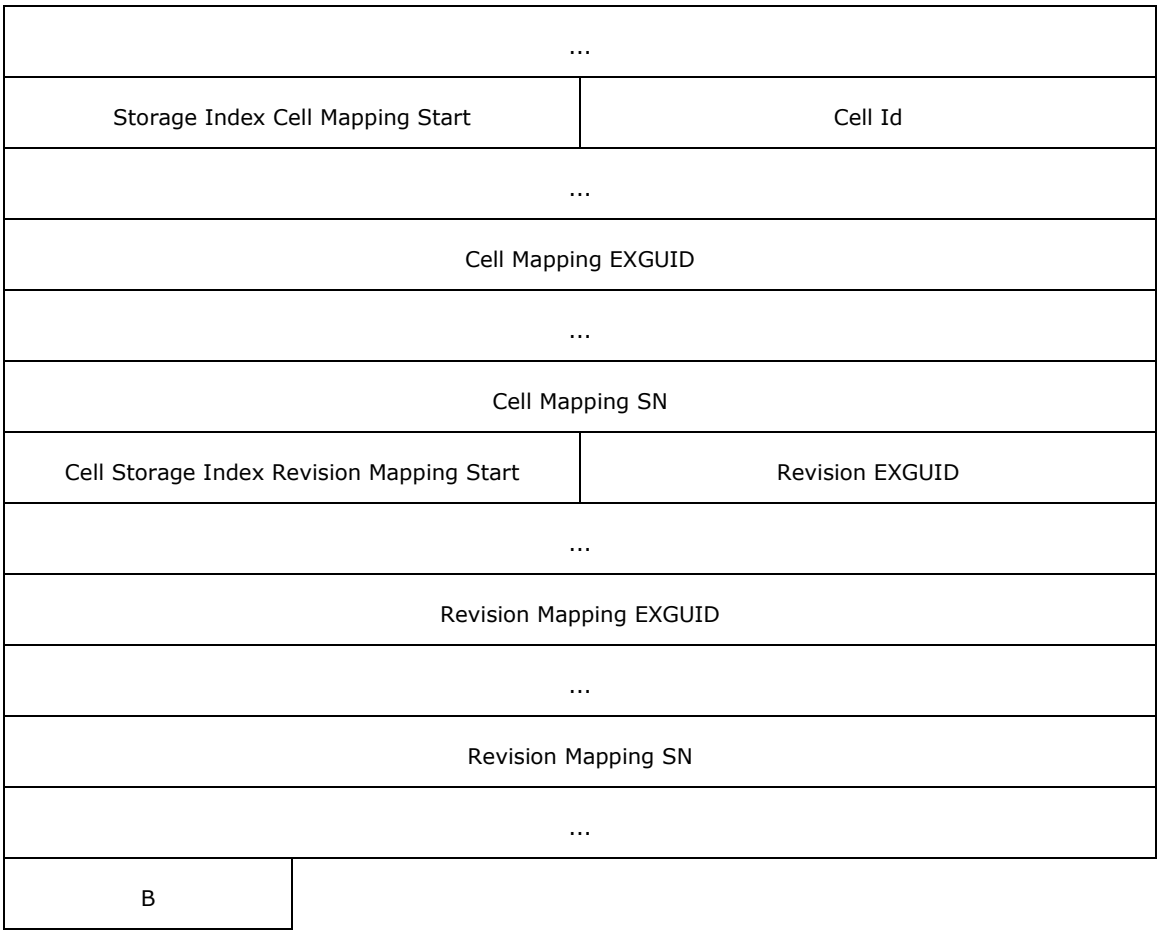
Data Element End (1 byte): 0x05 specifies the 8-bit **Stream Object Header** (section 2.2.1.5.3) for data element end.

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.

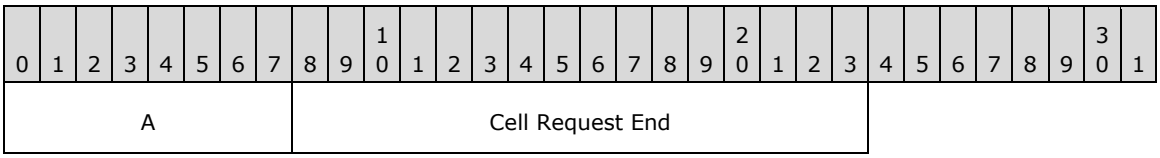
Revision Mapping SN: (0x80 {"8EABCF50-6491-98BF-0670-707E818DC102"} 0x00) specifies the **Revision Mapping Serial Number** decoded from 80 B8 50 CF AB 8E 91 64 BF 98 06 70 7E 81 8D C1 02 3F 00 00 00 00 00 00 00.

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).

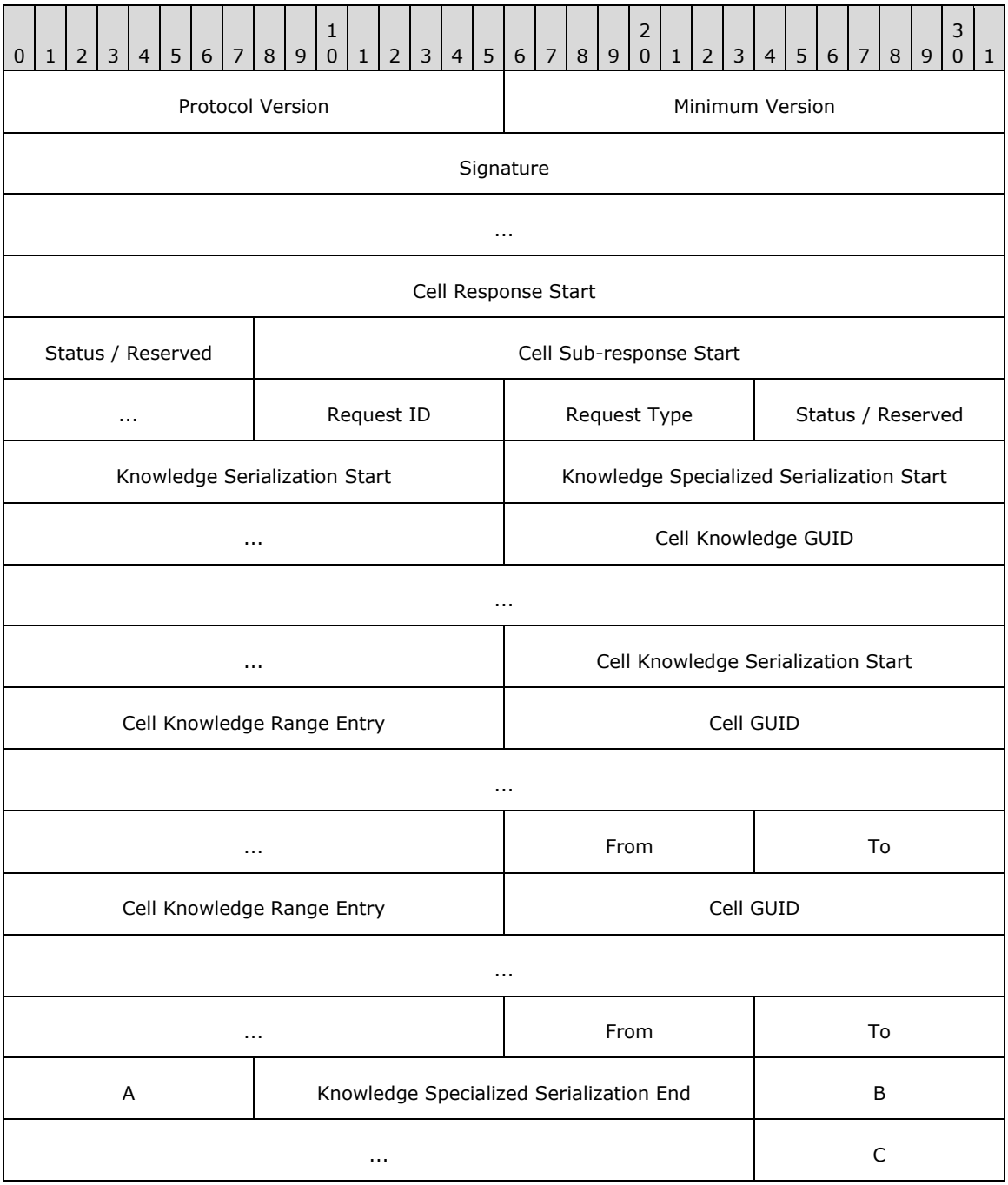
Cell Request End (2 bytes): 0x0103 specifies the 8-bit **Stream Object Header** for cell request end. This stream object was started in the **Request Header** section.

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
```



...	Content Knowledge Tag Entry	D
...		
Clock Data		
...	E	Knowledge Specialized Serialization End
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 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
- Windows 8.1 Update
- Microsoft Office 2016
- Windows 10 operating system
- Microsoft SharePoint Server 2016

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 2.2.1.12.6.1: SharePoint Server 2010 and SharePoint Workspace 2010 might return the wrong value.

<2> Section 2.2.1.12.6.3: SharePoint Server 2010 and SharePoint Workspace 2010 do not support the **Object Metadata Declaration**.

<3> Section 2.2.2: **Data Element Hashing is not supported by** Office 2010 and support is configuration-dependent for other versions.

<4> Section 2.2.2: Office 2013 uses only protocol schema version 13, and Office 2016 uses only protocol schema version 14.

<5> Section 2.2.2: SharePoint Server 2010 and SharePoint Workspace 2010 do not support the **Request Hashing Options Declaration** field.

<6> Section 2.2.2.1: SharePoint Server 2010 and SharePoint Server 2013 execute **Sub-requests** with different or same **Priority** in any order with respect to each other.

<7> Section 2.2.2.1: SharePoint Server 2010 and SharePoint Workspace 2010 do not support the **Target Partition Id** field.

<8> Section 2.2.2.1.3: Microsoft SharePoint Server 2010 and Microsoft SharePoint Workspace 2010 ignore the **D – Include Filtered Out Data Elements In Knowledge** field.

<9> Section 2.2.2.1.3: SharePoint Server 2010 and SharePoint Workspace 2010 ignore the **F – Include Storage Manifest** field.

<10> Section 2.2.2.1.3: SharePoint Server 2010 and SharePoint Workspace 2010 ignore the **G – Include Cell Changes** field.

<11> Section 2.2.2.1.4: SharePoint Server 2010 and SharePoint Workspace 2010 do not support the **Additional Flags** structure

<12> Section 2.2.2.1.4: SharePoint Server 2010 does not support this attribute.

<13> Section 2.2.2.1.4: Microsoft Office 2010 suites does not support the **Diagnostic Request Option Input** field.

<14> Section 2.2.3: Office 2013 uses only protocol schema version 13, Office 2016 uses only protocol schema version 14.

<15> Section 2.2.3.1.3: SharePoint Server 2010 and SharePoint Workspace 2010 do not support the **Applied Storage Index Id** field

<16> Section 2.2.3.1.3: SharePoint Server 2010 and SharePoint Workspace 2010 do not support the **Data Elements Added** field

<17> Section 2.2.3.1.3: Office 2010 does not support the **Diagnostic Request Option Output** field.

<18> Section 3.1.4.2: Data Element Hashing is not supported by Office 2010 and support is configuration-dependent for other versions.

<19> Section 3.1.4.4: SharePoint Server 2010 and SharePoint Workspace 2010 not support this sub-request.

<20> 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 this document since the last release. 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.
- The removal of a document from the documentation set.

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 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 changes were introduced. Minor editorial and formatting changes may have been made, but the technical content of the document is identical to the last released version.

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.
- 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 dochelp@microsoft.com.

Section	Tracking number (if applicable) and description	Major change (Y or N)	Change type
2.2.2 Request Message Syntax	Added Version field in the request message diagram art.	N	Content update.

9 Index

N

O

P

Q

R

S

T

V