[MS-FSSHTTP]:

File Synchronization via SOAP over HTTP Protocol

development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

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	No Change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.06	No Change	No changes to the meaning, language, or formatting of the technical content.
3/18/2011	1.06	No Change	No changes to the meaning, language, or formatting of the technical content.
6/10/2011	1.06	No Change	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	No Change	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	No Change	No changes to the meaning, language, or formatting of the technical content.
10/8/2012	2.2	Minor	Clarified the meaning of the technical content.
2/11/2013	3.0	Major	Significantly changed the technical content.
7/30/2013	3.1	Minor	Clarified the meaning of the technical content.
11/18/2013	3.2	Minor	Clarified the meaning of the technical content.
2/10/2014	3.3	Minor	Clarified the meaning of the technical content.
4/30/2014	3.4	Minor	Clarified the meaning of the technical content.
7/31/2014	3.5	Minor	Clarified the meaning of the technical content.
10/30/2014	3.6	Minor	Clarified the meaning of the technical content.
3/16/2015	4.0	Major	Significantly changed the technical content.

Table of Contents

1 Introduction

The File Synchronization via SOAP over HTTP Protocol enables one or more protocol clients to synchronize changes done on shared files stored on a server.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in [RFC2119]. Sections 1.5 and 1.9 are also normative but do not contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

The following terms are specific to this document:

absolute URL: The full Internet address of a page or other World Wide Web resource. The absolute URL includes a protocol, such as "http," a network location, and an optional path and file name — for example, http://www.treyresearch.net/.

base64 encoding: A binary-to-text encoding scheme whereby an arbitrary sequence of bytes is converted to a sequence of printable ASCII characters, as described in [RFC4648].

claim-based authentication mode: A set of operations that is used to establish trust relationships between claims providers and relying party applications. It involves the exchange of identifying certificates (1) that make it possible for a relying party to trust the content of a claim (2) that is issued by a claims provider.

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).

friendly name: A name for a user or object that can be read and understood easily by a human.

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 specified in [RFC4122] or [C706] must be used for generating the **GUID**. See also universally unique identifier (UUID).

HRESULT: An integer value that indicates the result or status of an operation. A particular HRESULT can have different meanings depending on the protocol using it. See [MS-ERREF] section 2.1 and specific protocol documents for further details.

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.

Information Rights Management (IRM): A technology that provides persistent protection to digital data by using encryption, certificates (1), and authentication (2). Authorized recipients or users acquire a license to gain access to the protected files according to the rights or business rules that are set by the content owner.

request token: A unique identifier that identifies a Request element in a service request.

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].

SOAP fault: A container for error and status information within a SOAP message. See [SOAP1.2-1/2007] section 5.4 for more information.

SOAP Message Transmission Optimization Mechanism (MTOM): A method that is used to optimize the transmission and format of SOAP messages by encoding parts of the message, as described in [SOAP1.2-MTOM].

Uniform Resource Locator (URL): A string of characters in a standardized format that identifies a document or resource on the World Wide Web. The format is as specified in [RFC1738].

Web Distributed Authoring and Versioning Protocol (WebDAV): The Web Distributed Authoring and Versioning Protocol, as described in [RFC2518] or [RFC4918].

Web Services Description Language (WSDL): An XML format for describing network services as a set of endpoints that operate on messages that contain either document-oriented or procedure-oriented information. The operations and messages are described abstractly and are bound to a concrete network protocol and message format in order to define an endpoint. Related concrete endpoints are combined into abstract endpoints, which describe a network service. WSDL is extensible, which allows the description of endpoints and their messages regardless of the message formats or network protocols that are used.

XML Information Set (Infoset): An abstract data set that provides a consistent set of definitions for use in specifications that refer to the information in a well-formed XML document, as described in [XMLINFOSET].

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

XML namespace prefix: An abbreviated form of an **XML namespace**, as described in [XML].

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

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

1.2 References

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-FPSE] Microsoft Corporation, "FrontPage Server Extensions Remote Protocol".

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

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

[MS-SHDACCWS] Microsoft Corporation, "Shared Access Web Service Protocol".

[MS-VERSS] Microsoft Corporation, "Versions Web Service Protocol".

[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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.rfc-editor.org/rfc/rfc2616.txt

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000, http://www.rfc-editor.org/rfc/rfc2818.txt

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

[SOAP1.1] Box, D., Ehnebuske, D., Kakivaya, G., et al., "Simple Object Access Protocol (SOAP) 1.1", May 2000, http://www.w3.org/TR/2000/NOTE-SOAP-20000508/

[SOAP1.2-MTOM] Gudgin, M., Ed., Mendelsohn, N., Ed., Nottingham, M., Ed., Ruellan, H., Ed., "SOAP Message Transmission Optimization Mechanism", W3C Recommendation, January 2005, http://www.w3.org/TR/2005/REC-soap12-mtom-20050125/

[WSDL] Christensen, E., Curbera, F., Meredith, G., and Weerawarana, S., "Web Services Description Language (WSDL) 1.1", W3C Note, March 2001, http://www.w3.org/TR/2001/NOTE-wsdl-20010315

[XMLINFOSET] Cowan, J., and Tobin, R., Eds., "XML Information Set (Second Edition)", W3C Recommendation, February 2004, http://www.w3.org/TR/2004/REC-xml-infoset-20040204/

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA1] Thompson, H., Beech, D., Maloney, M., and Mendelsohn, N., Eds., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

[XOP10] Gudgin, M., Ed., Mendelsohn, N., Ed., Nottingham, M., Ed., Ruellan, H., Ed., "XML-binary Optimized Packaging", W3C Recommendation, January 2005, http://www.w3.org/TR/2005/REC-xop10-20050125/

1.2.2 Informative References

[C706] The Open Group, "DCE 1.1: Remote Procedure Call", C706, August 1997, https://www2.opengroup.org/ogsys/catalog/c706

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

[MS-OCPROTO] Microsoft Corporation, "Office Client Protocols Overview".

[RFC1738] Berners-Lee, T., Masinter, L., and McCahill, M., Eds., "Uniform Resource Locators (URL)", RFC 1738, December 1994, http://www.ietf.org/rfc/rfc1738.txt

[RFC2518] Goland, Y., Whitehead, E., Faizi, A., et al., "HTTP Extensions for Distributed Authoring - WebDAV", RFC 2518, February 1999, http://www.ietf.org/rfc/rfc2518.txt

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and Schooler, E., "SIP: Session Initiation Protocol", RFC 3261, June 2002, http://www.ietf.org/rfc/rfc3261.txt

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

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006, http://www.rfc-editor.org/rfc/rfc4648.txt

[RFC4918] Dusseault, L, Ed., "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, June 2007, http://www.rfc-editor.org/rfc/rfc4918.txt

[SOAP1.2-1/2003] Gudgin, M., Hadley, M., Mendelsohn, N., et al., "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624

[SOAP1.2-1/2007] Gudgin, M., Hadley, M., Mendelsohn, N., et al., "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)", W3C Recommendation 27, April 2007, http://www.w3.org/TR/2007/REC-soap12-part1-20070427/

[SOAP1.2/1] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 1: Messaging Framework", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part1-20030624

[SOAP1.2/2] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., and Nielsen, H.F., "SOAP Version 1.2 Part 2: Adjuncts", W3C Recommendation, June 2003, http://www.w3.org/TR/2003/REC-soap12-part2-20030624

[XMLNS-2ED] World Wide Web Consortium, "Namespaces in XML 1.0 (Second Edition)", August 2006, http://www.w3.org/TR/2006/REC-xml-names-20060816/

[XML] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Fourth Edition)", W3C Recommendation 16 August 2006, edited in place 29 September 2006, http://www.w3.org/TR/2006/REC-xml-20060816/

1.3 Overview

This protocol enables a protocol client to call a request that allows for the upload or download of file changes, along with related metadata changes, to or from a single protocol server. In addition, the protocol server processes different types of locking operation requests sent by a client that allow for uploads to be done while preventing merge conflicts on the shared resource. For more details about the different types of locking operations, see sections 3.1.4.3, 3.1.4.4, and 3.1.4.5. The protocol is a request/response stateless message exchange protocol based on **SOAP** that uses HTTP 1.1 for its transport and **SOAP Message Transmission Optimization Mechanism (MTOM)** encoding.

The protocol involves two active entities: the protocol client and the protocol server.

The protocol assumes that the protocol server stores files addressable by **URLs**. Each file has one or more partitions associated with it. These partitions can be empty or contain binary file contents, information related to file coauthoring, or contents that are specific to a file format. The data in each partition can be synchronized independently by using this protocol. For more information about the abstract data model used for synchronization, see [MS-FSSHTTPB] section 3.1.1.

A user on the protocol client or an administrator on the protocol server first creates a document. For a download file request, the protocol client sends a download request to the protocol server for all the contents of a specific partition of a file specified by a URL. If the file exists on the protocol server, the protocol server responds with the requested content or partition data. If the file does not exist, it returns a **FileNotExistsOrCannotBeCreated** error code as part of the response. For more details about the **FileNotExistsOrCannotBeCreated** error code and other error codes, see section 2.2.5.6.

For an upload file request, the protocol client sends an upload request to the protocol server indicating the data that has changed that needs to be uploaded. The protocol client can also send an upload request for changes done in the partitions associated with a file at a given URL. The server responds with success or failure for that update.

In an upload or download request, the protocol allows for locking operations to be requested by the protocol client to the protocol server. The locking operations can be for an exclusive lock or a shared lock on a file. In the case of an exclusive lock, the protocol server ensures that only one client is allowed to edit the file and responds with success in locking the file for edit. For more details about the exclusive lock operation, see section 3.1.4.5. In the case of a shared lock, the protocol server allows multiple clients to edit the coauthorable file and responds with success in sharing the lock on the coauthorable file. Depending on the type of shared lock operation, the protocol server also keeps tracks of the clients editing the file and lets the protocol client know of the coauthoring status. For more details about the coauthoring status, see section 2.2.5.1. For more details about the shared lock operations, see section 3.1.4.3 and section 3.1.4.4.

In case of failure in an exclusive lock request or shared lock request, the protocol server responds with an error code value indicating the type of error. For more details about error code types, see section 2.2.5.4.

The following diagram illustrates file upload, download, and lock requests and responses.

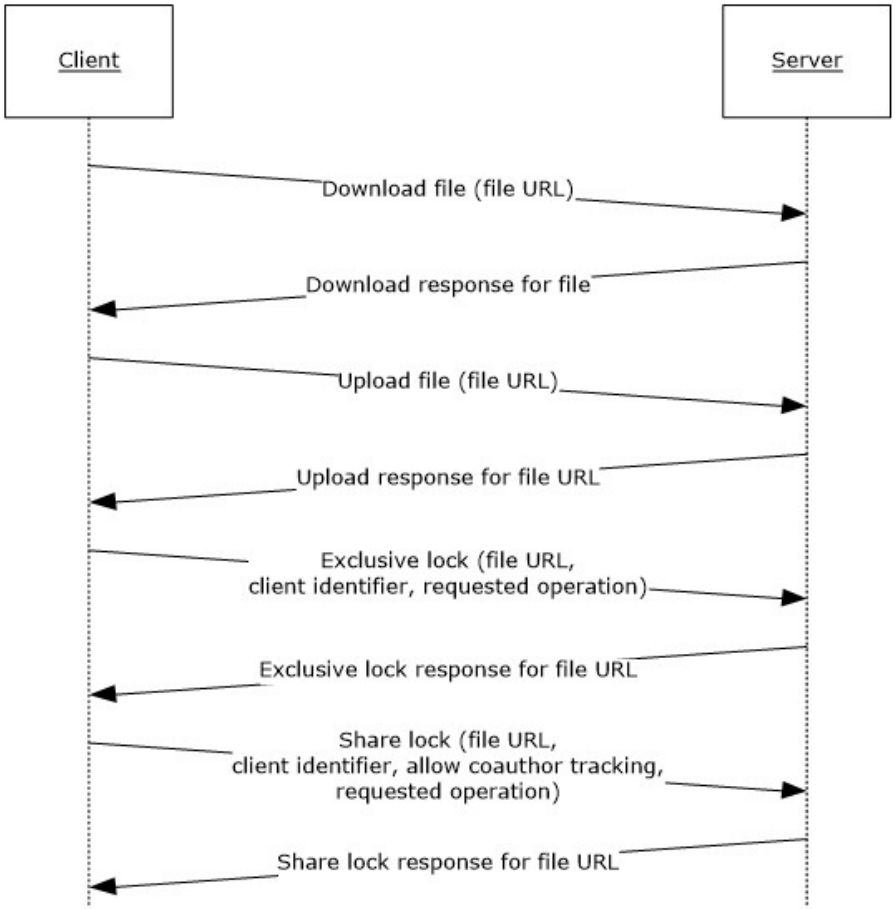

Figure 1: File upload and download to and from a server as well as lock requests to a server

The protocol provides a means to upload or download files from the protocol server without the need to retrieve the entire content or metadata for a given file every time. This is achieved by the protocol client working with the protocol as described in [MS-FSSHTTPB], which allows for incremental file synching, and the client local cache.

Because multiple clients can coauthor a file, if two or more clients sent an upload request simultaneously, all requests except the first one fail with a coherency error. Coherency failure error codes are as described in [MS-FSSHTTPB]. If the upload request fails with a coherency error, the protocol client sends a download request to get the latest changes to the file from the protocol server. The protocol client automatically merges the latest changes with its local version of the file. If the protocol client is unable to do an automatic merge, it exposes the merge conflict to the user and lets the user do a manual merge. The protocol client then sends another upload request to upload the merged version of the file to the server. The upload request succeeds if the file has not been updated by another client since the last download request made by the current client.

A typical scenario for using this protocol involves a word processing application that enables coauthoring and the multiuser editing of files that are stored on a single protocol server.

1.4 Relationship to Other Protocols

This protocol uses the Simple Object Access Protocol (SOAP) message protocol for formatting request and response messages, as described in [SOAP1.1], [SOAP1.2/1] and [SOAP1.2/2]. It transmits those messages by using **Hypertext Transfer Protocol (HTTP)**, as described in [RFC2616].

The File Synchronization via SOAP over HTTP Protocol uses SOAP over HTTP, as described in [RFC2616], as shown in the following layering diagram.

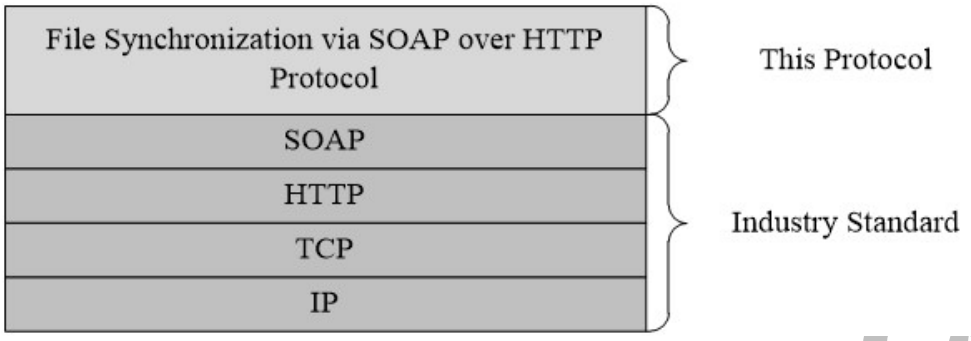

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

The protocol operates against a protocol server that is identified by a URL that is known by protocol clients. The protocol server endpoint is formed by appending "/_vti_bin/cellstorage.svc" to the URL of the protocol server. An example is http://www.contoso.com/_vti_bin/cellstorage.svc.

The protocol assumes that authentication has been performed by the underlying protocols. Authorization is dependent on the storage mechanisms of the protocol server and is not defined by this protocol.

1.6 Applicability Statement

The protocol does not control whether the upload request or download request sent by the protocol client is for all contents or for an incremental update of the file. The protocol provides means that allow for this type of specification in the request.

The advantages of this protocol can be seen when used in conjunction with the protocol as described in [MS-FSSHTTPB], which allows for upload requests of incremental updates to the contents or partition data associated with the file.

The protocol is advantageous when used for the upload and download of files that require coauthoring and are stored on a single protocol server.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

For transport, this protocol requires the following:

- Protocol servers MUST support SOAP over HTTP, as specified in [RFC2616], or HTTPS, as specified in [RFC2818].
- Protocol messages MUST be formatted as specified in [SOAP1.1] section 4.
- Protocol server MUST use MTOM encoding as specified in [SOAP1.2-MTOM].
- Protocol server faults MUST be returned either by using either HTTP status codes as specified in [RFC2616] section 10 or **SOAP faults** as specified in [SOAP1.1], section 4.4.

The **SOAPAction HTTP Header** field MUST be set to the following:

http://schemas.microsoft.com/sharepoint/soap/ICellStorages/ExecuteCellStorageRequest

2.2 Common Message Syntax

This section contains common definitions that are used by this protocol. The syntax of the definitions uses **XML schema**, as specified in [XMLSCHEMA1] and [XMLSCHEMA2], and **WSDL**, as specified in [WSDL].

2.2.1 Namespaces

This specification defines and references various **XML namespaces** using the mechanisms specified in [XMLNS]. Although this specification associates a specific **XML namespace prefix** for each XML namespace that is used, the choice of any particular XML namespace prefix is implementation-specific and is not significant for interoperability.

Prefix	Namespace URI	Reference
s	http://schemas.xmlsoap.org/soap/envelope/	[SOAP1.1]
xs	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1]
tns	http://schemas.microsoft.com/sharepoint/soap/	
i	http://www.w3.org/2004/08/xop/include	[XOP10]

2.2.2 Messages

Message	Description
Request	The **detail** element of the protocol request contains a **RequestVersion** element and a **RequestCollection** element.
Response	The **detail** element of the protocol response contains a **ResponseVersion** element and zero or one **ResponseCollection** elements.
SOAP fault	The **detail** element contains server-specific error information.

2.2.2.1 Request

The protocol request schema is specified by the following:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/sharepoint/soap/"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:i="http://www.w3.org/2004/08/xop/include">
<xs:import namespace="http://www.w3.org/2004/08/xop/include" />
 <xs:element name="Envelope">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Body">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="tns:RequestVersion" minOccurs="1" maxOccurs="1" />
              <xs:element ref="tns:RequestCollection" minOccurs="1" maxOccurs="1"/>
             </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
 </xs:element>
</xs:schema>
```

The **Body** element of each SOAP request message MUST contain a **RequestVersion** element and a **RequestCollection** element. Details about the **RequestVersion** element are specified in section 2.2.3.4, and details about the **RequestCollection** element are specified in section 2.2.3.3.

2.2.2.2 Response

The protocol response schema is specified by the following:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/sharepoint/soap/"
attributeFormDefault="unqualified" elementFormDefault="qualified"
targetNamespace="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:i="http://www.w3.org/2004/08/xop/include">
<xs:import namespace="http://www.w3.org/2004/08/xop/include" />

<xs:element name="Envelope">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Body">
    <xs:complexType>
     <xs:sequence>
      <xs:element ref="tns:ResponseVersion" minOccurs="1" maxOccurs="1" />
      <xs:element ref="tns:ResponseCollection" minOccurs="0" maxOccurs="1"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

The **Body** element of each SOAP response message MUST contain a **ResponseVersion** element and zero or more **ResponseCollection** elements. Details about the **ResponseVersion** element are specified in section 2.2.3.7, and details about the **ResponseCollection** element are specified in section 2.2.3.6.

2.2.2.3 SOAP Fault

This protocol enables a server to notify a client about unhandled and unexpected server-side exceptions by using a SOAP fault response. In a SOAP fault, the **detail** element contains server-specific error information. The fault codes returned as part of the SOAP Fault element are standard fault codes as specified in [SOAP1.1].

The following schema specifies the structure of the detail element in the SOAP fault used by this protocol:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/sharepoint/soap/"
attributeFormDefault="unqualified" elementFormDefault="qualified"
           targetNamespace="http://schemas.microsoft.com/sharepoint/soap/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:i="http://www.w3.org/2004/08/xop/include">

<xs:element name="detail">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="ErrorString" type="xs:string" minOccurs="0" maxOccurs="1"/>
      <xs:element name="ErrorCode" type="xs:string" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>

</xs:schema>
```

ErrorString: A string specifying the description of the error.

ErrorCode: A string specifying an operation-specific error code. Error codes are defined with the specific operations that return SOAP faults.
Request-specific or subrequest-specific error messages are communicated as part of the **Response** element or **SubResponse** element that is in a response message and not in a SOAP fault message.

2.2.3 Elements

The following table summarizes the set of common XML schema element definitions defined by this specification. XML schema element definitions that are specific to a particular operation are described with the operation.

Element	Description
Include	The **Include** element is used for encapsulating and sending large amounts of binary data. The details about when an **Include** element is sent as part of a cell storage service request message and the details about when an **Include** element is sent as part of a cell storage service response message are specified in section 2.2.3.1.
Request	The **Request** element contains a cell storage service request that is specific to a URL for the file with a unique identifier that identifies the request by using a **request token**.
RequestCollection	The **RequestCollection** element contains a collection of **Request** elements.
RequestVersion	The **RequestVersion** element specifies the version-specific information about the cell storage service request message.
Response	The **Response** element contains a cell storage service response that is specific to a URL for the file with a mapping response for the respective **RequestToken**.

ResponseCollection	The **ResponseCollection** element contains a collection of **Response** elements.
ResponseVersion	The **ResponseVersion** element specifies the version-specific information about the cell storage service response message.
SubRequest	The **SubRequest** element specifies subrequests within a **Request** element.
SubRequestData	The **SubRequestData** element specifies the data required for processing the subrequest.
SubResponse	The **SubResponse** element specifies the corresponding response for the subrequest.
SubResponseData	The **SubResponseData** element contains any data requested as part of the subrequest.

2.2.3.1 Include

The **Include** element is used for encapsulating and sending large amounts of binary data. The **Include** element MUST be sent only as part of the **SubRequestData** element in a cell storage service request message if the following condition is true:

- The **Type** attribute specified in the corresponding **SubRequest** element is set to a value of "Cell".

If the **Include** element is sent when the preceding condition is not **true**, the server MUST ignore it.

The **SubRequestData** element is specified in section 2.2.3.9. The **SubResponseData** element is specified in section 2.2.3.11. The **Type** attribute is specified in section 2.2.4.2. The **Include** element and the schema of the **Include** element are as specified in [XOP10] section 2.1. XML-binary Optimized Packaging (XOP) provides a means for more efficiently serializing an **XML Information Set (Infoset)** that has content types as specified in [XMLINFOSET].

2.2.3.2 Request

Each **Request** element maps to a synchronization request for a specific file. Each file MUST be uniquely identified by a URL for the file. The synchronization request for a file or the file's metadata is divided into subrequests. Each **Request** element MUST have one or more **SubRequest** elements.

```
<xs:element name="Request">
  <xs:complexType>
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:element name="SubRequest" type="tns:SubRequestElementGenericType" />
</xs:sequence>
    <xs:attribute name="Url" type="xs:string" use="required"/>
    <xs:attribute name="Interval" type="xs:nonNegativeInteger" use="optional"/>
    <xs:attribute name="MetaData" type="xs:integer" use="optional"/>
    <xs:attribute name="RequestToken" type="xs:nonNegativeInteger" use="required" />
  </xs:complexType>
</xs:element>
```

SubRequest: A **SubRequestElementGenericType** that specifies the type of subrequest for the URL for the file and input parameters needed by a protocol server for processing the subrequest. The **SubRequest** element is defined in section 2.2.3.8. The **SubRequestElementGenericType** is defined in section 2.2.4.2.

Url: A string that specifies the URL for the file that uniquely identifies the file whose contents or metadata contents are requested for uploading to the server or downloading from the server. The **Url** attribute MUST be specified for each **Request** element. The string specifying the file MUST NOT be an empty string.

Interval: A nonnegative integer that specifies the interval, in seconds, that the protocol client will repeat this request. This value is used by protocol servers when throttling requests.<1>

MetaData: A 32-bit value that specifies information about the scenario and urgency of the request. This value is used by protocol servers when throttling requests.<2>

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
A	B	C	D		E		F		G	H	I		J		K	L	M		Reserved												

A – Request Source (1-bit): A bit that specifies if the request was triggered by a user action or an automated process.

Value	Meaning
0	Automatically generated request.
1	User action–generated request.

B – User Presence (1-bit): A bit that specifies if the request was triggered when the protocol client considered the user present or absent.

Value	Meaning
0	User absent.
1	User present.

C- Cache Status (1-bit): A bit that specifies if the protocol client has a local cache.

Value	Meaning
0	The protocol client has no local cache.
1	The protocol client has a local cache.

D- Coauthoring Presence (2-bits): A 2-bit unsigned integer that specifies the protocol client's awareness of other protocol clients' coauthoring.

Value	Meaning
0	No other authors.
1	Other authors detected far from where the protocol client is working.
2	Other authors detected in the same section where the protocol client is working.
3	Other authors detected outside the same page where the protocol client is working.

E – Response Data View (2-bits): A 2-bit unsigned integer that specifies when the protocol client will present the response data to the user.

Value	Meaning
0	Future user action in a new session.
1	Future user action in the current session.
2	Automatically after a short delay.
3	Immediately.

F – Time Since User View (2-bits): A 2-bit unsigned integer that specifies how long it has been since the user viewed data referenced by the **Url**.

Value	Meaning
0	Recently.
1	Within one week.
2	Within one month.
3	More than one month.

G – Document Active State (1-bit): A bit that specifies if the protocol client considered the **URL** an active or inactive session at the time the request was triggered.

Value	Meaning
0	Active.
1	Inactive.

H – Data Type (1-bit): A bit that specifies if the request data is recoverable by the protocol client.

Value	Meaning
0	The data is content and requires permanent reliable storage.
1	The data is recoverable. Less reliable stores may be used.

I – Network Cost Level (2-bits): A 2-bit unsigned integer that specifies the cost of the network used to make the request.

Value	Meaning
0	Unspecified or unknown.
1	Low-cost network.
2	Medium-cost network.
3	High-cost network.

J – Network Quota State (2-bits): A 2-bit unsigned integer that specifies the network quota state when a metered network is being used to make the request.

Value	Meaning
0	No quota system.
1	The quota system is active; the quota consumed is one half or less.
2	The quota system is active; the quota consumed is from one half through three quarters.
3	The quota system is active; the quota consumed is near or over the limit.

K – Power State (1-bit): A bit that specifies the protocol client hardware power state.

Value	Meaning
0	Powered by a continuous power source.
1	Powered by battery.

L – Application State (1-bit): A bit that specifies the protocol client sync state.

Value	Meaning
0	Protocol client is functioning normally.
1	Protocol client is not functioning normally.

M– License Type (2-bits): A 2-bit unsigned integer that specifies the client's license type.

Value	Meaning
0	License type is unspecified or unknown.
1	License type is full.
2	License type is limited.
3	License type is free.

Reserved (13-bits): Reserved. MUST be zero.

RequestToken: A nonnegative integer that specifies the request token that uniquely identifies the **Request** element in a cell storage service request. The **RequestToken** MUST be set to a nonnegative integer value, with a minimum allowed value of zero, and a maximum allowed value of 4,294,967,295. For each new **Request** element in a cell storage service request that needs to be sent to the protocol server, **RequestToken** is incremented by the protocol client. **RequestToken** is reset to 1 by the protocol client for a new cell storage service request. The one-to-one mapping between the **Response** element and the **Request** element MUST be maintained by using **RequestToken**. A **RequestToken** value MUST be specified for each **Request** element.

Errors that occur during the parsing of the **Request** element cause the error code value to be set to "InvalidArgument". The protocol server MUST send the error code as an error code attribute in the **Response** element. The **ErrorCode** attribute is defined in section 2.2.3.5. Depending on the other types of errors, the error code for that type MUST be returned by the protocol server. Generic error code types are defined in section 2.2.5.5.

2.2.3.3 RequestCollection

The **RequestCollection** element MUST contain one or more **Request** elements. Each **Request** element is a cell storage service request for a unique URL for the file. The **Request** element is specified in section 2.2.3.2.

```
<xs:element name="RequestCollection">
    <xs:complexType>
      <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element ref="tns:Request" />
      </xs:sequence>
      <xs:attribute name="CorrelationId" type="tns:guid" use="required" />
    </xs:complexType>
</xs:element>
```

Request: A complex type that specifies the upload or download requests specific to a file. The **Request** element is specified in section 2.2.3.2.

CorrelationId: A **guid** that specifies a unique identifier that is generated by the client for every request message it sends. **CorrelationId** is used by the protocol server when logging server events. The logging of events with the **CorrelationId** value helps in the correlation of the server log traces to the specific client request. **CorrelationId** is of type **guid**. The **guid** type is defined in section 2.2.5.7.

Errors that occur during the parsing of the **RequestCollection** element MUST return a SOAP fault message. The SOAP fault message is defined in section 2.2.2.3.

2.2.3.4 RequestVersion

The **RequestVersion** element contains version-specific information for this cell storage service request message.

```
<xs:element name="RequestVersion" type="tns:VersionType" />
```

VersionType is specified in section 2.2.4.7.

Errors that occur because a version is not supported cause an **IncompatibleVersion** error code value to be set and sent as part of the **ResponseVersion** element. The **IncompatibleVersion** error code is defined in section 2.2.5.5. The **ResponseVersion** element is defined in section 2.2.3.7.

2.2.3.5 Response

For each **Request** element that is part of a cell storage service request, there MUST be a corresponding **Response** element in a cell storage service response. Each **Response** element MUST contain one or more **SubResponse** elements.

```
<xs:element name="Response">
  <!--Allows for the numbers to be displayed between the SubResponse elements-->
  <xs:complexType mixed="true">
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:element name="SubResponse" type="tns:SubResponseElementGenericType" />
    </xs:sequence>
    <xs:attribute name="Url" type="xs:string" use="required"/>
    <xs:attribute name="RequestToken" type="xs:nonNegativeInteger" use="required" />
    <xs:attribute name="HealthScore" type="xs:integer" use="required"/>
    <xs:attribute name="ErrorCode" type="tns:GenericErrorCodeTypes" use="optional" />
    <xs:attribute name="ErrorMessage" type="xs:string" use="optional"/>
  </xs:complexType>
</xs:element>
```

SubResponse: A **SubResponseElementGenericType** that specifies the response given by the protocol server for the corresponding subrequest requested as part of the **SubRequest** element. The **SubResponseElementGenericType** is defined in section 2.2.4.5. The **SubResponse** element is defined in section 2.2.3.10. The **SubRequest** element is defined in section 2.2.3.8.

Url: A string that specifies the URL for the file that uniquely identifies the file whose response is being generated. The **Url** attribute MUST be specified for each **Response** element.

RequestToken: A nonnegative integer that specifies the request token that uniquely identifies the **Request** element whose response is being generated. The **Request** element is defined in section 2.2.3.2. The one-to-one mapping between the **Response** element and the **Request** element MUST be maintained by using the request token. The **RequestToken** MUST be specified for each **Response** element.

HealthScore: An integer that specifies the server performance health, expressed as an integer ranging from 0 through 10, where a score of 0 specifies excellent server health and a score of 10 specifies very poor server health. The health score provides hints that help the protocol client throttle the sending of cell storage service requests, depending on the server health.

ErrorCode: A **GenericErrorCodeTypes** that specifies an error code value indicating the type of error that occurred during the processing of the mapping **Request** element. **GenericErrorCodeTypes** is defined in section 2.2.5.6. This attribute MUST be present only if any of the following is true:

- The **Url** attribute of the corresponding **Request** element does not exist or is an empty string.<3> (The **Url** attribute and the **Request** element are defined in section 2.2.3.2.)
- The **RequestToken** attribute of the corresponding **Request** element is an empty string<4>. (The **RequestToken** attribute and the **Request** element are defined in section 2.2.3.2.)
- An exception occurred during the processing of a subrequest that was not entirely handled by the subrequest processing logic.

ErrorMessage: A string that specifies a description of the error message and also specifies information about what was expected by the server. This attribute MUST be present when the **ErrorCode** attribute is present and does not equal "Success".

2.2.3.6 ResponseCollection

The **ResponseCollection** element MUST contain one or more **Response** elements. This element is used to send responses for each of the file upload and download requests.

```
<xs:element name="ResponseCollection">
    <xs:complexType>
      <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element ref="tns:Response" />
      </xs:sequence>
      <xs:attribute name="WebUrl" type="xs:string" use="required"/>
    </xs:complexType>
</xs:element>
```

Response: A complex type that specifies the response given by the protocol server for each corresponding request received as part of the **Request** element. The **Response** element is defined in section 2.2.3.5.

WebUrl: A string that specifies the **absolute URL** for the protocol server that uniquely identifies the protocol server that processed the cell storage service request and generated this corresponding cell storage service response message. The **WebUrl** attribute MUST be specified for each **ResponseCollection** element.

2.2.3.7 ResponseVersion

The **ResponseVersion** element contains version-specific information for this cell storage service response message.

```
<xs:element name="ResponseVersion">
 <xs:complexType>
  <xs:complexContent>
   <xs:extension base="tns:VersionType">
    <xs:attribute name="ErrorCode" type="tns:GenericErrorCodeTypes" use="optional" />
    <xs:attribute name="ErrorMessage" type="xs:string" use="optional" />
     </xs:extension>
  </xs:complexContent>
 </xs:complexType>
</xs:element>
```

VersionType: A complex type that is specified in section 2.2.4.7.

ErrorCode: A **GenericErrorCodeTypes** that specifies an error code value indicating the type of error that occurred during the processing of the mapping **RequestVersion** element. **GenericErrorCodeTypes** is defined in section 2.2.5.6. The **RequestVersion** element is defined in section 2.2.3.4. This attribute MUST be present only if any one of the following is true:

- The **Version** attribute of the **RequestVersion** element of the request message has a value that is less than 2. The **Version** attribute is defined in section 2.2.4.7. The **RequestVersion** element is defined in section 2.2.3.7.
- The protocol server identified by the **WebUrl** attribute of the **ResponseCollection** element does not exist or is not available. The **WebUrl** attribute and the **ResponseCollection** element are defined in section 2.2.3.6.

- The user does not have permission to issue a cell storage service request to the file identified by the **Url** attribute of the **Request** element. The **Url** attribute and the **Request** element are defined in section 2.2.3.2.
- This protocol is not enabled on the protocol server.

2.2.3.8 SubRequest

The **SubRequest** element defines the subrequest for a specific URL for the file. **SubRequestElementGenericType** is defined in section 2.2.4.2.

```
<xs:element name="SubRequest" type="tns:SubRequestElementGenericType" />
```

Errors that occur during the parsing or processing of the **SubRequest** element are returned as error codes in the **SubResponse** element. The **SubResponse** element is part of the cell storage service response message. The **SubResponse** element is defined in section 2.2.3.10.

2.2.3.9 SubRequestData

The **SubRequestData** element further qualifies the **SubRequest** element. **SubRequestData** specifies any data or input parameters that are used in processing the **SubRequest** element. **SubRequestDataGenericType** is defined in section 2.2.4.1.

```
<xs:element name="SubRequestData" minOccurs="0" maxOccurs="1"
type="tns:SubRequestDataGenericType" />
```

The conditions under which a **SubRequestData** element MUST be sent as part of the **SubRequest** element are specified in section 2.2.4.2. The conditions under which a **SubRequestData** element MUST NOT be sent as part of a **SubRequest** element are specified in section 2.2.4.2.

2.2.3.10 SubResponse

Within a **Response** element for each **SubRequest** element that is in a **Request** element of a cell storage service request message, there MUST be a corresponding **SubResponse** element. The **SubResponse** element specifies the success or failure of processing the corresponding **SubRequest** element. In the case of success, the **SubResponse** element specifies the information requested as part of the **SubRequest** element. In the case of failure, the error code attribute in a **SubResponse** element specifies the error code result returned during processing of the **SubRequest** element. The error code attribute is defined in section 2.2.4.6.

SubResponseElementGenericType is defined in section 2.2.4.5. Each **SubResponse** element MUST have zero or one **SubResponseData** elements. The **SubResponseData** element is defined in section 2.2.3.11.

```
<xs:element name="SubResponse" type="tns:SubResponseElementGenericType" />
```

2.2.3.11 SubResponseData

A **SubResponseData** element is in a **SubResponse** element of a cell storage service response. The **SubResponseData** element specifies the responses for specific requests in the corresponding **SubRequestData** element. **SubResponseDataGenericType** is defined in section 2.2.4.4.

```
<xs:element name="SubResponseData" minOccurs="0" maxOccurs="1"
type="tns:SubResponseDataGenericType" />
```

The conditions under which a **SubResponseData** element MUST be sent as part of the **SubResponse** element are specified in section 2.2.4.5.

2.2.4 Complex Types

The following table summarizes the set of common XML schema complex type definitions defined by this specification. XML schema complex type definitions that are specific to a particular operation are described with the operation.

Complex type	Description
SubRequestDataGenericType	A generic type definition of data that is part of a cell storage service subrequest.
SubRequestElementGenericType	A generic type definition of a cell storage service subrequest.
SubRequestType	The base type definition of a cell storage service subrequest. **SubRequestType** is used to extend **SubRequestElementGenericType**.
SubResponseDataGenericType	A generic type definition of data that is part of a cell storage service subresponse.
SubResponseElementGenericType	A generic type definition of a cell storage service subresponse.
SubResponseType	The base type definition of a cell storage service subresponse. **SubResponseType** is used to extend **SubResponseElementGenericType**.
VersionType	The version of the message.

2.2.4.1 SubRequestDataGenericType

The **SubRequestDataGenericType** complex type contains information about data or input parameters used in processing a cell storage service subrequest. **SubRequestDataGenericType** provides a generic subrequest data type definition.

```
<xs:complexType name="SubRequestDataGenericType" mixed="true">
    <xs:choice>
      <xs:element ref="i:Include" minOccurs="0" maxOccurs="1" />
    </xs:choice>
    <xs:attributeGroup ref="tns:SubRequestDataOptionalAttributes" />
</xs:complexType>
```

Include: A complex type, as specified in [XOP10] section 2.1, that is used for encapsulating and sending large amounts of binary data. The **Include** element is specified in section 2.2.3.1.

SubRequestDataOptionalAttributes: An attribute group that specifies the set of attributes that are provided for a **SubRequestData** element. **SubRequestDataOptionalAttributes** is defined in section 2.2.8.1.

Depending on the type of cell storage service subrequest, **SubRequestDataGenericType** MUST take one of the forms described in the following table.

Complex type	Description
CellSubRequestDataType	The type definition for cell subrequest data.
CoauthSubRequestDataType	The type definition for coauthoring subrequest data.
ExclusiveLockSubRequestDataType	The type definition for exclusive lock subrequest data.
SchemaLockSubRequestDataType	The type definition for schema lock subrequest data.
EditorsTableSubRequestDataType	The type definition for editors table subrequest data.

CellSubRequestDataType is specified in section 2.3.1.1. **CoauthSubRequestDataType** is specified in section 2.3.1.5. **ExclusiveLockSubRequestDataType** is specified in section 2.3.1.9. **SchemaLockSubRequestDataType** is specified in section 2.3.1.13. **EditorsTableSubRequestDataType** is specified in section 2.3.1.23.

The referenced **Include** element MUST be sent as part of the **SubRequestData** element in a cell storage service request message only if the **Type** attribute specified in the corresponding **SubRequest** element is set to a value of "Cell" and this cell subrequest is for the upload of the file's binary contents or metadata contents. The **Type** attribute is specified in section 2.2.4.2.

2.2.4.2 SubRequestElementGenericType

The **SubRequestElementGenericType** complex type contains information about a subrequest in a cell storage service request message. **SubRequestElementGenericType** provides a generic subrequest type definition. The **SubRequestType** definition from which **SubRequestElementGenericType** is extended is defined in section 2.2.4.3.

```
<xs:complexType name="SubRequestElementGenericType" mixed="true">
 <xs:complexContent>
  <xs:extension base="tns:SubRequestType">
   <xs:all>
    <xs:element name="SubRequestData" minOccurs="0" maxOccurs="1"
type="tns:SubRequestDataGenericType" />
   </xs:all>
   <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required" />
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

SubRequestData: A **SubRequestDataGenericType** that specifies the data or input parameters needed in processing the subrequest specified as part of the **SubRequest** element in a cell storage service request message. The **SubRequestDataGenericType** is defined in section 2.2.4.1. The **SubRequest** element is defined in section 2.2.3.8.

Type: A **SubRequestAttributeType** that specifies the type of subrequest to be processed. The subrequest is specified as part of the **SubRequest** element in a cell storage service request message. **SubRequestAttributeType** is defined in section 2.2.5.11. Depending on the type of subrequest, **SubRequestElementGenericType** MUST take one of the forms described in the following table.

Complex type	Description
CellSubRequestType	The type definition for a cell subrequest when the **Type** attribute is set to "Cell".
CoauthSubRequestType	The type definition for a coauthoring subrequest when the **Type** attribute is set to "Coauth".
ExclusiveLockSubRequestType	The type definition for an exclusive lock subrequest when the **Type** attribute is set to "ExclusiveLock".

SchemaLockSubRequestType	The type definition for a schema lock subrequest when the **Type** attribute is set to "SchemaLock".
ServerTimeSubRequestType	The type definition for a server time subrequest when the **Type** attribute is set to "ServerTime".
WhoAmISubRequestType	The type definition for a Who Am I subrequest when the **Type** attribute is set to "WhoAmI".
EditorsTableSubRequestType	The type definition for an Editors Table subrequest.
GetDocMetaInfoSubRequestType	The type definition for a Get Doc Meta Info subrequest when the **Type** attribute is set to "GetDocMetaInfo".
GetVersionsSubRequestType	The type definition for a Get Versions subrequest when the Type attribute is set to "GetVersions".

CellSubRequestType is specified in section 2.3.1.2. **CoauthSubRequestType** is specified in section 2.3.1.6. **ExclusiveLockSubRequestType** is specified in section 2.3.1.10. **SchemaLockSubRequestType** is specified in section 2.3.1.14. **ServerTimeSubRequestType** is specified in section 2.3.1.17. **WhoAmISubRequestType** is specified in section 2.3.1.20. **EditorsTableSubRequestType** is specified in section 2.3.1.23. **GetDocMetaInfoSubRequestType** is specified in section 2.3.1.26. **GetVersionsSubRequestType** is specified in section 2.3.1.31.

The **SubRequestData** element MUST be sent as part of the **SubRequest** element in a cell storage service request message if one of the following conditions is **true**:

- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "Coauth"
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "ExclusiveLock"
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "EditorsTable"
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "SchemaLock"
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "Cell" and the cell subrequest is for uploading or downloading the file's binary contents or metadata contents.

The **SubRequestData** element MUST NOT be sent as part of the **SubRequest** element in a cell storage service request message if one of the following conditions is **true**:

- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "WhoAmI".
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "ServerTime".
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "GetDocMetaInfo".
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "GetVersions".

2.2.4.3 SubRequestType

The **SubRequestType** complex type contains information about a basic cell storage service subrequest. The **SubRequestType** is used as the base complex type to extend the **SubRequestElementGenericType**. The **SubRequestElementGenericType** takes one of the following forms: **CellSubRequestType**, **CoauthSubRequestType**, **SchemaLockSubRequestType**, **ExclusiveLockSubRequestType**, **WhoAmISubRequestType**, **ServerTimeSubRequestType**, or **GetDocMetaInfoSubRequestType**. **CellSubRequestType** is specified in section 2.3.1.2. **CoauthSubRequestType** is specified in section 2.3.1.6. **ExclusiveLockSubRequestType** is specified in section 2.3.1.10. **SchemaLockSubRequestType** is specified in section 2.3.1.14. **ServerTimeSubRequestType** is specified in section 2.3.1.17. **WhoAmISubRequestType** is

specified in section 2.3.1.20. **EditorsTableSubRequestType** is specified in section 2.3.1.24. **GetDocMetaInfoSubRequestType** is specified in section 2.3.1.26.

```
<xs:complexType name="SubRequestType">
 <xs:attribute name="SubRequestToken" type="xs:nonNegativeInteger" use="required"/>
 <xs:attribute name="DependsOn" type="xs:nonNegativeInteger" use="optional" />
 <xs:attribute name="DependencyType" type="tns:DependencyTypes" use="optional" />
</xs:complexType>
```

SubRequestToken: A nonnegative integer that specifies a number that uniquely identifies the **SubRequest** element in a cell storage service request. **SubRequestToken** MUST be set to an unsigned integer value with a minimum allowed value of 0 and maximum allowed value of 4,294,967,295. For each new **SubRequest** element in a cell storage service request that needs to be sent to the protocol server, **SubRequestToken** gets incremented by the protocol client. **SubRequestToken** is reset to 1 by the protocol client for a new cell storage service request. The mapping subresponse that gets generated for the subrequest references **SubRequestToken** to indicate that it is the response for that subrequest. The **SubRequestToken** attribute MUST be specified for any type of **SubRequest** element.

DependsOn: A nonnegative integer that specifies the **SubRequestToken** of the **SubRequest** element that this specific subrequest is dependent on.

DependencyType: A **DependencyTypes** that specifies a value indicating the type of dependency between the **SubRequest** element and the **SubRequest** that is associated with the **SubRequestToken** indicated in the **DependsOn** attribute. **DependencyTypes** is specified in section 2.2.5.3.

2.2.4.4 SubResponseDataGenericType

The **SubResponseDataGenericType** complex type contains information requested as part of a cell storage service subrequest. **SubResponseDataGenericType** provides a generic subresponse data type definition.

```
<xs:complexType name="SubResponseDataGenericType" mixed="true">
  <xs:all>
     <xs:element ref="i:Include" minOccurs="0" maxOccurs="1" />
     <xs:element name="DocProps" minOccurs="0" maxOccurs="1"
type="tns:GetDocMetaInfoPropertySetType"/>
     <xs:element name="FolderProps" minOccurs="0" maxOccurs="1"
type="tns:GetDocMetaInfoPropertySetType"/>
  </xs:all>
  <xs:attributeGroup ref="tns:SubResponseDataOptionalAttributes" />
</xs:complexType>
```

Include: A complex type, as specified in [XOP10] section 2.1, used for encapsulating and sending large amounts of binary data. The referenced **Include** element is specified in section 2.2.3.1.

DocProps: An element of type **GetDocMetaInfoPropertySetType** (section 2.3.1.28) that specifies metadata properties pertaining to the server file.

FolderProps: An element of type **GetDocMetaInfoPropertySetType** (section 2.3.1.28) that specifies metadata properties pertaining to the parent directory of the server file.

SubResponseDataOptionalAttributes: An attribute group that specifies the set of attributes that are provided for a **SubResponseData** element that is part of a cell storage service response message. **SubResponseDataOptionalAttributes** is defined in section 2.2.8.2. The **SubResponseData** element is defined in section 2.2.3.11.

SubResponseDataGenericType MUST take one of the forms described in the following table, depending on the type of cell storage service subrequest for which this cell storage service **SubResponseData** is sent by the protocol server.

Complex type	Description
CellSubResponseDataType	The type definition for cell subresponse data.
CoauthSubResponseDataType	The type definition for coauthoring subresponse data.
ExclusiveLockSubResponseDataType	The type definition for exclusive lock subresponse data.
SchemaLockSubResponseDataType	The type definition for schema lock subresponse data.
ServerTimeSubResponseDataType	The type definition for server time subresponse data.
WhoAmISubResponseDataType	The type definition for Who Am I subresponse data.
GetDocMetaInfoSubResponseDataType	The type definition for Get Doc Meta Info subresponse data.

CellSubResponseDataType is specified in section 2.3.1.3. **CoauthSubResponseDataType** is specified in section 2.3.1.7. **ExclusiveLockSubResponseDataType** is specified in section 2.3.1.11. **SchemaLockSubResponseDataType** is specified in section 2.3.1.15. **ServerTimeSubResponseDataType** is specified in section 2.3.1.18. **WhoAmISubResponseDataType** is specified in section 2.3.1.21. **GetDocMetaInfoSubResponseDataType** is specified in section 2.3.1.27.

The referenced **Include** element MUST be sent as part of the **SubResponseData** element in a cell storage service response message only if the **Type** attribute specified in the corresponding **SubRequest** element is set to a value of "Cell" and this cell subrequest is for the download of the file's binary contents or metadata contents. The **Type** attribute is specified in section 2.2.4.2.

2.2.4.5 SubResponseElementGenericType

The **SubResponseElementGenericType** complex type contains information about the success or failure in processing the cell storage service subrequest. In the case of success, it contains information requested as part of the subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for the subrequest. The **ErrorCode** attribute is specified in section 2.2.4.6. **SubResponseElementGenericType** provides a generic subresponse type definition. The **SubResponseType** definition from which **SubResponseElementGenericType** is extended is defined in section 2.2.4.6.

```
<xs:complexType name="SubResponseElementGenericType">
 <xs:complexContent>
  <xs:extension base="tns:SubResponseType">
   <xs:sequence>
    <xs:element name="SubResponseData" minOccurs="0" maxOccurs="1"
type="tns:SubResponseDataGenericType" />
    <xs:element name="SubResponseStreamInvalid" minOccurs="0" maxOccurs="1" />
    <xs:element ref="GetVersionsResponse" minOccurs="0" maxOccurs="1" />
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

SubResponseData: A **SubResponseDataGenericType** that specifies the response from the protocol server providing the data or file metadata information requested as part of the subrequest. **SubResponseDataGenericType** is defined in section 2.2.4.4.

SubResponseStreamInvalid: An empty element that indicates the binary data in the **SubResponseData** is not valid because of a server race condition. The protocol client can retry the request if it sees this error indication element.

GetVersionsResponse: An element that specifies information about the success or failure in processing the **GetVersions** subrequest.Depending on the **Type** attribute specified in the **SubRequest** element, the **SubResponseElementGenericType** MUST take one of the forms described in the following table.

Complex type	Description
CellSubResponseType	The type definition for a cell subresponse.
CoauthSubResponseType	The type definition for a coauthoring subresponse.
ExclusiveLockSubResponseType	The type definition for an ExclusiveLock subresponse.
SchemaLockSubResponseType	The type definition for a SchemaLock subresponse.
ServerTimeSubResponseType	The type definition for a server time subresponse.
WhoAmISubResponseType	The type definition for a Who Am I subresponse.
EditorsTableSubResponseType	The type definition for an Editors Table subresponse.
GetDocMetaInfoSubResponseType	The type definition for a Get Doc Meta Info subresponse.
GetVersionsSubResponseType	The type definition for a Get Versions subresponse.

CellSubResponseType is specified in section 2.3.1.4. **CoauthSubResponseType** is specified in section 2.3.1.8. **ExclusiveLockSubResponseType** is specified in section 2.3.1.12. **SchemaLockSubResponseType** is specified in section 2.3.1.16. **ServerTimeSubResponseType** is specified in section 2.3.1.19. **WhoAmISubResponseType** is specified in section 2.3.1.22. **EditorsTableSubResponseType** is specified in section 2.3.1.25. **GetDocMetaInfoSubResponseType** is specified in section 2.3.1.30.

The **SubResponseData** element MUST be sent as part of the **SubResponse** element in a cell storage service response message if the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success" and one of the following conditions is **true**:

- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "Cell".
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "ExclusiveLock".
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "SchemaLock".
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "ServerTime".
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "Coauth".
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "WhoAmI".
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "GetDocMetaInfo".

The **Type** attribute is specified in section 2.2.4.2. The protocol server sets the value of the **ErrorCode** attribute to "Success" only if the protocol server succeeds in processing the cell storage service subrequest. The **ErrorCode** attribute is specified in section 2.2.4.6.

2.2.4.6 SubResponseType

The **SubResponseType** complex type contains information about a basic cell storage service subresponse. The **SubResponseType** is used as the base complex type to extend **SubResponseElementGenericType**. The **SubResponseElementGenericType** takes one of the following forms: **CellSubResponseType**, **CoauthSubResponseType**, **SchemaLockSubResponseType**, **ExclusiveLockSubResponseType**, **ServerTimeSubResponseType**, **WhoAmISubResponseType, EditorsTableSubResponseType** or **GetDocMetaInfoSubResponseType.CellSubResponseType** is specified in section 2.3.1.4. **CoauthSubResponseType** is specified in section 2.3.1.8. **ExclusiveLockSubResponseType** is specified in section 2.3.1.12. **SchemaLockSubResponseType** is specified in section 2.3.1.16. **ServerTimeSubResponseType** is specified in section 2.3.1.19. **WhoAmISubResponseType** is specified in section 2.3.1.22. **EditorsTableSubResponseType** is specified in section 2.3.1.25. **GetDocMetaInfoSubResponseType** is specified in section 2.3.1.30. **GetVersionsSubResponseType** is specified in section 2.3.1.32.

```
<xs:complexType name="SubResponseType">
   <xs:attribute name="SubRequestToken" type="xs:nonNegativeInteger" use="required"/>
   <xs:attribute name="ErrorCode" type="tns:ErrorCodeTypes" use="required" />
   <xs:attribute name="HResult" type="xs:integer" use="required"/>
   <xs:attribute name="ErrorMessage" type="xs:string" use="optional"/>
</xs:complexType>
```

SubRequestToken: A nonnegative integer that specifies a number that uniquely identifies the **SubRequest** element whose subresponse is being generated as part of the **SubResponse** element. The mapping subresponse that gets generated for the subrequest references the **SubRequestToken** to indicate that it is the response for that subrequest. The **SubRequestToken** attribute MUST be specified for a **SubResponse** element. The **SubResponse** element is defined in section 2.2.3.10. The **SubRequest** element is defined in section 2.2.3.8.

ErrorCode: An **ErrorCodeTypes** that specifies an error code value indicating the type of error that occurred during the processing of the corresponding **SubRequest** element. The **SubRequest** element is defined in section 2.2.3.8. **ErrorCodeTypes** is defined in section 2.2.5.4. The **ErrorCode** attribute MUST be specified for a **SubResponse** element.

HResult: An integer that specifies an error code specific to the subrequest that failed and that gives more hints about the cause of failure.

ErrorMessage: A string that specifies a description of the error code value and also provides additional information related to the error code. If the error code value is set to "FileAlreadyLockedOnServer", the protocol server returns the user name of the client that is currently holding the lock on the file.

2.2.4.7 VersionType

The **VersionType** complex type contains information about the version of the cell storage service message.

```
<xs:complexType name="VersionType">
   <xs:attribute name="Version" type="tns:VersionNumberType" use="required" />
   <xs:attribute name="MinorVersion" type="tns:MinorVersionNumberType" use="required" />
</xs:complexType>
```

Version: A **VersionNumberType** that specifies the major version number. **VersionNumberType** is defined in section 2.2.5.12.

MinorVersion: A **MinorVersionNumberType** that specifies the minor version number. **MinorVersion** is reserved. **MinorVersionNumberType** is defined in section 2.2.5.10.

2.2.5 Simple Types

The following table summarizes the set of common XML schema simple type definitions defined by this specification. XML schema simple type definitions that are specific to a particular operation are described with the operation.

Simple type	Description
CoauthStatusType	The type of the **CoauthStatus** attribute of a coauthoring subrequest or exclusive lock subrequest. **CoauthStatusType** is an enumeration of the coauthoring statuses of a file.
DependencyCheckRelatedErrorCodeTypes	An enumeration of error codes for errors that occur during subrequest dependency checks.
DependencyTypes	The type of the **DependencyType** attribute of the **SubRequest** element, which is part of the cell storage service request message. **DependencyTypes** is an enumeration of all the dependency conditions supported for the execution of a subrequest.
ErrorCodeTypes	The type of the **ErrorCode** attribute of the **SubResponse** element. **ErrorCodeTypes** is a union of all the possible error codes, including success, for the cell storage service response message.
ExclusiveLockReturnReasonTypes	An enumeration of values that indicate an exclusive lock granted on a file.
GenericErrorCodeTypes	A subset of error codes returned as part of a cell storage service response message. **GenericErrorCodeTypes** is an enumeration of all the generic error code types.
GUID	A **GUID** value.
LockAndCoauthRelatedErrorCodeTypes	A subset of error codes returned as part of a cell storage service response message. **LockAndCoauthRelatedErrorCodeTypes** is an enumeration of error codes specific to a coauthoring subrequest, schema lock subrequest, or exclusive lock subrequest.
LockTypes	The type of the **LockType** attribute, which is part of a **SubResponse** and **SubRequest** the element. **LockTypes** is an enumeration of all file lock types.
MinorVersionNumberType	The type of the **MinorVersion** attribute of the **RequestVersion** element and **ResponseVersion** element. The **RequestVersion** element and **ResponseVersion** element are used in cell storage service request and cell storage service response messages, respectively.
SubRequestAttributeType	The type of the **Type** attribute of the **SubRequest** element. **SubRequestAttributeType** is an enumeration of all the types of cell storage service subrequest.

VersionNumberType	The type of the **Version** attribute of the **RequestVersion** element and **ResponseVersion** element. The **RequestVersion** element and **ResponseVersion** element are used in cell storage service request and cell storage service response messages, respectively.

2.2.5.1 CoauthStatusType

The **CoauthStatusType** simple type is used to represent the coauthoring status of a targeted URL for the file as part of processing a coauthoring subrequest or exclusive lock subrequest of type "Convert to schema lock with coauthoring transition tracked". The exclusive lock subrequest of type "Convert to schema lock with coauthoring transition tracked" is defined in section 2.3.3.4.

```
<xs:simpleType name="CoauthStatusType">
    <xs:restriction base="xs:string">
      <!--None-->
      <xs:enumeration value="None"/>

      <!--Alone-->
      <xs:enumeration value="Alone"/>

      <!--Coauthoring-->
      <xs:enumeration value="Coauthoring"/>
    </xs:restriction>
</xs:simpleType>
```

The value of **CoauthStatusType** MUST be one of the values in the following table.

Value	Meaning
"None"	None.
"Alone"	A string value of "Alone", indicating a coauthoring status of alone. The alone status specifies that there is only one user in the coauthoring session who is editing the file.
"Coauthoring"	A string value of "Coauthoring", indicating a coauthoring status of coauthoring. The coauthoring status specifies that the targeted URL for the file has more than one user in the coauthoring session and that the file is being edited by more than one user.

2.2.5.2 DependencyCheckRelatedErrorCodeTypes

The **DependencyCheckRelatedErrorCodeTypes** simple type is used to represent error codes that occur during dependency checks done during the processing of a cell storage service request.

```
<xs:simpleType name="DependencyCheckRelatedErrorCodeTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="DependentRequestNotExecuted"/>
      <xs:enumeration value="DependentOnlyOnSuccessRequestFailed"/>
      <xs:enumeration value="DependentOnlyOnFailRequestSucceeded"/>
      <xs:enumeration value="DependentOnlyOnNotSupportedRequestGetSupported"/>
      <xs:enumeration value="InvalidRequestDependencyType"/>
    </xs:restriction>
```

```
</xs:simpleType>
```

The value of **DependencyCheckRelatedErrorCodeTypes** MUST be one of the values in the following table.

Value	Meaning
"DependentRequestNotExecuted"	Indicates an error when the subrequest on which this specific subrequest is dependent has not been executed and the **DependencyType** attribute in this subrequest is set to "OnExecute".
"DependentOnlyOnSuccessRequestFailed"	Indicates an error when the subrequest on which this specific subrequest is dependent has failed and the **DependencyType** attribute in this subrequest is set to "OnSuccess" or "OnSuccessOrNotSupported".
"DependentOnlyOnFailRequestSucceeded"	Indicates an error when the subrequest on which this specific subrequest is dependent has succeeded and the **DependencyType** attribute in this subrequest is set to "OnFail".
"DependentOnlyOnNotSupportedRequestGetSupported"	Indicates an error when the subrequest on which this specific subrequest is dependent is supported and the **DependencyType** attribute in this subrequest is set to "OnNotSupported" or "OnSuccessOrOnNotSupported".
"InvalidRequestDependencyType"	Indicates an error when a subrequest dependency type that is not valid is specified.

DependencyTypes is defined in section 2.2.5.3.

2.2.5.3 DependencyTypes

The **DependencyTypes** simple type is used to represent the type of dependency that a cell storage service subrequest has on another cell storage service subrequest. The other cell storage service subrequest is identified by the **DependsOn** attribute of the **SubRequest** element. **DependsOn** is defined in section 2.2.4.3. Depending on the dependency type, a cell storage service subrequest is either processed or not.

```
<xs:simpleType name="DependencyTypes">
   <xs:restriction base="xs:string">
     <xs:enumeration value="OnExecute"/>
     <xs:enumeration value="OnSuccess"/>
     <xs:enumeration value="OnFail"/>
     <xs:enumeration value="OnNotSupported"/>
     <xs:enumeration value="OnSuccessOrNotSupported"/>
   </xs:restriction>
</xs:simpleType>
```

The value of **DependencyTypes** MUST be one of the values in the following table.

Value	Meaning
"OnExecute"	Indicates that the subrequest MUST be processed only on the execution of the other subrequest.
"OnSuccess"	Indicates that the subrequest MUST be processed only on the successful execution of the other subrequest.
"OnFail"	Indicates that the subrequest MUST be processed only on the failed execution of the other subrequest.

"OnNotSupported"	Indicates that the subrequest MUST be processed only if the other subrequest is not supported.
"OnSuccessOrNotSupported"	Indicates that the subrequest MUST be processed only when one of the following conditions is true: ▪ On the successful execution of the other subrequest. ▪ If the other subrequest is not supported.

2.2.5.4 ErrorCodeTypes

The **ErrorCodeTypes** simple type is used to represent the error codes in a subresponse. **ErrorCodeTypes** is the type definition of the **ErrorCode** attribute, which is part of a cell storage service subresponse operation. **ErrorCodeTypes** is a union of simple types, namely **GenericErrorCodeTypes**, **CellRequestErrorCodeTypes**, **DependencyCheckRelatedErrorCodeTypes**, **LockAndCoauthRelatedErrorCodeTypes** and **NewEditorsTableCategoryErrorCodeTypes**.

```
<xs:simpleType name="ErrorCodeTypes">
 <xs:union memberTypes="tns:GenericErrorCodeTypes tns:CellRequestErrorCodeTypes
tns:DependencyCheckRelatedErrorCodeTypes tns:LockAndCoauthRelatedErrorCodeTypes
tns:NewEditorsTableCategoryErrorCodeTypes"/>
</xs:simpleType>
```

GenericErrorCodeTypes is defined in section 2.2.5.6. **CellRequestErrorCodeTypes** is defined in section 2.3.2.1. **DependencyCheckRelatedErrorCodeTypes** is defined in section 2.2.5.2. **LockAndCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.7. **NewEditorsTableCategoryErrorCodeTypes** is defined in section 2.2.5.14.

2.2.5.5 ExclusiveLockReturnReasonTypes

The **ExclusiveLockReturnReasonTypes** simple type is used to represent string values that indicate the reason why an exclusive lock is granted on a file in a cell storage service response message.

```
<xs:simpleType name="ExclusiveLockReturnReasonTypes">
   <xs:restriction base="xs:string">
      <xs:enumeration value="CoauthoringDisabled" />
      <xs:enumeration value="CheckedOutByCurrentUser" />
      <xs:enumeration value="CurrentUserHasExclusiveLock" />
   </xs:restriction>
</xs:simpleType>
```

The value of **ExclusiveLockReturnReasonTypes** MUST be one of the values in the following table.

Value	Meaning
"CoauthoringDisabled"	The string value "CoauthoringDisabled", indicating that an exclusive lock is granted on a file because coauthoring is disabled.
"CheckedOutByCurrentUser"	The string value "CheckedOutByCurrentUser", indicating that an exclusive lock is granted on the file because the file is checked out by the current user who sent the cell storage service request message.
"CurrentUserHasExclusiveLock"	The string value "CurrentUserHasExclusiveLock",

	indicating that an exclusive lock is granted on the file because the current user who sent the cell storage service request message already has an existing exclusive lock on the file<5>.

2.2.5.6 GenericErrorCodeTypes

The **GenericErrorCodeTypes** simple type is used to represent generic error code types that occur during cell storage service subrequest processing.

```
<xs:simpleType name="GenericErrorCodeTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Success"/>
      <xs:enumeration value="IncompatibleVersion"/>
      <xs:enumeration value="InvalidUrl"/>
      <xs:enumeration value="FileNotExistsOrCannotBeCreated"/>
      <xs:enumeration value="FileUnauthorizedAccess"/>
      <xs:enumeration value="InvalidSubRequest"/>
      <xs:enumeration value="SubRequestFail"/>
      <xs:enumeration value="BlockedFileType"/>
      <xs:enumeration value="DocumentCheckoutRequired"/>
      <xs:enumeration value="InvalidArgument"/>
      <xs:enumeration value="RequestNotSupported"/>
      <xs:enumeration value="InvalidWebUrl"/>
      <xs:enumeration value="WebServiceTurnedOff"/>
      <xs:enumeration value="ColdStoreConcurrencyViolation"/>
      <xs:enumeration value="HighLevelExceptionThrown"/>
      <xs:enumeration value="Unknown"/>
    </xs:restriction>
</xs:simpleType>
```

The value of **GenericErrorCodeTypes** MUST be one of the values in the following table.

Value	Meaning
"Success"	Indicates that the cell storage service subrequest succeeded for the given URL for the file.
"IncompatibleVersion"	Indicates an error when any an incompatible version number is specified as part of the **RequestVersion** element of the cell storage service.
"InvalidUrl"	Indicates that the associated protocol server site URL is empty.
"FileNotExistsOrCannotBeCreated"	Indicates an error when either the targeted URL for the file specified as part of the **Request** element does not exist or file creation failed on the protocol server.
"FileUnauthorizedAccess"	Indicates an error when the targeted URL for the file specified as part of the **Request** element does not have correct authorization.
"InvalidSubRequest"	Indicates an error when one or more **SubRequest** elements for a targeted URL for the file were unable to be parsed.
"SubRequestFail"	Indicates an unknown error when processing any **SubRequest** element for a targeted URL for the file.
"BlockedFileType"	Indicates an error when the targeted URL to the file's file type is blocked on the protocol server.
"DocumentCheckoutRequired"	Indicates an error when the targeted URL for the file is not yet checked out by the current client before

	sending a lock request on the file. The client sends a **CheckoutFile** web service request, as specified in [MS-LISTSWS], before the protocol client gets a lock on the file. If the document is not checked out by the current client, the protocol server MUST return an error code value set to "DocumentCheckoutRequired" in the cell storage service response message.
"InvalidArgument"	Indicates an error when any of the cell storage service subrequests for the targeted URL for the file contains input parameters that are not valid.
"RequestNotSupported"	Indicates an error when the targeted cell storage service subrequest is a valid subrequest, but the server does not support that subrequest.
"InvalidWebUrl"	Indicates an error when the associated protocol server site URL is not found.
"WebServiceTurnedOff"	Indicates an error when the web service is turned off during the processing of the cell storage service request.
"ColdStoreConcurrencyViolation"	Indicates an error when the file that is correctly stored on the server is modified by another user before the current user finished writing to the underlying store. The "ColdStoreConcurrencyViolation" error code value MUST only be sent as part of processing one of the following types of cell storage service request messages: ▪ Cell ▪ Coauth ▪ SchemaLock ▪ ExclusiveLock
"HighLevelExceptionThrown"<6>	Indicates any undefined error that occurs during the processing of the cell storage service request.
"Unknown"	Indicates any undefined error that occurs during the processing of the cell storage service request.

2.2.5.7 GUID

The **guid** type specifies a representation of a GUID value.

```
<xs:simpleType name="guid">
   <xs:restriction base="xs:string">
     <xs:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-
F]{12}" />
   </xs:restriction>
</xs:simpleType>
```

2.2.5.8 LockAndCoauthRelatedErrorCodeTypes

The **LockAndCoauthRelatedErrorCodeTypes** simple type is used to represent error codes that occur during the processing of a coauthoring, schema lock, or exclusive lock subrequest.

```
<xs:simpleType name="LockAndCoauthRelatedErrorCodeTypes">
   <xs:restriction base="xs:string">
```

```
        <xs:enumeration value="LockRequestFail"/>
        <xs:enumeration value="FileAlreadyLockedOnServer"/>
        <xs:enumeration value="FileNotLockedOnServer"/>
        <xs:enumeration value="FileNotLockedOnServerAsCoauthDisabled"/>
        <xs:enumeration value="LockNotConvertedAsCoauthDisabled"/>
        <xs:enumeration value="FileAlreadyCheckedOutOnServer"/>
        <xs:enumeration value="ConvertToSchemaFailedFileCheckedOutByCurrentUser"/>
        <xs:enumeration value="CoauthRefblobConcurrencyViolation"/>
        <xs:enumeration value="MultipleClientsInCoauthSession"/>
        <xs:enumeration value="InvalidCoauthSession"/>
        <xs:enumeration value="NumberOfCoauthorsReachedMax"/>
        <xs:enumeration value="ExitCoauthSessionAsConvertToExclusiveFailed"/>
    </xs:restriction>
  </xs:simpleType>
```

The value of **LockAndCoauthRelatedErrorCodeTypes** MUST be one of the values in the following table.

Value	Meaning
"LockRequestFail"	Indicates an undefined error that occurs during the processing of lock operations requested as part of a cell storage service subrequest.
"FileAlreadyLockedOnServer"	Indicates an error when there is an already existing exclusive lock on the targeted URL for the file or a schema lock on the file with a different schema lock identifier. When the "FileAlreadyLockedOnServer" error code is returned as the error code value in the **SubResponse** element, the protocol server returns the identity of the users who are currently holding the lock on the file in the **ErrorMessage** attribute. The **ErrorMessage** and **ErrorCode** attributes are defined in section 2.2.4.6.
"FileNotLockedOnServer"	Indicates an error when no exclusive lock exists on a file and a release of the lock or a conversion of the lock is requested as part of a cell storage service request.
"FileNotLockedOnServerAsCoauthDisabled"	Indicates an error when no shared lock exists on a file because coauthoring of the file is disabled on the server.
"LockNotConvertedAsCoauthDisabled"	Indicates an error when a protocol server fails to process a lock conversion request sent as part of a cell storage service request because coauthoring of the file is disabled on the server.
"FileAlreadyCheckedOutOnServer"	Indicates an error when the file is checked out by another client, which is preventing the file from being locked by the current client. When the "FileAlreadyCheckedOutOnServer" error code is returned as the error code value in the **SubResponse** element, the protocol server returns the identity of the user who has currently checked out the file in the error message attribute. The **ErrorMessage** and **ErrorCode** attributes are defined in section 2.2.4.6.
"ConvertToSchemaFailedFileCheckedOutByCurrentUser"	Indicates an error when converting to a shared lock fails because the file is checked out by the current client.
"CoauthRefblobConcurrencyViolation"	Indicates an error when a save of the **File coauthoring tracker**, which is maintained by the protocol server, fails after some other client edited

	the **File coauthoring tracker** before the save is done by the current client. The **File coauthoring tracker** is defined in section 3.1.1.
"MultipleClientsInCoauthSession"	Indicates an error when all of the following conditions are true: ▪ A coauthoring subrequest of type "Convert to exclusive lock" or schema lock subrequest of type "Convert to exclusive lock" is requested on a file. ▪ There is more than one client in the current coauthoring session for that file. ▪ The **ReleaseLockOnConversionToExclusiveFailure** attribute specified as part of the subrequest is set to **false**.
"InvalidCoauthSession"	Indicates an error when one of the following conditions is **true** when a coauthoring subrequest or schema lock subrequest is sent: ▪ No coauthoring session exists for the file. ▪ The current client does not exist in the coauthoring session for the file. ▪ The current client exists in the coauthoring session, but protocol server is unable to remove it from the coauthoring session for the file. A coauthoring session indicates a shared lock on the coauthorable file that is shared by one or more clients.
"NumberOfCoauthorsReachedMax"	Indicates an error when the number of users that coauthor a file has reached the threshold limit. The threshold limit specifies the maximum number of users allowed to coauthor a file at any instant in time. The threshold limit MUST be set to an integer value with a minimum allowed value of 2 and maximum allowed value of 99.
ExitCoauthSessionAsConvertToExclusiveFailed	Indicates an error when a coauthoring subrequest or schema lock subrequest of type "Convert to exclusive lock" is sent by the client with the **ReleaseLockOnConversionToExclusiveFailure** attribute set to **true**, and there is more than one client editing the file.

2.2.5.9 LockTypes

The **LockTypes** simple type is used to represent the type of file lock. **LockTypes** specifies the type of lock requested or granted in a cell storage service request or cell storage service response message.

```
<xs:simpleType name="LockTypes">
   <xs:restriction base="xs:string">
      <xs:enumeration value="None" />
      <xs:enumeration value="SchemaLock" />
      <xs:enumeration value="ExclusiveLock" />
   </xs:restriction>
</xs:simpleType>
```

The value of **LockTypes** MUST be one of the values in the following table.

Value	Meaning
"None"	The string value "None", indicating no type of file lock on the file. This value is only for server internal use, it will not appear in the response.
"SchemaLock"	The string value "SchemaLock", indicating a shared lock on the file. In a cell storage service request message, a shared lock indicates a request for sharing the lock on the file, which allows for coauthoring the file. In a cell storage service response message, a shared lock indicates that the current client is granted a shared lock on the file, which allows for coauthoring the file along with other clients.
"ExclusiveLock"	The string value "ExclusiveLock", indicating an exclusive lock on the file. In a cell storage service request message, an exclusive lock indicates a request for exclusive access to the file. In a cell storage service response message, an exclusive lock indicates that an exclusive lock is granted to the current client for that specific file. In a cell storage service response message, an exclusive lock also indicates that all other clients requesting an exclusive lock on that file MUST be allowed to open this file only in read-only mode.

2.2.5.10 MinorVersionNumberType

The **MinorVersionNumberType** simple type is used to represent the minor version number as unsigned short values.

```
<xs:simpleType name="MinorVersionNumberType">
    <xs:restriction base="xs:unsignedShort">
       <xs:minInclusive value="0"/>
       <xs:maxInclusive value="2"/>
    </xs:restriction>
</xs:simpleType>
```

The value of **MinorVersionNumberType** MUST be 0 or 2, as described in the following table.

Value	Meaning
0	In requests, indicates that the protocol client will manage the editors table through **PutChanges** requests as specified in [MS-FSSHTTPB]. In responses, indicates that the protocol server is not capable of managing the editors table and expects the protocol client to do so through **PutChanges** requests.
2	In requests, indicates that the protocol client is capable of managing the editors table. In responses, indicates that the protocol server is capable of managing the editors table.

2.2.5.11 SubRequestAttributeType

The **SubRequestAttributeType** simple type is used to represent the type of cell storage service subrequest. Depending on the type of subrequest, the subrequest is processed as one of the following types of subrequest operations:

- Cell
- Coauthoring
- Schema lock
- Exclusive lock
- Who Am I
- Server time
- Editors Table
- Get Doc Meta Info
- Get Versions

```
<xs:simpleType name="SubRequestAttributeType">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Cell" />
      <xs:enumeration value="Coauth" />
      <xs:enumeration value="SchemaLock" />
      <xs:enumeration value="WhoAmI" />
      <xs:enumeration value="ServerTime" />
      <xs:enumeration value="ExclusiveLock" />
      <xs:enumeration value="EditorsTable" />
      <xs:enumeration value="GetDocMetaInfo" />
      <xs:enumeration value="GetVersions" />
    </xs:restriction>
</xs:simpleType>
```

The value of **SubRequestAttributeType** MUST be one of the values in the following table.

Value	Meaning
"Cell"	The string value "Cell", indicating the cell storage service subrequest is to be processed as a cell subrequest operation.
"Coauth"	The string value "Coauth", indicating the cell storage service subrequest is to be processed as a coauthoring subrequest operation.
"SchemaLock"	The string value "The schemaLock", indicating the cell storage service subrequest is to be processed as a schema lock subrequest operation.
" WhoAmI"	The string value "WhoAmI", indicating the cell storage service subrequest is to be processed as a Who Am I subrequest operation.
"ServerTime"	The string value "ServerTime", indicating the cell storage service subrequest is to be processed as a server time subrequest operation.
"ExclusiveLock"	The string value "ExclusiveLock", indicating the cell storage service subrequest is to be processed as an exclusive lock subrequest operation.

"EditorsTable"	The string value "EditorsTable", indicating the cell storage service subrequest is to be processed as an editors table subrequest operation.
"GetDocMetaInfo"	The string value "GetDocMetaInfo", indicating the cell storage service subrequest is to be processed as a Get Doc Meta Info subrequest operation.
"GetVersions"	The string value "GetVersions", indicating the cell storage service subrequest is to be processed as a Get Versions subrequest operation.

2.2.5.12 TRUEFALSE

This type is used to specify a Boolean value.

```
<xs:simpleType name="TRUEFALSE">
  <xs:restriction base="xs:string">
    <xs:pattern value="[Tt][Rr][Uu][Ee]|[Ff][Aa][Ll][Ss][Ee]"/>
  </xs:restriction>
</xs:simpleType>
```

2.2.5.13 VersionNumberType

The **VersionNumberType** simple type is used to represent a version number as an unsigned short value.

```
<xs:simpleType name="VersionNumberType">
    <xs:restriction base="xs:unsignedShort">
      <xs:minInclusive value="2"/>
      <xs:maxInclusive value="2"/>
    </xs:restriction>
</xs:simpleType>
```

The value of **VersionNumberType** MUST be the value that is listed in the following table.

Value	Meaning
2	A version number of 2.

2.2.5.14 NewEditorsTableCategoryErrorCodeTypes

The **NewEditorsTableCategoryErrorCodeTypes** simple type is used to represent error codes that during the processing of an EditorsTable subrequest.<7>

```
<xs:simpleType name="NewEditorsTableCategoryErrorCodeTypes">
    <xs:restriction base="xs:string">
       <xs:enumeration value="EditorMetadataQuotaReached"/>
       <xs:enumeration value="EditorMetadataStringExceedsLengthLimit"/>
       <xs:enumeration value="EditorClientIdNotFound"/>
    </xs:restriction>
</xs:simpleType>
```

The value of **NewEditorsTableCategoryErrorCodeTypes** MUST be one of the values in the following table.

Value	Meaning
"EditorMetadataQuotaReached"	Indicates an error when the client has already exceeded its quota for number of key/value pairs.
"EditorMetadataStringExceedsLengthLimit"	Indicates an error when the key and value exceeds the server's length limit.
"EditorClientIdNotFound"	Indicates an error when the specify client does not currently exist in the editors table.

2.2.6 Attributes

This specification does not define any common XML schema attribute definitions.

2.2.7 Groups

This specification does not define any common XML schema group definitions.

2.2.8 Attribute Groups

The following table summarizes the set of common XML schema attribute group definitions defined by this specification. XML schema attribute group definitions that are specific to a particular operation are described with the operation.

Attribute Groups	Description
SubRequestDataOptionalAttributes	Contains XML schema attributes used in all **SubRequestData** elements. It is a union of subrequest data attributes used for all types of subrequests.
SubResponseDataOptionalAttributes	Contains XML schema attributes used in all **SubResponseData** elements. It is a union of subresponse data attributes used for all types of subresponses.

2.2.8.1 SubRequestDataOptionalAttributes

The **SubRequestDataOptionalAttributes** attribute group contains attributes that are used in **SubRequestData** elements of all types of subrequests. The attributes in **SubRequestDataOptionalAttributes** are used as input parameters for processing the data associated with subrequests. The definition of the **SubRequestDataOptionalAttributes** attribute group is as follows:

```
<xs:attributeGroup name="SubRequestDataOptionalAttributes">
    <xs:attributeGroup ref="tns:CellSubRequestDataOptionalAttributes"/>
    <xs:attributeGroup ref="tns:CoauthSubRequestDataOptionalAttributes"/>
    <xs:attributeGroup ref="tns:SchemaLockSubRequestDataOptionalAttributes"/>
    <xs:attributeGroup ref="tns:ExclusiveLockSubRequestDataOptionalAttributes"/>
    <xs:attributeGroup ref="tns:EditorsTableSubRequestDataOptionalAttributes"/>
    <xs:attribute name="ClientID" type="xs:string" use="optional"/>
    <xs:attribute name="AllowFallbackToExclusive" type="xs:boolean" use="optional" />
    <xs:attribute name="ReleaseLockOnConversionToExclusiveFailure" type="xs:boolean"
use="optional"/>
```

```
    <xs:attribute name="SchemaLockID" type="xs:string" use="optional" />
    <xs:attribute name="Timeout" type="xs:integer" use="optional" />
    <xs:attribute name="ExclusiveLockID" type="xs:string" use="optional"/>
    <xs:attribute name="BinaryDataSize" type="xs:long" use="optional" />
    <xs:attribute name="AsEditor" type="xs:boolean" use="optional" />
    <xs:attribute name="Key" type="xs:string" use="optional" />
    <xs:attribute name="Value" type="xs:binary" use="optional" />
</xs:attributeGroup>
```

CellSubRequestDataOptionalAttributes: An attribute group that specifies attributes that MUST be used only for **SubRequestData** elements whose parent **SubRequest** element's **Type** attribute is set to "Cell". This attribute group is defined in section 2.3.3.1.

CoauthSubRequestDataOptionalAttributes: An attribute group that specifies attributes that MUST be used only for **SubRequestData** elements whose parent **SubRequest** element's **Type** attribute is set to "Coauth". This attribute group is defined in section 2.3.3.3.

SchemaLockSubRequestDataOptionalAttributes: An attribute group that specifies attributes that MUST be used only for **SubRequestData** elements whose parent **SubRequest** element's **Type** attribute is set to "SchemaLock". This attribute group is defined in section 2.3.3.5.

ExclusiveLockSubRequestDataOptionalAttributes: An attribute group that specifies attributes that MUST be used only for **SubRequestData** elements whose parent **SubRequest** element's **Type** attribute is set to "ExclusiveLock". This attribute group is defined in section 2.3.3.4.

EditorsTableSubRequestDataOptionalAttributes: An attribute group that specifies attributes that MUST be used only for **SubRequestData** elements whose parent **SubRequest** element's **Type** attribute is set to "EditorsTable". This attribute group is defined in section 2.3.3.7 .

ClientID: A string that serves to uniquely identify each client that has access to a shared lock on a coauthorable file.

AllowFallbackToExclusive: A Boolean value that specifies to a protocol server whether a coauthoring subrequest of type "Join coauthoring session" or a schema lock subrequest of type "Get lock" is allowed to fall back to an exclusive lock subrequest when shared locking on the file is not supported. When shared locking on the file is not supported:

- An **AllowFallbackToExclusive** attribute value set to **true** indicates that a coauthoring subrequest of type "Join coauthoring session" or a schema lock subrequest of type "Get lock" is allowed to fall back to an exclusive lock subrequest.
- An **AllowFallbackToExclusive** attribute value set to **false** indicates that a coauthoring subrequest of type "Join coauthoring session" or a schema lock subrequest of type "Get lock" is not allowed to fall back to an exclusive lock subrequest.

ReleaseLockOnConversionToExclusiveFailure: A Boolean value that specifies to the protocol server whether the server is allowed to remove the **ClientID** entry associated with the current client in the **File coauthoring tracker** when all of the following conditions are true:

- Either the type of coauthoring subrequest is "Convert to an exclusive lock" or the type of the schema lock subrequest is "Convert to an Exclusive Lock"
- The conversion to an exclusive lock failed.

When all the preceding conditions are true:

- A **ReleaseLockOnConversionToExclusiveFailure** attribute set to **true** indicates that the protocol server is allowed to remove the **ClientID** entry associated with the current client in the **File coauthoring tracker**.

- A **ReleaseLockOnConversionToExclusiveFailure** attribute set to a value of **false** indicates that the protocol server is not allowed to remove the **ClientID** entry associated with the current client in the **File coauthoring tracker**.

SchemaLockID: A string that is globally unique and known among all protocol clients that share the same protocol version. This schema lock identifier is used by the protocol server to block other clients that have different schema lock identifiers. After a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST allow only other protocol clients that specify the same schema lock identifier to share the file lock. The protocol server ensures that at any instant of time, only clients having the same schema lock identifier can lock the file. After all the protocol clients have released their lock for that file, the protocol server MUST allow a protocol client with a different schema lock identifier to get a shared lock for that file. The string "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use for this attribute.<8>

Timeout: An integer that specifies the time, in seconds, after which the shared lock or exclusive lock for a specific file expires for a specific protocol client. When more than one client is editing the file, the protocol server MUST maintain a separate timeout value for each client.

ExclusiveLockID: A string that serves as a unique identifier for the exclusive lock on the file.

BinaryDataSize: A long value that specifies the number of bytes of data in the **SubRequestData** element of a cell subrequest. It MUST be present in the **SubRequestData** element of a cell subrequest. The **BinaryDataSize** attribute MUST be set to a value ranging from 1 through 9,223,372,036,854,775,807. It is ignored by the server. The **SubRequestData** element is defined in section 2.2.3.9.

If text is specified in the **SubRequestData** element, that text is base64 binary encoded data and indicates if the cell subrequest is for the upload or download of data in a partition. This is passed to the component on the protocol server responsible for implementing the protocols as specified in [MS-FSSHTTPB] section 3.1.4.2 and [MS-FSSHTTPB] section 3.1.4.4. The encoded data is opaque to the protocol.

AsEditor: A Boolean value that specifies to the server whether the client is opening the document as an editor or as a reader. The server MUST NOT allow a user with read-only access to join the editing session as a reader. The **AsEditor** attribute MUST be specified in all of the following types of editors table subrequests:

- Join editing session
- Refresh editing session

The types of editors table subrequests are defined in section 2.3.3.7.

Key: A string that specifies a unique key in an arbitrary key/value pair of the client's choice. The server stores this key/value pair for that particular file for that specific protocol client. These pairs are visible to other clients editing or reading the same document. The **Key** attribute MUST be specified in all of the following types of editors table subrequests:

- Update Editor Metadata
- Remove Editor Metadata

The types of editors table subrequests are defined in section 2.3.3.7.

Value: A binary value that is associated with a key in an arbitrary key/value pair of the client's choice. The server stores this key/value pair for that particular file for that specific protocol client. These pairs are visible to other clients editing or reading the same document. The **Value** attribute MUST be specified in an editors table subrequest type of "Update Editor Metadata".

The types of editors table subrequests are defined in section 2.3.3.7.

2.2.8.2 SubResponseDataOptionalAttributes

The **SubResponseDataOptionalAttributes** attribute group contains attributes that are used in **SubResponseData** elements associated with a **SubResponse** element. The **SubResponse** element is a subresponse for any type of cell storage service subrequest. The attributes in **SubResponseDataOptionalAttributes** provide the data that was requested as part of the subrequest. The definition of the **SubResponseDataOptionalAttributes** attribute group is as follows:

```
<xs:attributeGroup name="SubResponseDataOptionalAttributes">
    <xs:attributeGroup ref="tns:CellSubResponseDataOptionalAttributes"/>
    <xs:attributeGroup ref="tns:WhoAmISubResponseDataOptionalAttributes"/>
    <xs:attribute name="ServerTime" type="xs:positiveInteger" use="optional"/>
    <xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
    <xs:attribute name="CoauthStatus" type="tns:CoauthStatusType" use="optional"/>
    <xs:attribute name="TransitionID" type="tns:guid" use="optional"/>
    <xs:attribute name="ExclusiveLockReturnReason" type="tns:ExclusiveLockReturnReasonTypes"
use="optional" />
</xs:attributeGroup>
```

CellSubResponseDataOptionalAttributes: An attribute group that specifies attributes that MUST be used only for **SubResponseData** elements associated with a subresponse for a cell subrequest. The **CellSubResponseDataOptionalAttributes** attribute group is defined in section 2.3.3.2.

WhoAmISubResponseDataOptionalAttributes: An attribute group that specifies attributes that MUST be used only for **SubResponseData** elements associated with a subresponse for a **WhoAmI** subrequest. The **WhoAmISubResponseDataOptionalAttributes** attribute group is defined in section 2.3.3.6.

ServerTime: A positive integer that specifies the server time, which is expressed as a tick count. A single tick represents 100 nanoseconds, or one ten-millionth of a second. **ServerTime** specifies the number of 100-nanosecond intervals that have elapsed since 00:00:00 on January 1, 1601, which SHOULD<9> be **Coordinated Universal Time (UTC)**. The **ServerTime** attribute MUST be specified in a server time subresponse that is generated in response to a server time subrequest.

LockType: A **LockTypes** that specifies the type of lock granted in a coauthoring subresponse or a schema lock subresponse. **LockTypes** is defined in section 2.2.5.9. If the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success", the **LockType** attribute MUST be specified in a subresponse that is generated in response to one of the following types of cell storage service subrequest operations:

- A coauthoring subrequest of type "Join coauthoring session"
- A coauthoring subrequest of type "Refresh coauthoring session"
- A schema lock subrequest of type "Get lock"
- A schema lock subrequest of type "Refresh lock"

The types of coauthoring subrequests are defined in section 2.3.3.3. The types of schema lock subrequests are defined in section 2.3.3.5.

CoauthStatus: A **CoauthStatusType** that specifies the coauthoring status in either a coauthoring subresponse or an exclusive lock subresponse. The **CoauthStatusType** is defined in section 2.2.5.1. If the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success", the **CoauthStatus** attribute MUST be specified in a subresponse that is generated in response to one of the following types of cell storage service subrequest operations:

- A coauthoring subrequest of type "Join coauthoring session"<10>

- A coauthoring subrequest of type "Refresh coauthoring session"
- A coauthoring subrequest of type "Get coauthoring status"
- An exclusive lock subrequest of type "Convert to schema lock with coauthoring transition tracked"

The types of coauthoring subrequests are defined in section 2.3.3.3. The types of exclusive lock subrequests are defined in section 2.3.3.4.

TransitionID: A **guid** that specifies the unique file identifier stored for that file on the protocol server. This transition identifier serves as an input parameter to the **IsOnlyClient**, as specified in [MS-SHDACCWS]. The **guid** type is defined in section 2.2.5.7.

ExclusiveLockReturnReason: An **ExclusiveLockReturnReasonTypes** that specifies the reason why an exclusive lock is granted in either a coauthoring subresponse or a schema lock subresponse. **ExclusiveLockReturnReasonTypes** is defined in section 2.2.5.5. The **ExclusiveLockReturnReason** attribute MUST be specified in a subresponse that is generated in response to one of the following types of cell storage service subrequest operations when the **LockType** attribute in the subresponse is set to "ExclusiveLock":

- A coauthoring subrequest of type "Join coauthoring session"
- A schema lock subrequest of type "Get lock"

The types of coauthoring subrequests are defined in section 2.3.3.3. The types of schema lock subrequests are defined in section 2.3.3.5.

2.2.9 Common Data Structures

This specification does not define any common XML schema data structures.

2.3 Subsidiary Message Syntax

This section contains definitions that are used by this protocol. The syntax of the definitions uses XML schema, specified in [XMLSCHEMA1] and [XMLSCHEMA2], and WSDL, as specified in [WSDL].

2.3.1 Complex Types

The following table summarizes the set of other XML schema complex type definitions defined by this specification.

Complex type	Description
CellSubRequestDataType	The type definition for cell subrequest data.
CellSubRequestType	The type definition for a cell subrequest when the **Type** attribute is set to "Cell".
CellSubResponseDataType	The type definition for cell subresponse data.
CellSubResponseType	The type definition for a cell subresponse.
CoauthSubRequestDataType	The type definition for coauthoring subrequest data.
CoauthSubRequestType	The type definition for a coauthoring subrequest when the **Type** attribute is set to "Coauth".
CoauthSubResponseDataType	The type definition for coauthoring subresponse data.
CoauthSubResponseType	The type definition for a coauthoring subresponse.
ExclusiveLockSubRequestDataType	The type definition for exclusive lock subrequest data.
ExclusiveLockSubRequestType	The type definition for an exclusive lock subrequest when the **Type** attribute is set to "ExclusiveLock".
ExclusiveLockSubResponseDataType	The type definition for exclusive lock subresponse data.

ExclusiveLockSubResponseType	The type definition for an ExclusiveLock subresponse.
SchemaLockSubRequestDataType	The type definition for schema lock subrequest data.
SchemaLockSubRequestType	The type definition for a schema lock subrequest when the **Type** attribute is set to "SchemaLock".
SchemaLockSubResponseDataType	The type definition for schema lock subresponse data.
SchemaLockSubResponseType	The type definition for a schema lock subresponse.
ServerTimeSubRequestType	The type definition for a server time subrequest when the **Type** attribute is set to "ServerTime".
ServerTimeSubResponseDataType	The type definition for server time subresponse data.
ServerTimeSubResponseType	The type definition for a server time subresponse.
WhoAmISubRequestType	The type definition for a Who Am I subrequest when the **Type** attribute is set to "WhoAmI".
WhoAmISubResponseDataType	The type definition for Who Am I subresponse data.
WhoAmISubResponseType	The type definition for a Who Am I subresponse.
EditorsTableSubRequestDataType	The type definition for editors table subrequest data.
EditorsTableSubRequestType	The type definition for an editors table subrequest when the **Type** attribute is set to "EditorsTable".
EditorsTableSubResponseType	The type definition for an editors table subresponse.
GetDocMetaInfoSubRequestType	The type definition for a Get Doc Meta Info subrequest when the **Type** attribute is set to "GetDocMetaInfo"
GetDocMetaInfoSubResponseDataType	The type definition for Get Doc Meta Info subresponse data.
GetDocMetaInfoSubResponseType	The type definition for a Get Doc Meta Info subresponse.
GetVersionsSubRequestType	The type definition for a Get Versions subrequest when the attribute is set to "GetVersions"
GetVersionsSubResponseType	The type definition for a Get Versions subresponse.

2.3.1.1 CellSubRequestDataType

The **CellSubRequestDataType** complex type contains information about data or input parameters used in processing a cell subrequest.

```
<xs:complexType name="CellSubRequestDataType" mixed="true">
    <xs:all>
      <xs:element ref="i:Include" minOccurs="0" maxOccurs="1" />
    </xs:all>
    <xs:attributeGroup ref="tns:CellSubRequestDataOptionalAttributes" />
    <xs:attribute name="SchemaLockID" type="xs:string" use="optional" />
    <xs:attribute name="ExclusiveLockID" type="xs:string" use="optional"/>
    <xs:attribute name="Timeout" type="xs:integer" use="optional" />
    <xs:attribute name="BinaryDataSize" type="xs:long" use="required" />
</xs:complexType>
```

Include: A complex type, as specified in [XOP10] section 2.1, that is used for encapsulating and sending large amounts of binary data. The referenced **Include** element is specified in section 2.2.3.1. The referenced **Include** element MUST be sent as part of the **SubRequestData** element in a cell storage service request message only if the cell subrequest is for the upload of a file's binary or metadata contents.

CellSubRequestDataOptionalAttributes: An attribute group that specifies the set of attributes that is provided for a **SubRequestData** element whose parent **SubRequest** element's **Type** attribute is set to "Cell". **CellSubRequestDataOptionalAttributes** is defined in section 2.3.3.1.

SchemaLockID: A string that is globally unique and known among all protocol clients that share the same protocol version. This schema lock identifier is used by the protocol server to block other clients with a different schema lock identifier, if the ByPassLockID is not set or not same with this schema lock identified, the SchemaLockID will be ignored by the server. After a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST allow only other protocol clients that specify the same schema lock identifier to share the file lock. The protocol server ensures that at any instant in time, only clients having the same schema lock identifier can lock the file. After all the protocol clients have released their lock for that file, the protocol server MUST allow a protocol client with a different schema lock identifier to get a shared lock for that file. The **SchemaLockID** attribute MUST be specified in a cell subrequest if both of the following conditions are **true**:

- The **Coalesce** attribute is set to **true**, and the client holds a shared lock on the file. The **Coalesce** attribute is defined in section 2.3.3.1.
- The cell subrequest is for uploading binary file contents in a partition.

The string "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use for this attribute.<11>

ExclusiveLockID: A string that serves as a unique identifier for the type of exclusive lock on the file when a cell subrequest with the **Coalesce** attribute set to true is requested. The **Coalesce** attribute is defined in section 2.3.3.1. **ExclusiveLockID** is sent only if the protocol server supports file locking and the cell subrequest is for the first time upload of the file's binary contents.

Timeout: An integer that specifies the time, in seconds, after which the exclusive lock on the file will expire. The **Timeout** attribute MUST be set to a value ranging from 60 through 120,000. The **Timeout** attribute MUST be specified if the **ExclusiveLockID** attribute is specified.

BinaryDataSize: A long value that specifies the number of bytes of data in the **SubRequestData** element of a cell subrequest. It MUST be present in the **SubRequestData** element of a cell subrequest. The **BinaryDataSize** attribute MUST be set to a value ranging from 1 through 9,223,372,036,854,775,807. It is ignored by the server. The **SubRequestData** element is defined in section 2.2.3.9.

If text is specified in the **SubRequestData** element, that text is base64 binary encoded data and indicates if the cell subrequest is for the upload or download of data in a partition. This is passed to the component on the protocol server responsible for implementing the protocols as specified in [MS-FSSHTTPB] section 3.1.4.2 and [MS-FSSHTTPB] section 3.1.4.3. The encoded data is opaque to the protocol.

2.3.1.2 CellSubRequestType

The **CellSubRequestType** complex type contains information about a cell subrequest. The **SubRequestType** definition from which **CellSubRequestType** is extended is defined in section 2.2.4.3.

```
<xs:complexType name="CellSubRequestType">
   <xs:complexContent>
    <xs:extension base="tns:SubRequestType">
       <xs:sequence minOccurs="1" maxOccurs="1">
         <xs:element name="SubRequestData" type="tns:CellSubRequestDataType" />
       </xs:sequence>
     <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required" fixed="Cell"
/>
    </xs:extension>
```

```
    </xs:complexContent>
  </xs:complexType>
```

SubRequestData: A **CellSubRequestDataType** that specifies the data or input parameters needed in processing the cell subrequest. If no **SubRequestData** element is specified in the cell subrequest, the protocol server MUST process this as a no-operation instruction. **CellSubRequestDataType** is defined in section 2.3.1.1.

Type: A **SubRequestAttributeType** that specifies the type of subrequest. The **Type** attribute MUST be set to "Cell" for a cell subrequest. **SubRequestAttributeType** is defined in section 2.2.5.11.

2.3.1.3 CellSubResponseDataType

The **CellSubResponseDataType** complex type contains information requested as part of the corresponding cell subrequest.

```
<xs:complexType name="CellSubResponseDataType" mixed="true">
   <xs:all>
     <xs:element ref="i:Include" minOccurs="0" maxOccurs="1" />
   </xs:all>
   <xs:attributeGroup ref="tns:CellSubResponseDataOptionalAttributes" />
      <xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
   </xs:complexType>
```

Include: A complex type, as specified in [XOP10] section 2.1, that is used for encapsulating and sending large amounts of binary data. The referenced **Include** element is specified in section 2.2.3.1. As part of processing the cell subrequest, the referenced **Include** element MUST be sent as part of the **SubResponseData** element in a cell storage service response message only if the cell subrequest is for the download of a file's binary or metadata contents and only when these contents are non-empty.

CellSubResponseDataOptionalAttributes: An attribute group that specifies the set of attributes that is provided for a **SubResponseData** element whose parent **SubResponse** element's mapping **SubRequest** element is a cell subrequest. **CellSubResponseDataOptionalAttributes** is defined in section 2.3.3.2.

LockType: A **LockTypes** that specifies the type of lock granted in a cell subresponse. **LockTypes** is defined in section 2.2.5.9. The **LockType** attribute MUST be set to "ExclusiveLock" in the cell subresponse if the **ExclusiveLockID** attribute is sent in the cell subrequest and the protocol server is successfully able to take an exclusive lock. The condition under which the **ExclusiveLockID** attribute is sent in the cell subrequest is specified in section 2.3.1.1.

2.3.1.4 CellSubResponseType

The **CellSubResponseType** complex type contains information about the success or failure in processing the cell subrequest. In the case of success, it contains information requested as part of the cell subrequest. **SubResponseType** definition from which **CellSubResponseType** is extended is defined in section 2.2.4.6.

```
 <xs:complexType name="CellSubResponseType">
   <xs:complexContent>
     <xs:extension base="tns:SubResponseType">
       <xs:sequence>
          <xs:element name="SubResponseData" type="tns:CellSubResponseDataType" minOccurs="0"
maxOccurs="1" />
          <xs:element name="SubResponseStreamInvalid" minOccurs="0" maxOccurs="1" />
       </xs:sequence>
```

```
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

SubResponseData: A **CellSubResponseDataType** that specifies the file contents or specific file metadata information provided by the protocol server that was requested as part of the cell subrequest. **CellSubResponseDataType** is defined in section 2.3.1.3.

SubResponseStreamInvalid: An empty element that indicates the binary data in the **SubResponseData** is not valid because of a server race condition. The protocol client can retry the request if it sees this error indication element.

2.3.1.5 CoauthSubRequestDataType

The **CoauthSubRequestDataType** complex type contains information about data or input parameters used in processing a coauthoring subrequest. The **SchemaLockID** attribute and the **CoauthRequestType** attribute specified in the **CoauthSubRequestDataOptionalAttributes** attribute group MUST both be specified for a coauthoring subrequest. The **SchemaLockID** attribute and the **CoauthRequestType** attribute is specified as part of the **SubRequestData** element associated with a coauthoring **SubRequest** element. **CoauthSubRequestDataOptionalAttributes** is defined in section 2.3.3.3. If the specified attributes are not provided, an "InvalidArgument"<12> error code MUST be returned as part of the **SubResponseData** element associated with the coauthoring subresponse.

```
<xs:complexType name="CoauthSubRequestDataType">
    <xs:attributeGroup ref="tns:CoauthSubRequestDataOptionalAttributes" />
    <xs:attribute name="ClientID" type="xs:string" use="required"/>
    <xs:attribute name="AllowFallbackToExclusive" type="xs:boolean" use="optional" />
    <xs:attribute name="ReleaseLockOnConversionToExclusiveFailure" type="xs:boolean"
use="optional"/>
    <xs:attribute name="SchemaLockID" type="xs:string" use="required" />
    <xs:attribute name="Timeout" type="xs:integer" use="optional" />
    <xs:attribute name="ExclusiveLockID" type="xs:string" use="optional"/>
</xs:complexType>
```

CoauthSubRequestDataOptionalAttributes: An attribute group that specifies the set of attributes that is provided for a **SubRequestData** element whose parent **SubRequest** element's **Type** attribute is set to "Coauth". The attributes **CoauthSubRequestDataOptionalAttributes** is defined in section 2.3.3.3.

ClientID: A string that serves to uniquely identify each client that has access to a shared lock on a coauthorable file. **ClientID** MUST be specified on all types of coauthoring subrequests. The types of coauthoring subrequest are defined in section 2.3.3.3.

AllowFallbackToExclusive: A Boolean value that specifies to a protocol server whether a coauthoring subrequest is allowed to fall back to an exclusive lock subrequest provided shared locking on the file is not supported. When shared locking on the file is not supported:

- An **AllowFallbackToExclusive** attribute value set to **true** indicates that a coauthoring subrequest is allowed to fall back to an exclusive lock subrequest.
- An **AllowFallbackToExclusive** attribute value set to **false** indicates that a coauthoring subrequest is not allowed to fall back to an exclusive lock subrequest.

The **AllowFallbackToExclusive** attribute is specified as part of a coauthoring subrequest of type "Join coauthoring session". The types of coauthoring subrequest are defined in section 2.3.3.3.

ReleaseLockOnConversionToExclusiveFailure: A Boolean value that specifies to the protocol server whether the server is allowed to remove the **ClientID** entry associated with the current client in the **File coauthoring tracker**, provided that all of the following conditions are true:

- The type of coauthoring subrequest is "Convert to an exclusive lock".
- The conversion to an exclusive lock failed.

When all the preceding conditions are true, the following apply:

- A **ReleaseLockOnConversionToExclusiveFailure** attribute set to a value of **true** indicates that the protocol server is allowed to remove the **ClientID** entry associated with the current client in the **File coauthoring tracker**.
- A **ReleaseLockOnConversionToExclusiveFailure** attribute set to a value of **false** indicates that the protocol server is not allowed to remove the **ClientID** entry associated with the current client in the **File coauthoring tracker**.
- A **ReleaseLockOnConversionToExclusiveFailure** attribute MUST be sent only when the coauthoring subrequest type is set to "Convert to exclusive lock". The types of coauthoring subrequest are defined in section 2.3.3.3. The **File coauthoring tracker** is defined in section 3.1.1.

SchemaLockID: A string that is globally unique and known among all protocol clients that share the same protocol version. The schema lock identifier is used by the protocol server to block other clients with different schema identifiers. After a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST allow only other protocol clients that specify the same schema lock identifier to share the file lock. The protocol server ensures that at any instant of time, only clients having the same schema lock identifier can lock the file. After all the protocol clients have released their lock for that file, the protocol server MUST allow a protocol client with a different schema lock identifier to get a shared lock for that file. The **SchemaLockID** attribute MUST be sent on all types of coauthoring subrequests. The string "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use for this attribute.<13>

Timeout: An integer that specifies the time, in seconds, after which the shared lock for that particular file will expire for that specific protocol client. The **Timeout** attribute MUST be set to a value ranging from 3,600 to 120,000. When the **Timeout** attribute is set to a value ranging from 60 to 3600, the server also returns success but sets **Timeout** to an implementation-specific default value. When more than one client is editing the file, the protocol server MUST maintain a separate timeout value for each client in the **File coauthoring tracker**. The **File coauthoring tracker** is defined in section 3.1.1. The client's timeout on a shared lock for a file is refreshed by sending a coauthoring subrequest of type, "Refresh coauthoring session". The **Timeout** attribute MUST be sent in the following types of coauthoring subrequests:

- Join coauthoring session
- Refresh coauthoring session
- Convert to exclusive lock

The types of coauthoring subrequests are defined in section 2.3.3.3.

ExclusiveLockID: A string that serves as a unique identifier for the exclusive lock on the file when a coauthoring request of type "Convert to exclusive lock" is requested. **ExclusiveLockID** MUST be sent when the type of the coauthoring subrequest is "Convert to exclusive lock" or "Join coauthoring session" and the **AllowFallbackToExclusive** attribute is set to **true**.

2.3.1.6 CoauthSubRequestType

The **CoauthSubRequestType** complex type contains information about a coauthoring subrequest. The **SubRequestType** definition from which **CoauthSubRequestType** is extended is defined in section 2.2.4.3.

```
<xs:complexType name="CoauthSubRequestType">
  <xs:complexContent>
    <xs:extension base="tns:SubRequestType">
      <xs:sequence minOccurs="1" maxOccurs="1">
         <xs:element name="SubRequestData" type="tns:CoauthSubRequestDataType" />
      </xs:sequence>
      <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="Coauth" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

SubRequestData: A **CoauthSubRequestDataType** that specifies the data or input parameters needed for processing the coauthoring subrequest. **CoauthSubRequestDataType** is defined in section 2.3.1.5.

Type: A **SubRequestAttributeType** that specifies the type of the subrequest. The **Type** attribute MUST be set to "Coauth" for a coauthoring subrequest. **SubRequestAttributeType** is defined in section 2.2.5.11.

2.3.1.7 CoauthSubResponseDataType

The **CoauthSubResponseDataType** complex type contains information requested as part of the corresponding coauthoring subrequest.

```
<xs:complexType name="CoauthSubResponseDataType">
    <xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
    <xs:attribute name="CoauthStatus" type="tns:CoauthStatusType" use="optional"/>
    <xs:attribute name="TransitionID" type="tns:guid" use="optional"/>
    <xs:attribute name="ExclusiveLockReturnReason" type="tns:ExclusiveLockReturnReasonTypes"
use="optional" />
</xs:complexType>
```

LockType: A **LockTypes** that specifies the type of lock granted in a coauthoring subresponse. **LockTypes** is defined in section 2.2.5.9. If the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success", the **LockType** attribute MUST be specified in a coauthoring subresponse that is generated in response to all of the following types of coauthoring subrequests:

- Join coauthoring session
- Refresh coauthoring session

The different types of coauthoring subrequests are defined in section 2.3.3.3.

CoauthStatus: A **CoauthStatusType** that specifies the coauthoring status in a coauthoring subresponse. The **CoauthStatusType** is defined in section 2.2.5.1. If the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success", **CoauthStatus** MUST be specified in a coauthoring subresponse that is generated in response to all of the following types of coauthoring subrequests:

- Join coauthoring session<14>
- Refresh coauthoring session
- Get coauthoring status.

The different types of coauthoring subrequest are defined in section 2.3.3.3.

TransitionID: A **guid** that specifies the unique file identifier stored for that file on the protocol server. The **guid** type is defined in section 2.2.5.7. **TransitionID** serves as an input parameter to the **IsOnlyClient** web service request as specified in [MS-SHDACCWS]. The transition identifier MUST be returned by a coauthoring subrequest of type "Join coauthoring session".

ExclusiveLockReturnReason: An **ExclusiveLockReturnReasonTypes** that specifies the reason why an exclusive lock is granted in a coauthoring subresponse. **ExclusiveLockReturnReasonTypes** is defined in section 2.2.5.5. The **ExclusiveLockReturnReason** attribute MUST be specified in a coauthoring subresponse that is generated in response to the **JoinCoauthoring** type of coauthoring subrequest when the **LockType** attribute in the subresponse is set to "ExclusiveLock".

The types of coauthoring subrequests are defined in section 2.3.3.3.

2.3.1.8 CoauthSubResponseType

The **CoauthSubResponseType** complex type contains information about the success or failure in processing the coauthoring subrequest. In the case of success, it contains coauthoring information requested as part of the coauthoring subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this subrequest. **ErrorCode** is specified in section 2.2.4.6. The **SubResponseType** definition from which **CoauthSubResponseType** is extended is defined in section 2.2.4.6.

```
<xs:complexType name="CoauthSubResponseType">
  <xs:complexContent>
    <xs:extension base="tns:SubResponseType">
      <xs:sequence minOccurs="0" maxOccurs="1">
         <xs:element name="SubResponseData" type="tns:CoauthSubResponseDataType" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

SubResponseData: A **CoauthSubResponseDataType** that specifies coauthoring-related information provided by the protocol server that was requested as part of the coauthoring subrequest. **CoauthSubResponseDataType** is defined in section 2.3.1.7. As part of processing the coauthoring subrequest, the **SubResponseData** element MUST be sent as part of the **SubResponse** element in a cell storage service response message only if the following condition is true:

- The **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success".

The protocol server sets the value of the **ErrorCode** attribute to "Success" only if the protocol server succeeds in processing the coauthoring subrequest. **ErrorCode** is specified in section 2.2.4.6.

2.3.1.9 ExclusiveLockSubRequestDataType

The **ExclusiveLockSubRequestDataType** complex type contains information about data or input parameters used in processing an exclusive lock subrequest. The **ExclusiveLockID** attribute and the **ExclusiveLockRequestType** attribute specified in the **ExclusiveLockSubRequestDataOptionalAttributes** attribute group MUST both be specified for an exclusive lock subrequest. The **ExclusiveLockID** attribute and the **ExclusiveLockRequestType** attribute are specified as part of the **SubRequestData** element associated with an exclusive lock **SubRequest** element. **ExclusiveLockSubRequestDataOptionalAttributes** is defined in section 2.3.3.4. If the specified attributes are not provided, an "InvalidArgument" error code MUST be returned as part of the **SubResponseData** element associated with the exclusive lock subresponse.<15>

```
<xs:complexType name="ExclusiveLockSubRequestDataType">
    <xs:attributeGroup ref="tns:ExclusiveLockSubRequestDataOptionalAttributes"/>
    <xs:attribute name="ClientID" type="xs:string" use="optional"/>
    <xs:attribute name="SchemaLockID" type="xs:string" use="optional" />
    <xs:attribute name="Timeout" type="xs:integer" use="optional" />
    <xs:attribute name="ExclusiveLockID" type="xs:string" use="required"/>
</xs:complexType>
```

ExclusiveLockSubRequestDataOptionalAttributes: An attribute group that specifies the set of attributes that are provided only for a **SubRequestData** element whose parent **SubRequest** element's **Type** attribute is set to "ExclusiveLock". **ExclusiveLockSubRequestDataOptionalAttributes** is defined in section 2.3.3.4.

ClientID: A string that serves to uniquely identify each client that has access to a shared a lock on a coauthorable file. **ClientID** MUST be specified when the exclusive lock subrequest has an **ExclusiveLockSubRequestType** attribute set to "ConvertToSchemaLock" or "ConvertToSchemaJoinCoauth". **ExclusiveLockSubRequestType** specifies the different types of exclusive lock subrequest and is defined in section 2.3.3.4.

SchemaLockID: A string that is globally unique and known among all the protocol clients that share the same protocol version. The schema lock identifier is used by the protocol server to block other clients that have different schema identifiers. After a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST allow only other protocol clients that specify the same schema lock identifier to share the file lock. The protocol server ensures that at any instant in time, only clients having the same schema lock identifier can lock the document. After all the protocol clients have released their lock for that file, the protocol server MUST allow a protocol client with a different schema lock identifier to get a shared lock for that file. The **SchemaLockID** attribute MUST be sent when the exclusive lock subrequest has an **ExclusiveLockSubRequestType** attribute set to "ConvertToSchemaLock" or "ConvertToSchemaJoinCoauth". **ExclusiveLockSubRequestType** specifies the different types of exclusive lock subrequests and is defined in section 2.3.3.4. The string "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use for this attribute.<16>

Timeout: An integer that specifies the time, in seconds, after which the exclusive lock for that particular file will expire for that specific protocol client. The **Timeout** attribute MUST be set to a value ranging from 60 to 120,000. The **Timeout** attribute MUST be sent when an exclusive lock subrequest is one of the following types:

- Get lock
- Refresh lock
- Convert to schema lock
- Convert to schema lock with coauthoring transition tracked

The types of exclusive lock subrequest are defined in section 2.3.3.4.

ExclusiveLockID: A string that serves as a unique identifier for the exclusive lock on the file when an exclusive lock is requested. **ExclusiveLockID** MUST be specified on all types of exclusive lock subrequests.

2.3.1.10 ExclusiveLockSubRequestType

The **ExclusiveLockSubRequestType** complex type contains information about an exclusive lock subrequest. The **SubRequestType** definition from which **ExclusiveLockSubRequestType** is extended is defined in section 2.2.4.3.

```
<xs:complexType name="ExclusiveLockSubRequestType">
   <xs:complexContent>
     <xs:extension base="tns:SubRequestType">
       <xs:sequence minOccurs="1" maxOccurs="1">
         <xs:element name="SubRequestData" type="tns:ExclusiveLockSubRequestDataType" />
       </xs:sequence>
       <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="ExclusiveLock" />
     </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

SubRequestData: An **ExclusiveLockSubRequestDataType** that specifies the data or input parameters needed for processing the exclusive lock subrequest. **ExclusiveLockSubRequestDataType** is defined in section 2.3.1.9.

Type: A **SubRequestAttributeType** that specifies the type of the subrequest. The **Type** attribute MUST be set to "ExclusiveLock" for an exclusive lock subrequest. **SubRequestAttributeType** is defined in section 2.2.5.11.

2.3.1.11 ExclusiveLockSubResponseDataType

The **ExclusiveLockSubResponseDataType** complex type contains information requested as part of the corresponding exclusive lock subrequest.

```
<xs:complexType name="ExclusiveLockSubResponseDataType">
    <xs:attribute name="CoauthStatus" type="tns:CoauthStatusType" use="optional"/>
    <xs:attribute name="TransitionID" type="tns:guid" use="optional"/>
</xs:complexType>
```

CoauthStatus: A **CoauthStatusType** that specifies the coauthoring status in an exclusive lock subresponse. **CoauthStatusType** is defined in section 2.2.5.1. **CoauthStatus** MUST be specified only in an exclusive lock subresponse that is generated in response to an exclusive lock subrequest of type "Convert to schema lock with coauthoring transition tracked" if the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success". The types of exclusive lock subrequest are defined in section 2.3.3.4.

TransitionID: A **guid** that specifies the unique file identifier for that file in the protocol server. The **guid** type is defined in section 2.2.5.7. **TransitionID** MUST be returned as part of the response for an exclusive lock subrequest of type "Convert to Schema lock with coauthoring transition tracked" if the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success". **TransitionID** serves as an input parameter to the **IsOnlyClient** web service request as specified in [MS-SHDACCWS].

2.3.1.12 ExclusiveLockSubResponseType

The **ExclusiveLockSubResponseType** complex type contains information about the success or failure in processing the exclusive lock subrequest. In the case of success, it contains information requested as part of the exclusive lock subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this subrequest. **ErrorCode** is specified in section 2.2.4.6. The **SubResponseType** definition from which **ExclusiveLockSubResponseType** is extended is defined in section 2.2.4.6.

```
<xs:complexType name="ExclusiveLockSubResponseType">
 <xs:complexContent>
   <xs:extension base="tns:SubResponseType">
    <xs:sequence minOccurs="1" maxOccurs="1">
```

```
      <xs:element name="SubResponseData" type="tns:ExclusiveLockSubResponseDataType" />
     </xs:sequence>
    </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

SubResponseData: An **ExclusiveLockSubResponseDataType** that specifies exclusive lock–related information provided by the protocol server that was requested as part of the exclusive lock subrequest. **ExclusiveLockSubResponseDataType** is defined in section 2.3.1.11.

2.3.1.13 SchemaLockSubRequestDataType

The **SchemaLockSubRequestDataType** complex type contains information about data or input parameters used in processing a schema lock subrequest. The **SchemaLockID** attribute and the **SchemaLockRequestType** attribute specified in the **SchemaLockSubRequestDataOptionalAttributes** attribute group MUST both be specified for a schema lock subrequest. The **SchemaLockID** attribute and the **SchemaLockRequestType** attribute are specified as part of the **SubRequestData** element associated with a schema lock **SubRequest** element. **SchemaLockSubRequestDataOptionalAttributes** is defined in section 2.3.3.5. If the specified attributes are not provided, an "InvalidArgument" <17>error code MUST be returned as part of the **SubResponseData** element associated with the schema lock subresponse.

```
<xs:complexType name="SchemaLockSubRequestDataType">
    <xs:attributeGroup ref="tns:SchemaLockSubRequestDataOptionalAttributes"/>
    <xs:attribute name="ClientID" type="xs:string" use="optional"/>
    <xs:attribute name="AllowFallbackToExclusive" type="xs:boolean" use="optional" />
    <xs:attribute name="ReleaseLockOnConversionToExclusiveFailure" type="xs:boolean"
use="optional"/>
    <xs:attribute name="SchemaLockID" type="xs:string" use="required" />
    <xs:attribute name="Timeout" type="xs:integer" use="optional" />
    <xs:attribute name="ExclusiveLockID" type="xs:string" use="optional"/>
</xs:complexType>
```

SchemaLockSubRequestDataOptionalAttributes: An attribute group that specifies the set of attributes that are provided only for a **SubRequestData** element whose parent **SubRequest** element's **Type** attribute is set to "SchemaLock". **SchemaLockSubRequestDataOptionalAttributes** is defined in section 2.3.3.5.

ClientID: A string that serves to uniquely identify each client that has access to a shared lock on a coauthorable file. **ClientID** MUST be specified on all types of schema lock subrequests. The different types of schema lock subrequest are defined in section 2.3.3.5.

AllowFallbackToExclusive: A Boolean value that specifies to a protocol server whether a schema lock subrequest is allowed to fall back to an exclusive lock subrequest provided that shared locking on the file is not supported. When shared locking on the file is not supported:

- An **AllowFallbackToExclusive** attribute value set to **true** indicates that a schema lock subrequest is allowed to fall back to an exclusive lock subrequest.
- An **AllowFallbackToExclusive** attribute value set to **false** indicates that a schema lock subrequest is not allowed to fall back to an exclusive lock subrequest.

The AllowFallbackToExclusive attribute is specified as part of a schema lock subrequest of type "Get Lock". The types of schema lock subrequest are defined in section 2.3.3.5.

ReleaseLockOnConversionToExclusiveFailure: A Boolean value that specifies to the protocol server whether the server is allowed to remove the **ClientID** entry associated with the current client in the **File coauthoring tracker**, provided that all of the following conditions are true:

- The type of the schema lock subrequest is "Convert to an Exclusive Lock".
- The conversion to an exclusive lock failed.

When all the preceding conditions are true, the following apply:

- A **ReleaseLockOnConversionToExclusiveFailure** attribute set to a value of **true** indicates that the protocol server is allowed to remove the **ClientID** entry associated with the current client in the **File coauthoring tracker**.
- A **ReleaseLockOnConversionToExclusiveFailure** attribute set to a value of **false** indicates that the protocol server is not allowed to remove the **ClientID** entry associated with the current client in the **File coauthoring tracker**.

The **ReleaseLockOnConversionToExclusiveFailure** attribute MUST be sent only when the schema lock subrequest type is set to "Convert to exclusive lock". The types of schema lock subrequests are defined in section 2.3.3.5. The **File coauthoring tracker** is defined in section 3.1.1.

SchemaLockID: A string that is globally unique and known among all protocol clients that share the same protocol version. The schema lock identifier is used by the protocol server to block other clients that have different schema identifiers. After a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST allow only other protocol clients that specify the same schema lock identifier to share the file lock. The protocol server ensures that at any instant in time, only clients having the same schema lock identifier can lock the file. After all the protocol clients have released their lock for that file, the protocol server MUST allow a protocol client with a different schema lock identifier to get a shared lock for that file. The **SchemaLockID** attribute MUST be sent on all types of schema lock subrequests. The string "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use for this attribute.<18>

Timeout: An integer that specifies the time, in seconds, after which the shared lock for that particular file will expire for that specific protocol client. The **Timeout** attribute MUST be set to a value ranging from 3,600 to 120,000. When the Timeout is set to a value ranging from 60 to 3600, the server also returns success but sets the Timeout to an implementation-specific default value. When more than one client is editing the file, the protocol server MUST maintain a separate timeout value for each client. The client's timeout on a shared lock for a file is refreshed by sending a schema lock subrequest of type "Refresh lock". The **Timeout** attribute MUST be specified in all of the following types of schema lock subrequests:

- Get lock
- Refresh lock
- Convert to exclusive lock

The types of schema lock subrequests are defined in section 2.3.3.5.

ExclusiveLockID: A string that serves as a unique identifier for the exclusive lock on the file when a schema lock subrequest of type "Convert to exclusive lock" is requested. **ExclusiveLockID** MUST be specified when the type of the schema lock subrequest is "Convert to exclusive lock" or "Get lock" and the **AllowFallbackToExclusive** attribute is set to **true**.

2.3.1.14 SchemaLockSubRequestType

The **SchemaLockSubRequestType** complex type contains information about a schema lock subrequest. The **SubRequestType** definition from which **SchemaLockSubRequestType** is extended is defined in section 2.2.4.3.

```
<xs:complexType name="SchemaLockSubRequestType">
 <xs:complexContent>
  <xs:extension base="tns:SubRequestType">
```

```
    <xs:sequence minOccurs="1" maxOccurs="1">
      <xs:element name="SubRequestData" type="tns:SchemaLockSubRequestDataType" />
    </xs:sequence>
    <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="SchemaLock" />
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

SubRequestData: A **SchemaLockSubRequestDataType** that specifies the data or input parameters needed for processing the schema lock subrequest. **SchemaLockSubRequestDataType** is defined in section 2.3.1.13.

Type: A **SubRequestAttributeType** that specifies the type of the subrequest. The **Type** attribute MUST be set to "SchemaLock" for a schema lock subrequest. **SubRequestAttributeType** is defined in section 2.2.5.11.

2.3.1.15 SchemaLockSubResponseDataType

The **SchemaLockSubResponseDataType** complex type contains information requested as part of the corresponding schema lock subrequest.

```
<xs:complexType name="SchemaLockSubResponseDataType">
    <xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
    <xs:attribute name="ExclusiveLockReturnReason" type="tns:ExclusiveLockReturnReasonTypes"
use="optional" />
</xs:complexType>
```

LockType: A **LockTypes** that specifies the type of lock granted in a schema lock subresponse. **LockTypes** is defined in section 2.2.5.9. If the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success", **LockType** MUST be specified in a schema lock subresponse that is generated in response to a schema lock subrequest of type "Get lock" or "Refresh lock". The types of schema lock subrequests are defined in section 2.3.3.5.

ExclusiveLockReturnReason: An **ExclusiveLockReturnReasonTypes** that specifies the reason why an exclusive lock is granted in a schema lock subresponse. **ExclusiveLockReturnReasonTypes** is defined in section 2.2.5.5. The **ExclusiveLockReturnReason** attribute MUST be specified in a schema lock subresponse that is generated in response to a schema lock subrequest of type "Get lock" when the **LockType** attribute in the subresponse is set to "ExclusiveLock". The types of schema lock subrequests are defined in section 2.3.3.5.

2.3.1.16 SchemaLockSubResponseType

The **SchemaLockSubResponseType** complex type contains information about the success or failure in processing the schema lock subrequest. In the case of success, it contains information requested as part of the schema lock subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this subrequest. **ErrorCode** is specified in section 2.2.4.6. The **SubResponseType** definition from which **SchemaLockSubResponseType** is extended is defined in section 2.2.4.6.

```
<xs:complexType name="SchemaLockSubResponseType">
 <xs:complexContent>
  <xs:extension base="tns:SubResponseType">
   <xs:sequence minOccurs="1" maxOccurs="1">
     <xs:element name="SubResponseData" type="tns:SchemaLockSubResponseDataType" />
   </xs:sequence>
  </xs:extension>
 </xs:complexContent>
```

```
</xs:complexType>
```

SubResponseData: A **SchemaLockSubResponseDataType** that specifies schema lock–related information provided by the protocol server that was requested as part of the schema lock subrequest. **SchemaLockSubResponseDataType** is defined in section 2.3.1.15.

2.3.1.17 ServerTimeSubRequestType

The **ServerTimeSubRequestType** complex type contains information about a server time subrequest. The **SubRequestType** definition from which **ServerTimeSubRequestType** is extended is defined in section 2.2.4.3. The **SubRequestData** element is not contained in a **SubRequest** element of type **ServerTimeSubRequestType**.

```
<xs:complexType name="ServerTimeSubRequestType">
  <xs:complexContent>
    <xs:extension base="tns:SubRequestType">
      <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="ServerTime" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Type: A **SubRequestAttributeType** that specifies the type of subrequest. The **Type** attribute MUST be set to "ServerTime" for a server time subrequest. **SubRequestAttributeType** is defined in section 2.2.5.11.

2.3.1.18 ServerTimeSubResponseDataType

The **ServerTimeSubResponseDataType** complex type contains server time–specific information requested as part of the corresponding server time subrequest.

```
<xs:complexType name="ServerTimeSubResponseDataType">
    <xs:attribute name="ServerTime" type="xs:positiveInteger" use="optional"/>
</xs:complexType>
```

ServerTime: A positive integer that specifies the server time, which is expressed as a tick count. A single tick represents 100 nanoseconds, or one ten-millionth of a second. **ServerTime** specifies the number of 100-nanosecond intervals that have elapsed since 00:00:00 on January 1, 1601, which SHOULD<19> be Coordinated Universal Time (UTC). If the request for server time information from the server is successful, the **ServerTime** attribute MUST be specified in a server time subresponse that is generated in response to a server time subrequest.

2.3.1.19 ServerTimeSubResponseType

The **ServerTimeSubResponseType** complex type contains information about the success or failure in processing the server time subrequest. In the case of success, it contains information requested as part of a server time subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this subrequest. **ErrorCode** is specified in section 2.2.4.6. The **SubResponseType** definition from which **ServerTimeSubResponseType** is extended is defined in section 2.2.4.6.

```
<xs:complexType name="ServerTimeSubResponseType">
  <xs:complexContent>
    <xs:extension base="tns:SubResponseType">
     <xs:sequence minOccurs="1" maxOccurs="1">
```

```
        <xs:element name="SubResponseData" type="tns:ServerTimeSubResponseDataType"/>
      </xs:sequence>
     </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

SubResponseData: A **ServerTimeSubResponseDataType** that specifies server time–specific information provided by the protocol server that was requested as part of the server time subrequest. **ServerTimeSubResponseDataType** is defined in section 2.3.1.18.

2.3.1.20 WhoAmISubRequestType

The **WhoAmISubRequestType** complex type contains information about Who Am I subrequest. The **SubRequestType** definition from which **WhoAmISubRequestType** is extended is defined in section 2.2.4.3. The **SubRequestData** element is not contained in a **SubRequest** element of type, **WhoAmISubRequestType**.

```
<xs:complexType name="WhoAmISubRequestType">
  <xs:complexContent>
     <xs:extension base="tns:SubRequestType">
       <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="WhoAmI" />
     </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Type: A **SubRequestAttributeType** that specifies the type of subrequest. The **Type** attribute MUST be set to "WhoAmI" for a Who Am I subrequest. **SubRequestAttributeType** is defined in section 2.2.5.11.

2.3.1.21 WhoAmISubResponseDataType

The **WhoAmISubResponseDataType** complex type contains client-specific information requested as part of the corresponding **WhoAmI** subrequest.

```
<xs:complexType name="WhoAmISubResponseDataType">
    <xs:attributeGroup ref="tns:WhoAmISubResponseDataOptionalAttributes"/>
</xs:complexType>
```

WhoAmISubResponseDataOptionalAttributes: An attribute group that specifies the set of attributes that are provided for a **SubResponseData** element whose parent **SubResponse** element's mapping **SubRequest** element is a **WhoAmI** subrequest. **WhoAmISubResponseDataOptionalAttributes** is defined in section 2.3.3.6.

2.3.1.22 WhoAmISubResponseType

The **WhoAmISubResponseType** complex type contains information about the success or failure in processing a **WhoAmI** subrequest. In the case of success, it contains information requested as part of **WhoAmI** subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this subrequest. **ErrorCode** is specified in section 2.2.4.6. The **SubResponseType** definition from which **WhoAmISubResponseType** is extended is defined in section 2.2.4.6.

```
<xs:complexType name="WhoAmISubResponseType">
  <xs:complexContent>
```

```
      <xs:extension base="tns:SubResponseType">
        <xs:sequence minOccurs="0" maxOccurs="1">
           <xs:element name="SubResponseData" type="tns:WhoAmISubResponseDataType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

SubResponseData: A **WhoAmISubResponseDataType** that specifies client-specific information provided by the protocol server that was requested as part of the **WhoAmI** subrequest. **WhoAmISubResponseDataType** is defined in section 2.3.1.21. As part of processing the **WhoAmI** subrequest, the **SubResponseData** element MUST be sent as part of the **SubResponse** element in a cell storage service response message only if the following condition is true:

- The **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success".

The protocol server sets the value of the **ErrorCode** attribute to "Success" only if the protocol server succeeds in processing the **WhoAmI** subrequest. **ErrorCode** is specified in section 2.2.4.6.

2.3.1.23 EditorsTableSubRequestDataType

The **EditorsTableSubRequestDataType** complex type contains information about data or input parameters used in processing an editors table subrequest. The **ClientID** attribute and the **EditorsTableRequestType** attribute specified in the **EditorsTableSubRequestDataOptionalAttributes** attribute group MUST both be specified for an editors table subrequest. **ClientID** and **EditorsTableRequestType** are specified as part of the **SubRequestData** element associated with an editors table **SubRequest** element. **EditorsTableSubRequestDataOptionalAttributes** is defined in section 2.3.3.7. If the specified attributes are not provided, an "InvalidArgument" error code MUST be returned as part of the **SubResponseData** element associated with the editors table subresponse.

```
<xs:complexType name="EditorsTableSubRequestDataType" mixed="true">
    <xs:attributeGroup ref="tns:EditorsTableSubRequestDataOptionalAttributes"/>
    <xs:attribute name="ClientID" type="xs:string" use="required"/>
    <xs:attribute name="AsEditor" type="xs:boolean" use="optional" />
    <xs:attribute name="Timeout" type="xs:integer" use="optional" />
    <xs:attribute name="Key" type="xs:string" use="optional" />
    <xs:attribute name="Value" type="xs:binary" use="optional" />
</xs:complexType>
```

SchemaLockSubRequestDataOptionalAttributes: An attribute group that specifies the set of attributes that are provided only for a **SubRequestData** element whose parent **SubRequest** element's **Type** attribute is set to "EditorsTable". **EditorsTableSubRequestDataOptionalAttributes** is defined in section 2.3.3.7.

ClientID: A string that serves to uniquely identify each client that has access to an editors table on a coauthorable file. **ClientID** MUST be specified on all types of editors table subrequests. The different types of editors table subrequest are defined in section 2.3.3.7.

AsEditor: A Boolean value that specifies to the server whether the client is opening the document as an editor or as a reader. The server MUST NOT allow a user with read-only access to join the editing session as a reader. The **AsEditor** attribute MUST be specified in all of the following types of editors table subrequests:

- Join editing session
- Refresh editing session

The types of editors table subrequests are defined in section 2.3.3.7.

Timeout: An integer that specifies the time, in seconds, after which the editors table entry for that particular file will expire for that specific protocol client. The **Timeout** attribute MUST be set to a value ranging from 3,600 to 120,000. When the Timeout is set to a value ranging from 60 to 3600, the server also returns success but sets the Timeout to an implementation-specific default value. When more than one client is editing the file, the protocol server MUST maintain a separate timeout value for each client. The client's timeout on an editors table entry for a file is refreshed by sending an editors table subrequest of type "Refresh editing session". The **Timeout** attribute MUST be specified in all of the following types of editors table subrequests:

- Join editing session
- Refresh editing session

The types of editors table subrequests are defined in section 2.3.3.7.

Key: A string that specifies a unique key in an arbitrary key/value pair of the client's choice, the length for **Key** is limited to 64 bytes and at most 4 key/value pairs can be associated with a given editor. The server stores this key/value pair for that particular file for that specific protocol client. These pairs are visible to other clients editing or reading the same document. The **Key** attribute MUST be specified in all of the following types of editors table subrequests:

- Update Editor Metadata
- Remove Editor Metadata

The types of editors table subrequests are defined in section 2.3.3.7.

Value: Ignore.

Text: A binary value that is associated with a key in an arbitrary key/value pair of the client's choice. The length for **Value** is limited to 1024 bytes. The server stores this key/value pair for that particular file for that specific protocol client. These pairs are visible to other clients editing or reading the same document. The text of the **SubRequestData** element MUST be specified in an **EditorsTable** (section 3.1.4.8) subrequest type of "Update Editor Metadata". This text is **base64** binary encoded data and indicates the value of the **Value** attribute.

The types of **EditorsTable** subrequests are defined in section 2.3.3.7.

2.3.1.24 EditorsTableSubRequestType

The **EditorsTableSubRequestType** complex type contains information about an editors table subrequest. The **SubRequestType** definition from which **WhoAmISubRequestType** is extended is defined in section 2.2.4.3.

```
<xs:complexType name="EditorsTableSubRequestType">
 <xs:complexContent>
  <xs:extension base="tns:SubRequestType">
   <xs:sequence minOccurs="1" maxOccurs="1">
     <xs:element name="SubRequestData" type="tns:EditorsTableSubRequestDataType" />
   </xs:sequence>
   <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="EditorsTable" />
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

SubRequestData: An **EditorsTableSubRequestDataType** that specifies the data or input parameters needed for processing the editors table subrequest. **EditorsTableSubRequestDataType** is defined in section 2.3.1.23.

Type: A **SubRequestAttributeType** that specifies the type of the subrequest. The **Type** attribute MUST be set to "EditorsTable" for an editors table subrequest. **SubRequestAttributeType** is defined in section 2.2.5.11.

2.3.1.25 EditorsTableSubResponseType

The **EditorsTableSubResponseType** complex type contains information about the success or failure in processing an **EditorsTable** (section 3.1.4.8) subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this subrequest. The **ErrorCode** attribute is specified in section 2.2.4.6. The **SubResponseType** definition from which **EditorsTableSubResponseType** is extended is defined in section 2.2.4.6.

```
<xs:complexType name="EditorsTableSubResponseType">
  <xs:complexContent>
    <xs:extension base="tns:SubResponseType">
       <xs:sequence minOccurs="0" maxOccurs="1">
          <xs:element name="SubResponseData">
            <xs:complexType>
              <xs:complexContent>
                <xs:restriction base="xs:anyType"/>
              </xs:complexContent>
            </xs:complexType>
          </xs:element>
        </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

SubResponseData: It MUST be an empty element without any attributes. As part of processing the **EditorsTable** subrequest, the **SubResponseData** element MUST be sent as part of the **SubResponse** element in a cell storage service response message only if the following condition is true:

- The **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success".

The protocol server sets the value of the **ErrorCode** attribute to "Success" only if the protocol server succeeds in processing the editors table subrequest. **ErrorCode** is specified in 2.2.4.6.

2.3.1.26 GetDocMetaInfoSubRequestType

The **GetDocMetaInfoSubRequestType** complex type contains information about a **GetDocMetaInfo** subrequest. The **SubRequestType** definition from which **GetDocMetaInfoSubRequestType** is extended is defined in section 2.2.4.3. The **SubRequestData** element is not contained in a **SubRequest** element of type **GetDocMetaInfoSubRequestType**.

```
<xs:complexType name="GetDocMetaInfoSubRequestType">
  <xs:complexContent>
    <xs:extension base="tns:SubRequestType">
      <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="GetDocMetaInfo" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

Type: A **SubRequestAttributeType** that specifies the type of the subrequest. The **Type** attribute MUST be set to "GetDocMetaInfo" for a Get Doc Meta Info subrequest. **SubRequestAttributeType** is defined in section 2.2.5.11.

2.3.1.27 GetDocMetaInfoSubResponseDataType

The **GetDocMetaInfoSubResponseDataType** complex type contains no additional data beyond the **SubResponseDataGenericType** that it extends. **SubResponseDataGenericType** is defined in section 2.2.4.4.

```
<xs:complexType name="GetDocMetaInfoSubResponseDataType">
    <xs:sequence>
      <xs:element name="DocProps" type="tns:GetDocMetaInfoPropertySetType"/>
      <xs:element name="FolderProps" type="tns:GetDocMetaInfoPropertySetType"/>
    </xs:sequence>
</xs:complexType>
```

DocProps: An element of type **GetDocMetaInfoPropertySetType** (section 2.3.1.28) that specifies metadata properties pertaining to the server file.

FolderProps: An element of type **GetDocMetaInfoPropertySetType** (section 2.3.1.28) that specifies metadata properties pertaining to the parent directory of the server file.

2.3.1.28 GetDocMetaInfoPropertySetType

The **GetDocMetaInfoPropertySetType** complex type contains a sequence of **Property** elements to describe the set of metainfo related to the file.

```
<xs:complexType name="GetDocMetaInfoPropertySetType">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element name="Property" type="tns:GetDocMetaInfoPropertyType"/>
    </xs:sequence>
</xs:complexType>
```

Property: A given metainfo property. **GetDocMetaInfoPropertyType** is defined in section 2.3.1.29.

2.3.1.29 GetDocMetaInfoPropertyType

The **GetDocMetaInfoProperty** complex type contains a metainfo key/value pair that is related either to the file against which the request is made or its parent directory as part of the corresponding **GetDocMetaInfo** subrequest.

```
<xs:complexType name="GetDocMetaInfoPropertyType">
    <xs:attribute name="Key" type="xs:string" use="required"/>
    <xs:attribute name="Value" type="xs:string" use="required"/>
</xs:complexType>
```

Key: A string as specified in [MS-FPSE] section 2.2.4 that describes the metainfo described in this property.

Value: A string as specified in [MS-FPSE] section 2.2.4 that describes the value of this property.

2.3.1.30 GetDocMetaInfoSubResponseType

The **GetDocMetaInfoSubResponseType** complex type contains information about the success or failure in processing **GetDocMetaInfo** subrequest. In the case of success, its child elements contain information requested as part of the **GetDocMetaInfo** subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this subrequest. The **ErrorCode** attribute is specified in section 2.2.4.6. **SubResponseType** is defined in section 2.2.4.6.

```
<xs:complexType name="GetDocMetaInfoSubResponseType">
  <xs:complexContent>
    <xs:extension base="tns:SubResponseType">
      <xs:sequence minOccurs="0" maxOccurs="1">
          <xs:element name="SubResponseData" type="tns:GetDocMetaInfoSubResponseDataType"/>
       </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

SubResponseData: A **SubResponseType** whose child nodes contain information provided by the protocol server that was requested as part of the **GetDocMetaInfo** subrequest. As part of processing the **GetDocMetaInfo** subrequest, the **SubResponseData** element MUST be sent as part of the **SubResponse** element in a cell storage service response message only if the following condition is true:

- The **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success".

The protocol server sets the value of the **ErrorCode** attribute to "Success" only if the protocol server succeeds in processing the **GetDocMetaInfo** subrequest. The **ErrorCode** attribute is specified in section 2.2.4.6.

2.3.1.31 GetVersionsSubRequestType

The **GetVersionsSubRequestType** complex type contains information about the **GetVersions** subrequest. The **SubRequestType** definition from which **GetVersionsSubRequestType** is extended is defined in section 2.2.4.3.

```
<xs:complexType name="GetVersionsSubRequestType">


  <xs:complexContent>
    <xs:extension base="tns:SubRequestType">
      <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="GetVersions" />
          </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

2.3.1.32 GetVersionsSubResponseType

The **GetVersionsSubResponseType** complex type contains information about the success or failure in processing the **GetVersions** subrequest. In the case of success, it contains information requested as part of the **GetVersions** subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this subrequest. **ErrorCode** is specified in section 2.2.4.6. The **SubResponseType** definition from which **GetVersionsSubResponseType** is extended is defined in section 2.2.4.6.

```
<xs:complexType name="GetVersionsSubResponseType">
  <xs:complexContent>
    <xs:extension base="tns:SubResponseType">
      <xs:sequence minOccurs="0" maxOccurs="1">
         <xs:element ref="tns:GetVersionsResponse"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```

GetVersionsResponse: An element that specifies information about a file's versions, as specified in [MS-VERSS] section 3.1.4.3.2.2.

2.3.2 Simple Types

The following table summarizes the set of other XML schema simple type definitions defined by this specification.

Simple type	Description
CellRequestErrorCodeTypes	A subset of error codes returned for a cell subrequest as part of a cell storage service response message. **CellRequestErrorCodeTypes** is an enumeration of error codes specific to a cell subrequest.
CoauthRequestTypes	The type of the **CoauthRequestType** attribute, which is part of a coauthoring subrequest. **CoauthRequestTypes** is an enumeration of all the coauthoring request types.
ExclusiveLockRequestTypes	The type of the **ExclusiveLockRequestType** attribute, which is part of an exclusive lock subrequest. **ExclusiveLockRequestTypes** is an enumeration of all the exclusive lock request types.
SchemaLockRequestTypes	The type of the **SchemaLockRequestType** attribute, which is part of a schema lock subrequest. **SchemaLockRequestTypes** is an enumeration of all the schema lock request types.
UserLoginType	A user login value.
UserNameType	A user name value.

2.3.2.1 CellRequestErrorCodeTypes

The **CellRequestErrorCodeTypes** simple type is used to represent error codes that occur during cell subrequest processing.

```
<xs:simpleType name="CellRequestErrorCodeTypes">
  <xs:restriction base="xs:string">
    <!--cell request fail-->
    <xs:enumeration value="CellRequestFail"/>
       <!--cell request etag not matching-->
    <xs:enumeration value="IRMDocLibarysOnlySupportWebDAV"/>
  </xs:restriction>
</xs:simpleType>
```

The value of **CellRequestErrorCodeTypes** MUST be one of the values in the following table.

Value	Meaning
"CellRequestFail"	Indicates an error when processing a cell subrequest for the given URL for the file.
"IRMDocLibarysOnlySupportWebDAV"	Indicates an error when the requested file is an **Information Rights Management (IRM)** protected document that is supported only through **Web Distributed Authoring and Versioning Protocol (WebDAV)**.

2.3.2.2 CoauthRequestTypes

The **CoauthRequestTypes** simple type is used to represent the type of coauthoring subrequest. **CoauthRequestTypes** is the type definition of the **CoauthRequestType** attribute, which is part of a coauthoring subrequest operation.

```
<xs:simpleType name="CoauthRequestTypes">
    <xs:restriction base="xs:string">
       <!--JoinCoauthoring-->
       <xs:enumeration value="JoinCoauthoring"/>
       <!--ExitCoauthoring-->
       <xs:enumeration value="ExitCoauthoring"/>
       <!--RefreshCoauthoring-->
       <xs:enumeration value="RefreshCoauthoring"/>
       <!-- ConvertToExclusive-->
       <xs:enumeration value="ConvertToExclusive"/>
       <!--CheckLockAvailability-->
       <xs:enumeration value="CheckLockAvailability"/>
       <!--MarkTransitionComplete-->
       <xs:enumeration value="MarkTransitionComplete"/>
       <!-- GetCoauthoringStatus-->
       <xs:enumeration value="GetCoauthoringStatus"/>
    </xs:restriction>
</xs:simpleType>
```

The value of **CoauthRequestTypes** MUST be one of the values in the following table.

Value	Meaning
"JoinCoauthoring"	The string value "JoinCoauthoring", specifying a coauthoring subrequest of type Join Coauthoring Session.
"ExitCoauthoring"	The string value "ExitCoauthoring", specifying a coauthoring subrequest of type Exit Coauthoring Session.
"RefreshCoauthoring "	The string value "RefreshCoauthoring", specifying a coauthoring subrequest of type Refresh Coauthoring Session.
"ConvertToExclusive"	The string value "ConvertToExclusive", specifying a coauthoring subrequest of type Convert To Exclusive Lock.
"CheckLockAvailability"	The string value "CheckLockAvailability", specifying a coauthoring subrequest of type Check Lock Availability.
"MarkTransitionComplete"	The string value "MarkTransitionComplete ", specifying a coauthoring subrequest of type Mark Transition To Complete.
"GetCoauthoringStatus"	The string value "GetCoauthoringStatus", specifying a

	coauthoring subrequest of type Get Coauthoring Status.

2.3.2.3 ExclusiveLockRequestTypes

The **ExclusiveLockRequestTypes** simple type is used to represent the type of exclusive lock subrequest. **ExclusiveLockRequestTypes** is the type definition of the **ExclusiveLockRequestType** attribute, which is part of an exclusive lock subrequest operation.

```
<xs:simpleType name="ExclusiveLockRequestTypes">
    <xs:restriction base="xs:string">
      <!--GetLock-->
      <xs:enumeration value="GetLock"/>
      <!--ReleaseLock-->
      <xs:enumeration value="ReleaseLock"/>
      <!--RefreshLock-->
      <xs:enumeration value="RefreshLock"/>
      <!--ConvertToSchemaJoinCoauth-->
      <xs:enumeration value="ConvertToSchemaJoinCoauth"/>
      <!--ConvertToSchemaLock-->
      <xs:enumeration value="ConvertToSchema"/>
      <!--CheckLockAvailability-->
      <xs:enumeration value="CheckLockAvailability"/>
    </xs:restriction>
</xs:simpleType>
```

The value of **ExclusiveLockRequestTypes** MUST be one of the values in the following table.

Value	Meaning
"GetLock"	The string value "GetLock", indicating an exclusive lock subrequest of type Get Lock.
"ReleaseLock"	The string value "ReleaseLock", indicating an exclusive lock subrequest of type Release Lock.
"RefreshLock"	The string value "RefreshLock", indicating an exclusive lock subrequest of type Refresh Lock.
"ConvertToSchemaJoinCoauth"	The string value "ConvertToSchemaJoinCoauth", indicating an exclusive lock subrequest of type Convert To Schema Lock With Coauthoring Transition Tracked.
"ConvertToSchema"	The string value "ConvertToSchema", indicating an exclusive lock subrequest of type Convert To Schema Lock.
"CheckLockAvailability"	The string value "CheckLockAvailability", indicating an exclusive lock subrequest of type Check Lock Availability.

2.3.2.4 SchemaLockRequestTypes

The **SchemaLockRequestTypes** simple type is used to represent the type of schema lock subrequest. **SchemaLockRequestTypes** is the type definition of the **SchemaLockRequestType** attribute, which is part of a schema lock subrequest operation.

```
<xs:simpleType name="SchemaLockRequestTypes">
```

```
    <xs:restriction base="xs:string">
      <!--GetLock-->
      <xs:enumeration value="GetLock"/>
      <!--ReleaseLock-->
      <xs:enumeration value="ReleaseLock"/>
      <!--RefreshLock-->
      <xs:enumeration value="RefreshLock"/>
      <!--ConvertToExclusiveLock,-->
      <xs:enumeration value="ConvertToExclusive"/>
      <!--CheckLockAvailability-->
      <xs:enumeration value="CheckLockAvailability"/>
    </xs:restriction>
</xs:simpleType>
```

The value of **SchemaLockRequestTypes** MUST be one of the values in the following table.

Value	Meaning
"GetLock"	The string value "GetLock", indicating a schema lock subrequest of type Get Lock.
"ReleaseLock"	The string value "ReleaseLock", indicating a schema lock subrequest of type Release Lock.
"RefreshLock"	The string value "RefreshLock ", indicating a schema lock subrequest of type Refresh Lock.
"ConvertToExclusive"	The string value "ConvertToExclusive", indicating a schema lock subrequest of type Convert To Exclusive Lock.
"CheckLockAvailability"	The string value "CheckLockAvailability", indicating a schema lock subrequest of type Check Lock Availability.

2.3.2.5 EditorsTableRequestTypes

The **EditorsTableRequestType** simple type is used to represent the type of editors table subrequest. **EditorsTableRequestTypes** is the type definition of the **EditorsTableRequestType** attribute, which is part of an editors table subrequest operation.

```
<xs:simpleType name="EditorsTableRequestTypes">
    <xs:restriction base="xs:string">
      <!--JoinEditingSession-->
      <xs:enumeration value="JoinEditingSession"/>
      <!--LeaveEditingSession-->
      <xs:enumeration value="LeaveEditingSession"/>
      <!--RefreshEditingSession-->
      <xs:enumeration value="RefreshEditingSession"/>
      <!--UpdateEditorMetadata-->
      <xs:enumeration value="UpdateEditorMetadata"/>
      <!--RemoveEditorMetadata-->
      <xs:enumeration value="RemoveEditorMetadata"/>
    </xs:restriction>
</xs:simpleType>
```

The value of **EditorsTableRequestType** MUST be one of the values in the following table.

Value	Meaning
"JoinEditingSession"	The string value "JoinEditingSession", indicating an editors table subrequest of type Join Editing Session.

"LeaveEditingSession"	The string value "LeaveEditingSession", indicating an editors table subrequest of type Leave Editing Session.
"RefreshEditingSession"	The string value "RefreshEditingSession", indicating an editors table subrequest of type Refresh Editing Session.
"UpdateEditorMetadata"	The string value "UpdateEditorMetadata", indicating an editors table subrequest of type Update Editor Metadata.
"RemoveEditorMetadata"	The string value "RemoveEditorMetadata", indicating an editors table subrequest of type Remove Editor Metadata.

2.3.2.6 UserLoginType

The **UserLoginType** simple type specifies a representation of a user login value as specified in [RFC2822].

```
<xs:simpleType name="UserLoginType">
   <xs:restriction base="xs:string">
   </xs:restriction>
</xs:simpleType>
```

UserLoginType is the type definition for the **UserLogin** attribute, which is part of the subresponse for a Who Am I subrequest.<20> For example, "contoso\user01".

2.3.2.7 UserNameType

The **UserNameType** simple type specifies a representation of a user name value as specified in [RFC2822].

```
<xs:simpleType name="UserNameType">
    <xs:restriction base="xs:string">
    </xs:restriction>
</xs:simpleType>
```

UserNameType is the type definition of the **UserName** attribute, which is part of the subresponse for a Who Am I subrequest.

2.3.3 Attribute Groups

The following table summarizes the set of other XML schema attribute group definitions defined by this specification.

Attribute	Description
CellSubRequestDataOptionalAttributes	Contains XML schema attributes that are used in cell subrequests.
CellSubResponseDataOptionalAttributes	Contains XML schema attributes that are used in cell subresponses.
CoauthSubRequestDataOptionalAttributes	Contains XML schema attributes that are used in

	coauthoring subrequests.
ExclusiveLockSubRequestDataOptionalAttributes	Contains XML schema attributes that are used in exclusive lock subrequests.
SchemaLockSubRequestDataOptionalAttributes	Contains XML schema attributes that are used in schema lock subrequests.
WhoAmISubResponseDataOptionalAttributes	Contains XML schema attributes that are used in Who Am I subresponses.

2.3.3.1 CellSubRequestDataOptionalAttributes

The **CellSubRequestDataOptionalAttributes** attribute group contains attributes that MUST be used only for **SubRequestData** elements associated with the parent **SubRequest** element where the **Type** attribute has a value of **CellSubRequestType**. The attributes in **CellSubRequestDataOptionalAttributes** are used as input parameters for processing the data associated with a cell subrequest. The definition of the **CellSubRequestDataOptionalAttributes** attribute group is as follows:

```
<xs:attributeGroup name="CellSubRequestDataOptionalAttributes">
    <xs:attribute name="Coalesce" type="xs:boolean" use="optional" />
    <xs:attribute name="GetFileProps" type="xs:boolean" use="optional" />
    <xs:attribute name="CoauthVersioning" type="xs:boolean" use="optional" />
    <xs:attribute name="Etag" type="xs:string" use="optional" />
    <xs:attribute name="ContentChangeUnit" type="xs:string" use="optional" />
    <xs:attribute name="ClientFileID" type="xs:string" use="optional" />
    <xs:attribute name="PartitionID" type="tns:guid" use="optional" />
    <xs:attribute name="ExpectNoFileExists" type="xs:boolean" use="optional" />
    <xs:attribute name="BypassLockID" type="xs:string" use="optional" />
    <xs:attribute name="LastModifiedTime" type="xs:integer" use="optional"/>
</xs:attributeGroup>
```

Coalesce: A Boolean value that specifies whether the protocol server SHOULD<21> fully save all changes to the underlying store without storing the changes in any intermediate write cache after processing the subrequest. If the **Coalesce** attribute is set to **true** in a cell subrequest, the protocol server persists all changes to the file after processing the subrequest. The **Coalesce** attribute is set to **false** in the cell subrequest for updates to other partitions that do not contain binary file contents.

GetFileProps: A Boolean value that specifies whether the file properties have been requested. When set to **true**, file properties have been requested as part of the cell subrequest. When set to **false**, file properties have not been requested as part of the cell subrequest. When set to **true**, the protocol server MUST return **CreateTime** and **LastModifiedTime** as attributes in the cell **SubResponseData** element.<22>

CoauthVersioning: A Boolean value. If the coauthoring status of a client is "Coauthoring", this value MUST be **true** for upload requests, which helps prevent the protocol server from doing multiple version updates in a short period of time. Otherwise, it MUST be **false**. For more details about the coauthoring status, see section 2.2.5.1.

Etag: A unique string value that gets updated every time the file contents are changed. The unique string gets updated irrespective of which protocol client updated the file contents in a coauthorable file. Any time the protocol client specifies the **Etag** attribute in a cell subrequest, the server MUST check to ensure that the **Etag** sent by the client matches the **Etag** specified for that file on the server. If there is a mismatch, the protocol server MUST send an error code value set to "CellRequestFail" in the cell subresponse message. The protocol server processes this value as specified in [RFC2616].

ContentChangeUnit: A string value that uniquely identifies the synchronization version of the file contents. It is the value of the **vti_contentchangeunit** property, as specified in [MS-LISTSWS].

ClientFileID: A string value that uniquely identifies the file contents on the protocol client and that is stored by the protocol server. When a protocol client sends requests for the upload of these file contents, the protocol server uses the unique **ClientFileID** to identify the specific file content to which the cell subrequest applies. It is the value of the **vti_clientid** property, as specified in [MS-LISTSWS].

PartitionID: A **guid** that specifies a unique identifier for the **Partition-block** in which the file contents or file metadata contents has been requested to be updated. The **guid** type is defined in section 2.2.5.7. The **Partition-block** is defined in section 3.1.1. A protocol client using a value of 2.2 or greater for the **VersionNumberType**, as specified in section 2.2.4.7, MUST specify the **Target Partition Id** attribute in a cell subrequest, as specified in [MS-FSSHTTPB] section 2.2.2.1.1, and MUST NOT include this attribute if communicating with a protocol server that has a version number of 2.2 or greater. The protocol client MUST specify the **PartitionID** attribute if communicating with a protocol server that has a versions number of 2.0. A protocol server that has a version number of 2.2 MUST accept a **PartitionID** attribute from clients using protocol numbers of 2.0 and 2.2. The value of **PartitionID** for the file contents is "00000000-0000-0000-0000-000000000000".

The value of **PartitionID** MUST be one of the values in the following table.

Value	Description
"00000000-0000-0000-0000-000000000000"	Default. The partition identifier for the file contents.
"383adc0b-e66e-4438-95e6-e39ef9720122"	The partition identifier for metadata.
"7808f4dd-2385-49d6-b7ce-37aca5e43602"	The partition identifier for editors table.

ExpectNoFileExists: A Boolean value that specifies whether the protocol server can expect that no file contents can be found when an empty **Etag** is sent by a client during an upload of file content.

BypassLockID: A unique string value. If a client has got an exclusive lock, this value MUST be the same as the value of **ExclusiveLockID**, as specified in section 2.3.1.1. If a client has got a shared lock, this value MUST be the same as the value of **SchemaLockID**, as specified in section 2.3.1.1. The value of **LockID**, as specified in [MS-FSSHTTPB] section 2.2.2.1.4.2, is the same as this value if the value of **VersionNumberType**, as specified in section 2.2.4.7, is 2.2 or greater. A protocol client that has a version number of 2.2 MUST specify **LockID** by using the cell subrequest parameter and MUST NOT include this attribute if communicating with a protocol server that has a version number of 2.2 or greater. The protocol client MUST specify the **BypassLockID** attribute if communicating with a protocol server that has a version number of 2.0. A protocol server that has a version number of 2.2 MUST accept both **LockID** and **BypassLockID**.

LastModifiedTime: An integer that specifies the last modified time, which is expressed as a tick count. A single tick represents 100 nanoseconds, or one ten-millionth of a second. **LastModifiedTime** 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). The protocol server MUST save the file with this value as the LastModifiedTime instead of the current time.

2.3.3.2 CellSubResponseDataOptionalAttributes

The **CellSubResponseDataOptionalAttributes** attribute group contains attributes that MUST be used only in **SubResponseData** elements associated with a **SubResponse** for a cell subrequest. The attributes in **CellSubResponseDataOptionalAttributes** provide the data that was requested as part of the cell subrequest. The definition of the **CellSubResponseDataOptionalAttributes** attribute group is as follows:

```
<xs:attributeGroup name="CellSubResponseDataOptionalAttributes">
    <xs:attribute name="Etag" type="xs:string" use="optional" />
    <xs:attribute name="CreateTime" type="xs:integer" use="optional"/>
```

```
    <xs:attribute name="LastModifiedTime" type="xs:integer" use="optional"/>
    <xs:attribute name="ModifiedBy" type="tns:UserNameType" use="optional" />
    <xs:attribute name="CoalesceErrorMessage" type="xs:string" use="optional"/>
    <xs:attribute name="CoalesceHResult" type="xs:integer" use="optional"/>
    <xs:attribute name="ContainsHotboxData" type="tns:TRUEFALSE" use="optional"/>
    <xs:attribute name="HaveOnlyDemotionChanges" type="tns:TRUEFALSE" use="optional"/>
</xs:attributeGroup>
```

Etag: A unique string value that is updated every time the file contents are changed. The unique string gets updated irrespective of which protocol client updated the file contents in a coauthorable file. **Etag** defines the file version and allows for the client to know the version of the file.

When the **Etag** attribute is specified as part of a response to a cell subrequest, the **Etag** attribute value specifies the updated file version. The **Etag** attribute value for that file MUST be used by the client in the next cell subrequest that it sends for that file. The **Etag** attribute value is an opaque string value to the protocol server. The protocol server processes this value as specified in [RFC2616] section 14.19.

CreateTime: An integer that specifies the time, which is expressed as a tick count, at which the file was created. A single tick represents 100 nanoseconds, or one ten-millionth of a second. **CreateTime** 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). The protocol server MUST return and specify the **CreateTime** attribute in the cell **SubResponseData** element only when the **GetFileProps** attribute is set to **true** in the cell subrequest.

LastModifiedTime: An integer that specifies the last modified time, which is expressed as a tick count. A single tick represents 100 nanoseconds, or one ten-millionth of a second. **LastModifiedTime** 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). The protocol server MUST return and specify the **LastModifiedTime** attribute in the cell **SubResponseData** element only when the **GetFileProps** attribute is set to **true** in the cell subrequest.

ModifiedBy: A **UserNameType** that specifies the user name for the client that last modified the file. **UserNameType** is defined in section 2.3.2.6.

CoalesceErrorMessage: A string that specifies a description of the error that occurs when the protocol server fully saves all changes with the underlying file provider. **CoalesceErrorMessage** also specifies information about what was expected by the protocol server. **CoalesceErrorMessage** MUST be sent only when the **CoalesceHResult** attribute is set to an integer value which is not equal to 0.<23>

CoalesceHResult: An integer that MUST be 0 except when the protocol server attempts to fully save all the changes in the underlying store. It specifies the **HRESULT** when the protocol server attempts to fully save all the changes in the underlying store. **CoalesceHResult** MUST be set to a value ranging from -2,147,483,648 through 2,147,483,647. A **CoalesceHResult** value of 0 indicates success. If **CoalesceHResult** is not equal to 0, it indicates an exception or failure condition that occurred.<24>

ContainsHotboxData: A TRUEFALSE value that specifies whether the binary contents sent as part of the cell subresponse contains data that is not fully persisted to the underlying store. If **ContainsHotboxData** is set to **false**, all binary contents sent as part of the cell subresponse contains data that is persisted to the underlying store. If **ContainsHotboxData** is set to **true**, the binary contents sent as part of the cell subresponse are not persisted to the underlying store.

HaveOnlyDemotionChanges: A Boolean value that specifies whether the returned binary content has only protocol server property demotion changes compared to the client's last synced status. Property demotion changes are repeatable, automated changes made by a protocol server following a

protocol client's **Put Changes** request as specified in [MS-FSSHTTPB] section 2.2.2.1.4. If the value is set to **False**, the returned binary content contains changes other than property demotion changes.

2.3.3.3 CoauthSubRequestDataOptionalAttributes

The **CoauthSubRequestDataOptionalAttributes** attribute group contains attributes that MUST be used only for **SubRequestData** elements associated with the parent **SubRequest** element for a coauthoring subrequest. The attributes in **CoauthSubRequestDataOptionalAttributes** are used as input parameters for processing the data associated with a coauthoring subrequest. The definition of the **CoauthSubRequestDataOptionalAttributes** attribute group is as follows:

```
<xs:attributeGroup name="CoauthSubRequestDataOptionalAttributes">
    <xs:attribute name="CoauthRequestType" type="tns:CoauthRequestTypes" use="required"/>
</xs:attributeGroup>
```

CoauthRequestType: A **CoauthRequestTypes** that specifies the type of coauthoring subrequest. **CoauthRequestTypes** is defined in section 2.3.2.2.

Depending on the type of the coauthoring subrequest, the following table shows a mapping between the type of coauthoring subrequest and the attributes that MUST be specified for that **CoauthRequestType**.

In the following table, **Timeout**, **AllowFallbackToExclusive**, and **ExclusiveLockID** release the lock on conversion failure, and **ClientID**, and **SchemaLockID** are attributes that MUST be specified for the **SubRequestData** element associated with the coauthoring subrequest, depending on the type of coauthoring subrequest.

In the following table, a check mark signifies that the attribute MUST be specified as part of the **SubRequestData** element associated with the coauthoring subrequest.

Value of CoauthRequestType	Timeout	AllowFallbackToExclusive	ExclusiveLockID	Release lock on conversion failure	ClientID	SchemaLockID
"JoinCoauthoring" (Join Coauthoring)	✓		If **AllowFallbackToExclusive** is set to **true**, the attribute for the exclusive lock identifier MUST be specified.		✓	✓
"ExitCoauthoring" (Exit Coauthoring)					✓	✓
"RefreshCoauthoring" (Refresh Coauthoring)	✓				✓	✓
"ConvertToExclusive" (Convert To	✓		✓	✓	✓	✓

Exclusive Lock Coauthoring)						
"CheckLockAvailability" (Check Lock Availability)						✓
"MarkTransitionComplete" (Mark Coauthoring Transition Complete)					✓	✓
"GetCoauthoringStatus" (Get Coauthoring status)					✓	✓

2.3.3.4 ExclusiveLockSubRequestDataOptionalAttributes

The **ExclusiveLockSubRequestDataOptionalAttributes** attribute group contains attributes that MUST be used only for **SubRequestData** elements associated with the parent **SubRequest** element for an exclusive lock subrequest. The attributes in **ExclusiveLockSubRequestDataOptionalAttributes** are used as input parameters for processing the data associated with an exclusive lock subrequest. The definition of the **ExclusiveLockSubRequestDataOptionalAttributes** attribute group is as follows:

```
<xs:attributeGroup name="ExclusiveLockSubRequestDataOptionalAttributes">
   <xs:attribute name="ExclusiveLockRequestType" type="tns:ExclusiveLockRequestTypes"
use="required"/>
</xs:attributeGroup>
```

ExclusiveLockRequestType: An **ExclusiveLockRequestTypes** that specifies the type of exclusive lock subrequest. **ExclusiveLockRequestTypes** is defined in section 2.3.2.3.

The following table shows a mapping between the type of exclusive lock subrequest and the attributes that MUST be specified for that **ExclusiveLockRequestType**.

In the following table, **Timeout**, **SchemaLockID**, **ClientID**, and **ExclusiveLockID** are attributes that MUST be specified for the **SubRequestData** element associated with the exclusive lock subrequest, depending on the type of exclusive lock subrequest.

In the following table, a check mark signifies that the attribute MUST be specified as part of the **SubRequestData** element associated with the exclusive lock subrequest.

Value of Exclusive LockRequestType	Timeout	SchemaLockID	ClientID	Exclusivelock ID
"GetLock" (Get Lock)	✓			✓
"ReleaseLock" (Release Lock)				✓
"RefreshLock" (Refresh Lock)	✓			✓
"ConvertToSchemaJoinCoauth" (Convert To Schema Lock With Coauthoring Transition Tracked)	✓	✓	✓	✓
"ConvertToSchema" (Convert To Schema Lock)	✓	✓	✓	✓
"CheckLockAvailability" (Check Lock Availability)				✓

2.3.3.5 SchemaLockSubRequestDataOptionalAttributes

The **SchemaLockSubRequestDataOptionalAttributes** attribute group contains attributes that MUST be used only for **SubRequestData** elements associated with parent **SubRequest** element for a schema lock subrequest. The attributes in **SchemaLockSubRequestDataOptionalAttributes** are used as input parameters for processing the data associated with a schema lock subrequest. The definition of the **SchemaLockSubRequestDataOptionalAttributes** attribute group is as follows:

```
<xs:attributeGroup name="SchemaLockSubRequestDataOptionalAttributes">
    <xs:attribute name="SchemaLockRequestType" type="tns:SchemaLockRequestTypes"
use="optional"/>
</xs:attributeGroup>
```

SchemaLockRequestType: A **SchemaLockRequestTypes** that specifies the type of schema lock subrequest. **SchemaLockRequestTypes** is defined in section 2.3.2.4.

The following table shows a mapping between the type of schema lock subrequest and the attributes that MUST be specified for that **SchemaLockRequestType**.

In the following table, **Timeout**, **AllowFallbackToExclusive**, and **ExclusiveLockID** release the lock on conversion failure, and **ClientID**, and **SchemaLockID** are attributes that MUST be specified for the **SubRequestData** element associated with the coauthoring subrequest, depending on the type of coauthoring subrequest.

In the following table, a check mark signifies that the attribute MUST be specified as part of the **SubRequestData** element associated with the schema lock subrequest.

Value of SchemaLockRequestType	Timeout	AllowFallbackToExclusive	ExclusiveLockID	Release lock on conversion failure	ClientID	SchemalockID
"GetLock" (Get Lock)	✓		If **AllowFallbackToExclusive** is set to **true**, the attribute for the exclusive lock identifier MUST be specified.		✓	✓
"ReleaseLock" (Release Lock)					✓	✓
"RefreshLock" (Refresh Lock)	✓				✓	✓
"ConvertToExclusive" (Convert To Exclusive Lock)	✓		✓	✓	✓	✓
"CheckLockAvailability" (Check Lock Availability)						✓

2.3.3.6 WhoAmISubResponseDataOptionalAttributes

The **WhoAmISubResponseDataOptionalAttributes** attribute group contains attributes that MUST be used only in **SubResponseData** elements associated with a subresponse for a Who Am I subrequest. The attributes in **WhoAmISubResponseDataOptionalAttributes** provide the data that was requested as part of the Who Am I subrequest. The schema definition of the **WhoAmISubResponseDataOptionalAttributes** attribute group is as follows:

```
<xs:attributeGroup name="WhoAmISubResponseDataOptionalAttributes">
    <xs:attribute name="UserName" type="tns:UserNameType" use="optional"/>
    <xs:attribute name="UserEmailAddress" type="xs:string" use="optional"/>
    <xs:attribute name="UserSIPAddress" type="xs:string" use="optional" />
    <xs:attribute name="UserIsAnonymous" type="xs:boolean" use="optional" />
    <xs:attribute name="UserLogin" type="xs:UserLoginType" use="required" />
</xs:attributeGroup>
```

UserName: A **UserNameType** that specifies the user name for the client. **UserNameType** is defined in section 2.3.2.6.

UserEmailAddress: 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.

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

UserIsAnonymous: A Boolean value that specifies whether the client is an anonymous guest user. Protocol servers SHOULD return TRUE if the user is an anonymous guest user.<25>

UserLogin: A **UserLoginType** that specifies the user login alias of the client. **UserLoginType** is defined in section 2.3.2.5. The **UserLogin** attribute MUST be specified in a **WhoAmI** subresponse that is generated in response to a **WhoAmI** subrequest.

2.3.3.7 EditorsTableSubRequestDataOptionalAttributes

The **EditorsTableSubRequestDataOptionalAttributes** attribute group contains attributes that MUST be used only for **SubRequestData** elements associated with the parent **SubRequest** element for an editors table subrequest. The attributes in **EditorsTableSubRequestDataOptionalAttributes** are used as input parameters for processing the data associated with an editors table subrequest. The definition of the **EditorsTableSubRequestDataOptionalAttributes** attribute group is as follows:

```
<xs:attributeGroup name="EditorsTableSubRequestDataOptionalAttributes">
    <xs:attribute name="EditorsTableRequestType" type="tns:EditorsTableRequestTypes"
use="optional"/>
</xs:attributeGroup>
```

EditorsTableRequestType: An **EditorsTableRequestType** that specifies the type of editors table subrequest. **EditorsTableRequestTypes** is defined in section 2.3.2.5.

Depending on the type of the editors table subrequest, the following table shows a mapping between the type of editors table subrequest and the attributes that MUST be specified for that **EditorsTableRequestType**.

In the following table, **Timeout**, **ClientID**, **AsEditor**, **Key**, and **Value** are attributes that MUST be specified for the **SubRequestData** element associated with the editors table subrequest, depending on the type of editors table subrequest.

In the following table, a check mark signifies that the attribute MUST be specified as part of the **SubRequestData** element associated with the editors table subrequest.

Value of SchemaLockRequestType	Timeout	ClientID	AsEditor	Key	Value
"JoinEditingSession" (Join Editing Session)	✓	✓	✓		
"LeaveEditingSession" (Leave Editing Session)		✓			
"RefreshEditingSession" (Refresh Editing Session)	✓	✓	✓		
"UpdateEditorMetadata" (Update Editor Metadata)		✓		✓	✓
"RemoveEditorMetadata" (Remove Editor Metadata)		✓		✓	

3 Protocol Details

The client side of this protocol is simply a pass-through. That is, no additional timers or other state is required on the client side of this protocol. Calls made by the higher-layer protocol or application are passed directly to the transport, and the results returned by the transport are passed directly back to the higher-layer protocol or application.

Except where specified, protocol clients MUST interpret HTTP status codes returned by the protocol server as specified in [RFC2616] section 10.

This protocol allows protocol servers to notify protocol clients of request-specific or subrequest-specific error codes as part of the message itself and not via a SOAP fault message.

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.

The following states specify the data organization that a protocol server maintains to participate in the protocol:

- **Partition-block:** A partition that contains binary file data in the abstract data model as specified in [MS-FSSHTTPB]. A file is associated with one or more partitions. The server stores the binary file contents and file format–specific data in partitions. File contents are stored in the default partition. Each partition is uniquely identified by a partition identifier.
- **File coauthoring tracker:** A tracking mechanism that the protocol server uses for coauthoring. For each file, the protocol server keeps track of the client identifier associated with each client that is coauthoring the file, the type of lock on the file—that is, an exclusive lock or a shared lock—and the timeout associated with each client's shared lock on the coauthorable file.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Message Processing Events and Sequencing Rules

The following table summarizes the list of operations as defined by this specification.

Operation	Description
Cell subrequest	Retrieves or uploads a file's binary contents or a file's metadata contents.
Coauth subrequest	Gets a shared lock on a coauthorable file that allows for all clients with the same schema lock identifier to share the lock. The protocol server also keeps tracks

	of the clients sharing the lock on a file at any instant of time.
SchemaLock subrequest	Gets a shared lock on a coauthorable file that allows all clients with the same schema lock identifier to share the lock.
ExclusiveLock subrequest	Gets an exclusive lock on the file, which ensures only one client edits the file at an instant in time.
WhoAmI subrequest	Retrieves the client's **friendly name** and other client-specific information for a client with a unique client identifier.
ServerTime subrequest	Retrieves the server time.
Editors Table subrequest	Adds the client to the editors table, which is accessible to all clients editing or reading a document.
GetDocMetaInfo subrequest	Retrieves various properties for the file and the parent folder as a series of string pairs.
GetVersions subrequest	Sends back information about the previous versions of a file.

3.1.4.1 Common Message Processing Rules and Events

The protocol server MUST follow the following common processing rules for all types of subrequests:

- The **Url** attribute in the cell storage service **Request** element specifies the unique URL for the file that the request is to be processed for. The **Url** attribute is specified in section 2.2.3.2.
- The **SubRequestToken** attribute in the cell storage service **SubRequest** element uniquely identifies a subrequest for a file. The **SubRequestToken** attribute is specified in section 2.2.4.3.
- The **Type** attribute in the cell storage service **SubRequest** element specifies the type of cell storage service subrequest. The protocol server uses the **Type** attribute to identify the type of cell storage service subrequest. The **Type** attribute is specified in section 2.2.4.2.
- The **DependencyType** attribute in the cell storage service **SubRequest** element specifies the type of dependency that a subrequest has on another subrequest. **DependencyType** attribute for a **SubRequest** element is specified in section 2.2.4.3. The protocol server identifies the dependency type to check if the subrequest is to be processed.
- The **DependsOn** attribute specifies the **SubRequestToken** of the subrequest that this subrequest depends on. The **DependsOn** attribute for a **SubRequest** element is specified in section 2.2.4.3.

The protocol server uses a combination of the **DependsOn** and **DependencyType** attributes for a specific subrequest to decide if a cell storage service subrequest will be executed. The protocol server sends a **Response** element for each **Request** element and a **SubResponse** element corresponding to each **SubRequest** element contained within a **Request** element.

If the protocol server supports shared locking, the protocol server MUST support at least one of the following subrequests:

- The coauthoring subrequest
- The schema lock subrequest

If the protocol server supports the coauthoring subrequest, it MUST support tracking the coauthoring transition. The coauthoring transition allows for the number of users editing the file to increase from 1 to *n* or to decrease from *n* to 1, where *n* is the maximum number of users who are allowed to edit a

single file at an instant in time. If the protocol server supports the coauthoring subrequest, it MUST return a coauthoring status as specified in section 2.3.1.7.

A shared lock on a file is granted by sending one of the following subrequests to the protocol server:

- A coauthoring subrequest
- A schema lock subrequest
- An exclusive lock subrequest of type "Convert to schema lock"
- An exclusive lock subrequest of type "Convert to schema lock with coauthoring transition tracked"

An exclusive lock on a file is granted by sending one of the following subrequests to the protocol server:

- An exclusive lock subrequest
- A coauthoring subrequest of type "Convert to exclusive lock"
- A schema lock subrequest of type "Convert to exclusive lock"

The protocol server does the following to decide when to release the lock on the file: When the timeout expires for a client, and no refresh on the timeout was received, the protocol server MUST release that client's lock on the shared file. When the timeouts for all clients holding a shared lock on the file expire, the shared lock on the file MUST<26> be released by the protocol server. To prevent the lock on the file from expiring, the protocol client MUST send a request to refresh the timeout at a regular interval that is shorter than the lock timeout.

The protocol server is used by clients to synchronize both coauthorable files and files that are not coauthorable. The protocol server is also used by clients to synchronize file's metadata contents.

The protocol server MAY return an error code value set to "RequestNotSupported" for a cell storage service subrequest if the following conditions are all true:

- The protocol client sent a coauthoring subrequest.
- The protocol server supports shared locking with tracking of the coauthoring transition.
- The coauthoring administrator setting for the server is turned off.

3.1.4.2 Cell Subrequest

This operation is used to retrieve or upload the binary or metadata contents of a file. The contents are uploaded or downloaded fully or partially. The download or upload of the contents is accomplished by indicating the partition identifier whose contents are to be uploaded or downloaded. The cell subrequest acts as a wrapper around the actual file synchronization processing logic as specified in [MS-FSSHTTPB].

The protocol client sends a cell **SubRequest** message, which is of type **CellSubRequestType** as specified in section 2.3.1.2. The protocol server responds with a cell **SubResponse** message, which is of type **CellSubResponseType** as specified in section 2.3.1.4. This is done as follows:

- The protocol client prepares a request containing a URL for the file, a unique **Request** token, and one or more **SubRequest** elements, as defined in section 2.2.3.2 and section 2.2.3.8. The **SubRequest** element is of type "Cell", and the **SubRequestData** element contains attributes that are input parameters used by the server when processing the cell subrequest. The **SubRequestData** element is of type **CellSubRequestDataType** and is defined in section 2.3.1.1.
- The binary data sent in the **SubRequestData** element indicates if this is a file content upload or download. The meaning of the data is as specified in [MS-FSSHTTPB] section 2.2.3.1.3. The cell subrequest for an upload of the file contents is processed as specified in [MS-FSSHTTPB] section 3.1.4.3. The cell subrequest for a download of the file contents is processed as specified in [MS-

FSSHTTPB] section 3.1.4.2. The schema lock identifier MUST be specified in a cell subrequest if the current client already shared the lock on the file and is in a coauthoring session. The schema lock identifier is defined in section 2.3.1.1. When the schema lock identifier is specified in the cell subrequest, the protocol server returns a success error code if the file currently has a shared lock with the specified schema lock identifier.

- The bypass lock identifier MUST be present when uploading the binary or metadata contents of a file. It MUST NOT be present when retrieving the binary or metadata contents of a file. The bypass lock identifier is defined in section 2.3.3.1. When the bypass lock identifier is specified in a cell subrequest, exactly one of the following is true:
  - The file is schema locked. In this case, the bypass lock identifier MUST be the same as the schema lock identifier. They have the same semantics, in this case.
  - The file is exclusive locked. In this case, the bypass lock identifier MUST be the same as the exclusive lock identifier. They have the same semantics, in this case. The exclusive lock identifier is defined in section 2.3.3.1.
- If the **ExpectNoFileExists** attribute is set to **true** in a file content upload cell subrequest, the **Etag** attribute MUST be an empty string. In this case, the protocol server MUST cause the cell subrequest to fail with a coherency error if and only if the file already exists on the server. The **ExpectNoFileExists** attribute is defined in section 2.3.3.1. The **Etag** attribute is defined in section 2.3.3.1. Coherency failure error codes are as specified in [MS-FSSHTTPB].
- If the **Coalesce** attribute is set to **true** in a cell subrequest, the protocol server persists all changes to the file in the underlying store. The **Coalesce** attribute is defined in section 2.3.3.1. For all first-time saves of the file's binary file contents in a partition, the following MUST be specified:
  - The **ExclusiveLockID** attribute (section 2.3.1.1) is set to a unique lock identifier.
- When the protocol server receives the **ExclusiveLockID** in a cell subrequest, the server performs the following steps as an atomic operation:
  1. Get an exclusive lock on the file.
  2. Commit the file's binary contents, as specified in [MS-FSSHTTPB] section 3.1.4.3. (Note that for updates to other partitions that do not contain binary file contents, the **Coalesce** attribute is set to **false** in the cell subrequest.)
- The protocol server receives the request and parses the logic to send the information to the underlying store. If the request is a download request, the encoded file contents are sent back to the client. If file properties–specific information is requested, the file properties are prepared as a response and sent back to the protocol client.
  - The **Response** element is defined in section 2.2.3.5, and the **SubResponse** element is defined in section 2.2.3.10. The **CellSubResponseDataType** specifies the type of the **SubResponseData** element inside the cell **SubResponse** element. The **CellSubResponseDataType** is defined in section 2.3.1.3. In the case where the protocol server finishes performing the request but in the middle of generating the **SubResponseData** data, another request changes the file content or metadata, the protocol server returns an additional **SubResponseStreamInvalid** element to indicate that the binary data in the **SubResponseData** is not valid. The protocol client can resend the request to get the data. The **SubResponseStreamInvalid** element is defined in 2.3.1.4.
- The protocol server returns results based on the following conditions:
  - If the protocol server was unable to find the URL for the file specified in the **Url** attribute, the protocol server reports a failure by returning an error code value set to "CellRequestFail" in the **ErrorCode** attribute sent back in the **SubResponse** element, and the binary data in the

returned **SubRequestData** element indicates an **HRESULT Error** as described in [MS-FSSHTTPB] section 2.2.3.2.4. But for **Put Changes** subrequest, as described in [MS-FSSHTTPB] section 2.2.2.1.4, the protocol server creates a new file using the specified **Url**.

- Depending on the type of error, the **ErrorCode** returned as an attribute of the **SubResponse** element is updated with a specific error code as specified in section 2.2.5.4.
- **ErrorCode** includes "Success" to indicate success in processing the file upload or download request.

3.1.4.3 Coauth Subrequest

This operation is used to make a request to the protocol server for a shared lock on the file or for shared lock operations on a file. A protocol server that supports a coauthoring subrequest MUST also support tracking the coauthoring transition. Depending on the **CoauthRequestType** attribute value, the protocol server interprets the subrequest as one of the following types of shared lock operations:

- Join coauthoring session
- Exit coauthoring session
- Refresh coauthoring session
- Convert to exclusive lock
- Check lock availability
- Mark transition to complete
- Get coauthoring status

The **CoauthRequestType** attribute is defined in section 2.3.3.3. The **CoauthRequestType** attribute is one of the attributes for the coauthoring **SubRequestData** element, which is of type **CoauthSubRequestDataType** as specified in section 2.3.1.5.

The protocol client sends a coauthoring **SubRequest** message, which is of type **CoauthSubRequestType** as specified in section 2.3.1.6. The protocol server responds with a coauthoring **SubResponse** message, which is of type **CoauthSubResponseType** as specified in section 2.3.1.8. This is done as follows:

- The protocol client prepares a request containing a URL for the file, a unique **Request** token, and one or more **SubRequest** elements, as defined in section 2.2.3.2 and section 2.2.3.8. The **SubRequest** element is of type "Coauth", and the **SubRequestData** element contains attributes that are input parameters used by the server when processing the coauthoring subrequest. The **SubRequestData** element is of type **CoauthSubRequestDataType** and is defined in section 2.3.1.5.
- The protocol server receives the request and parses the logic. Depending on the type of coauthoring request, the protocol server processes the subrequest as specified in section 3.1.4.3.1, 3.1.4.3.2, 3.1.4.3.3, 3.1.4.3.4, 3.1.4.3.5, 3.1.4.3.6, or 3.1.4.3.7. The protocol server uses the **ClientID** attribute sent in a coauthoring subrequest to do the following:
  - Uniquely identify each client
  - Keep track of each client and its timeout on the shared lock for the file
  - Decide when to release the shared lock on the file

- The **Response** element is defined in section 2.2.3.5, and the **SubResponse** element is defined in section 2.2.3.10. **CoauthSubResponseDataType** defines the type of the **SubResponseData** element inside the coauthoring **SubResponse** element. **CoauthSubResponseDataType** is defined in section 2.3.1.7.

- If the coauthoring subrequest is of type "Join coauthoring session" or "Refresh coauthoring session", the protocol server MUST return the lock type granted to the client as part of the response message to the client—if the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success". The lock type is specified in the **LockType** attribute in the coauthoring **SubResponseData** element. The **SubResponseData** element returned for a coauthoring subrequest is of type **CoauthSubResponseDataType** and is defined in section 2.3.1.7. The **LockType** attribute is specified in section 2.3.1.7.

- The protocol server returns results based on the following conditions:

  - Depending on the type of error, **ErrorCode** is returned as an attribute of the **SubResponse** element. The **ErrorCode** attribute that is part of the **SubResponse** element is updated with a specific error code as specified in section 2.2.5.4.

  - If the protocol server was unable to find the URL for the file specified in the **Url** attribute, the protocol server reports a failure by returning an error code value set to "FileNotExistsOrCannotBeCreated" in the **ErrorCode** attribute sent back in the **SubResponse** element.

  - The protocol server returns an error code value set to "CoauthRefblobConcurrencyViolation" when there is a concurrency violation and the coauthoring subrequest is one of the following types:

    - Join coauthoring session

    - Exit coauthoring session

    - Refresh coauthoring session

    - Convert to an exclusive lock

    - Mark in transition complete

  - A concurrency violation happens when a current client's request to save the file coauthoring tracker fails because another client's request to edit and save the file coauthoring tracker is in progress on the server before the save is done by the current client. "CoauthRefblobConcurrencyViolation" is specified in section 2.2.5.7. The file coauthoring tracker is defined in section 3.1.1.

  - The protocol server returns an error code value set to "RequestNotSupported" if the server does not support this request type.

  - An **ErrorCode** value of "Success" indicates success in processing the coauthoring request.

3.1.4.3.1 Join Coauthoring Session

If the **CoauthRequestType** attribute is set to "JoinCoauthoring", the protocol server considers the coauthoring subrequest to be of type "Join coauthoring session". The protocol server processes this request to get a shared lock on the coauthorable file and joins the coauthoring session of the file by adding the client's associated **ClientID** and timeout to the file coauthoring tracker. The protocol server also checks and gets the coauthoring status of the file.

If the file already has a shared lock on the server with the given schema lock identifier, and the client has already joined the coauthoring session, the protocol server does both of the following:

- Refreshes the timeout value associated with the **ClientID** in the file coauthoring tracker.
- Returns an error code value set to "Success".

If the coauthoring feature is disabled on the protocol server, it does one of the following:

- If the **AllowFallbackToExclusive** attribute is set to **true**, the protocol server gets an exclusive lock on the file.
- If the **AllowFallbackToExclusive** attribute is set to **false**, the protocol server returns an error code value set to "FileNotLockedOnServerAsCoauthDisabled".<27>

The **AllowFallbackToExclusive** attribute is defined in section 2.3.1.5. The result of the lock type gotten by the server MUST be sent as the **LockType** attribute in the **CoauthSubResponseDataType**. The **CoauthSubResponseDataType** is defined in section 2.3.1.7. **LockType** attribute values are defined in section 2.2.5.9.

To get the coauthoring status, the protocol server checks the number of clients editing the file at that instant in time. If the current client is the only client editing the file, the protocol server MUST return a **CoauthStatus** set to "Alone", which indicates that no one else is editing the file. If the current client is the second, third, or later coauthor joining the coauthoring session, the protocol server MUST return a **CoauthStatus** set to "Coauthoring", which indicates that the current client is coauthoring when editing the document. The **CoauthStatus** attribute sent as part of the **SubResponseData** element is defined in section 2.2.8.2.

If the current client is the second client to join the coauthoring session, the server creates a new transition request by using the **TransitionID** attribute for the file. **TransitionID** is defined in section 2.3.1.7. Since there is a transition request present for the file, when the first client that was added to the coauthoring session makes the **IsOnlyClient** web service request, the server returns "false". The descriptions of the transition request and the **IsOnlyClient** web service request are as specified in [MS-SHDACCWS]. This sequence of subrequests is described in the figure in section 3.1.4.3.6.

If there is a current exclusive lock on the file or a shared lock on the file with a different schema lock identifier, the protocol server returns an error code value set to "FileAlreadyLockedOnServer". If the coauthorable file is checked out on the server and checked out by a client with a different user name than the current client, the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer".

The protocol server returns an error code value set to "NumberOfCoauthorsReachedMax" when all of the following conditions are true:

- The maximum number of coauthorable clients allowed to join a coauthoring session to edit a coauthorable file has been reached.
- The current client is not allowed to edit the file because the limit has been reached.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8, and generic error code types are defined in section 2.2.5.6. For other unknown error types, the protocol server returns an error code value set to "LockRequestFail".

3.1.4.3.2 Exit Coauthoring Session

If the **CoauthRequestType** attribute is set to "ExitCoauthoring", the protocol server considers this coauthoring subrequest to be of type, "Exit coauthoring session". The protocol server processes the request to exit the coauthoring session and checks the number of clients editing the document at that instant in time. When the protocol server receives a coauthoring subrequest of type "Exit coauthoring

session" from the last and only client editing the document, the protocol server does both of the following:

- Delete the client identifier entry associated with the client in the file coauthoring tracker. The file coauthoring tracker is defined in section 3.1.1.
- Delete the coauthoring session and the shared lock on the file if the client that sent the subrequest of type "Exit coauthoring session" is the last client in the file coauthoring tracker.

If the current client is not already present in the coauthoring session, the protocol server does one of the following:

- Return an error code of "Success"<28>, if there are other clients present in the coauthoring session.

- Return an error code of "FileNotLockedOnServer" if no clients are present in the coauthoring session.

The protocol server SHOULD<29> return an error code of "FileAlreadyLockedOnServer" if there is a current exclusive lock on the file or a shared lock on the file with a different schema lock identifier.

If the coauthoring session has already been deleted, the protocol server returns an error code value set to "Success", as specified in section 2.2.5.5.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8, and generic error code types are defined in section 2.2.5.6. For other unknown error types, the protocol server returns an error code value set to "LockRequestFail".

3.1.4.3.3 Refresh Coauthoring Session

If the **CoauthRequestType** attribute is set to "RefreshCoauthoring", the protocol server considers this coauthoring subrequest to be of type, "Refresh coauthoring session". The protocol server refreshes the client's timeout of the shared lock on the coauthorable file and checks the coauthoring status of the file.

The protocol server refreshes the shared lock on the file. If the refresh of the shared lock on the file for that specific client fails because the file is no longer locked since the timeout value expired on the lock in the file coauthoring tracker, the protocol server does one of the following:

- If the coauthoring feature is enabled on the protocol server, the server considers this coauthoring subrequest to be of type, "Join coauthoring session", and it gets a new shared lock on the file.

- If the coauthoring feature is disabled, then the protocol server returns an error code value set to "FileNotLockedOnServerAsCoauthDisabled".<30>

The **Timeout** attribute is defined in section 2.3.1.5, and the file coauthoring tracker is defined in section 3.1.1.

After the protocol server has ensured that there is a shared lock with the same schema lock identifier used by the current client and that the client is sharing the shared lock on the file, the protocol server MUST update the client's timeout on the shared lock on the file. The new timeout value is the timeout value sent as part of the coauthoring subrequest. If the client is sending a request to refresh the shared lock with a timeout value less than or equal to the current timeout on the shared lock, the protocol server considers the coauthoring subrequest of type "Refresh coauthoring session" to be a no-operation instruction and does nothing. If the client is sending a request to refresh the shared lock with a timeout value greater than the current timeout on the shared lock, the protocol server updates the file coauthoring tracker with the new timeout value for that specific client and that specific file.

The **Timeout** attribute is defined in section 2.3.1.5, and the file coauthoring tracker is defined in section 3.1.1.

To get the coauthoring status, the protocol server checks the number of clients editing the file at that instant in time. If the current client is the only client editing the file, the protocol server MUST return a **CoauthStatus** set to "Alone", which indicates that no one else is editing the file. If the current client is the second, third, or later coauthor joining the coauthoring session, the protocol server MUST return a **CoauthStatus** set to "Coauthoring", which indicates that the current client is coauthoring when editing the document. The **CoauthStatus** attribute sent as part of the **SubResponseData** element is defined in section 2.2.8.2.

If there is a current exclusive lock on the file or a shared lock on the file with a different schema lock identifier, the protocol server returns an error code value set to "FileAlreadyLockedOnServer". If the coauthorable file is checked out on the server and checked out by a client with a different user name than the current client, the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer".

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8, and generic error code types are defined in section 2.2.5.6. For other unknown error types, the protocol server returns an error code value set to "LockRequestFail".

3.1.4.3.4 Convert to Exclusive Lock

If the **CoauthRequestType** attribute is set to "ConvertToExclusive", the protocol server considers this coauthoring subrequest to be of type, "Convert to Exclusive lock". The protocol server processes the request to convert a shared lock on the coauthorable file to an exclusive lock on the file.

The protocol server performs the following two operations:

- The conversion of the shared lock to an exclusive lock.
- The deletion of coauthoring session.

The protocol server returns an error code value set to "InvalidCoauthSession" to indicate a failure if any of the following conditions is true:

- There is no shared lock.
- There is no coauthoring session for the file.
- The current client is not present in the coauthoring session.

If there is a current exclusive lock on the file or if there is a shared lock on the file from another client with a different schema lock identifier, the request cannot be processed successfully so the protocol server returns "InvalidCoauthSession" as defined in section 2.2.5.8.

The shared lock is converted to an exclusive lock only if one client is currently editing the document. If the shared lock is successfully converted to an exclusive lock, the protocol server MUST use the unique value of the **ExclusiveLockID** attribute sent by the client to identify the lock. The **ExclusiveLockID** attribute is specified in section 2.3.1.5.

If there is more than one client currently editing the file and the **ReleaseLockOnConversionToExclusiveFailure** attribute is set to **false**, the protocol server returns an error code value set to "MultipleClientsInCoauthSession" to indicate the failure to convert to an exclusive lock. The protocol server returns an error code value set to "ExitCoauthSessionAsConvertToExclusiveFailed" when the following conditions are both **true**:

- The **ReleaseLockOnConversionToExclusiveFailure** attribute is set to **true** in the coauthoring subrequest.
- Multiple clients are in the coauthoring session.

When the **ReleaseLockOnConversionToExclusiveFailure** attribute is set to **true** and the conversion to an exclusive lock failed, the protocol server removes the client from the coauthoring session for the file, and it removes the current client's **ClientID** from the file coauthoring tracker that was associated with that file. The **ReleaseLockOnConversionToExclusiveFailure** attribute is specified in section 2.3.1.5.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8, and generic error code types are defined in section 2.2.5.6. For other unknown error types, the protocol server returns an error code value set to "LockRequestFail".

3.1.4.3.5 Check Lock Availability

If the **CoauthRequestType** attribute is set to "CheckLockAvailability", the protocol server considers this coauthoring subrequest to be of type, "Check lock availability". The protocol server checks if a file is available to take a shared lock or exclusive lock.

If there is a current exclusive lock on the file or if there is a shared lock on the file with a different schema lock identifier, the protocol server returns an error code value set to "FileAlreadyLockedOnServer". If the coauthorable file is checked out on the server and it is checked out by a client with a different user name than the current client, the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer". In all other cases, the protocol server returns an error code value set to "Success" to indicate the availability of the file for locking.

3.1.4.3.6 Mark Transition to Complete

If the **CoauthRequestType** attribute is set to "MarkTransitionComplete", the protocol server considers this coauthoring subrequest to be of type, "Mark transition to complete". The protocol server deletes the transition request for the file using the **TransitionID**. The description of when the transition request is created for the file is specified in section 3.1.4.3.1.

A coauthoring subrequest of type "Mark transition to complete" is requested by a protocol client that first joined a coauthoring session or by a protocol client that has a coauthoring status of "Alone". The request is sent by the protocol client when it receives "false" from the protocol server as a response to the **IsOnlyClient** web service request. The **IsOnlyClient** web service request is as specified in [MS-SHDACCWS].

The following diagram specifies a sequence in which a coauthoring subrequest of type "Mark transition to complete" is sent by Client1 (C1). C1 is the first client that joined the coauthoring session to edit the file. Client2 (C2) is the second client sharing the lock and joining the coauthoring session.

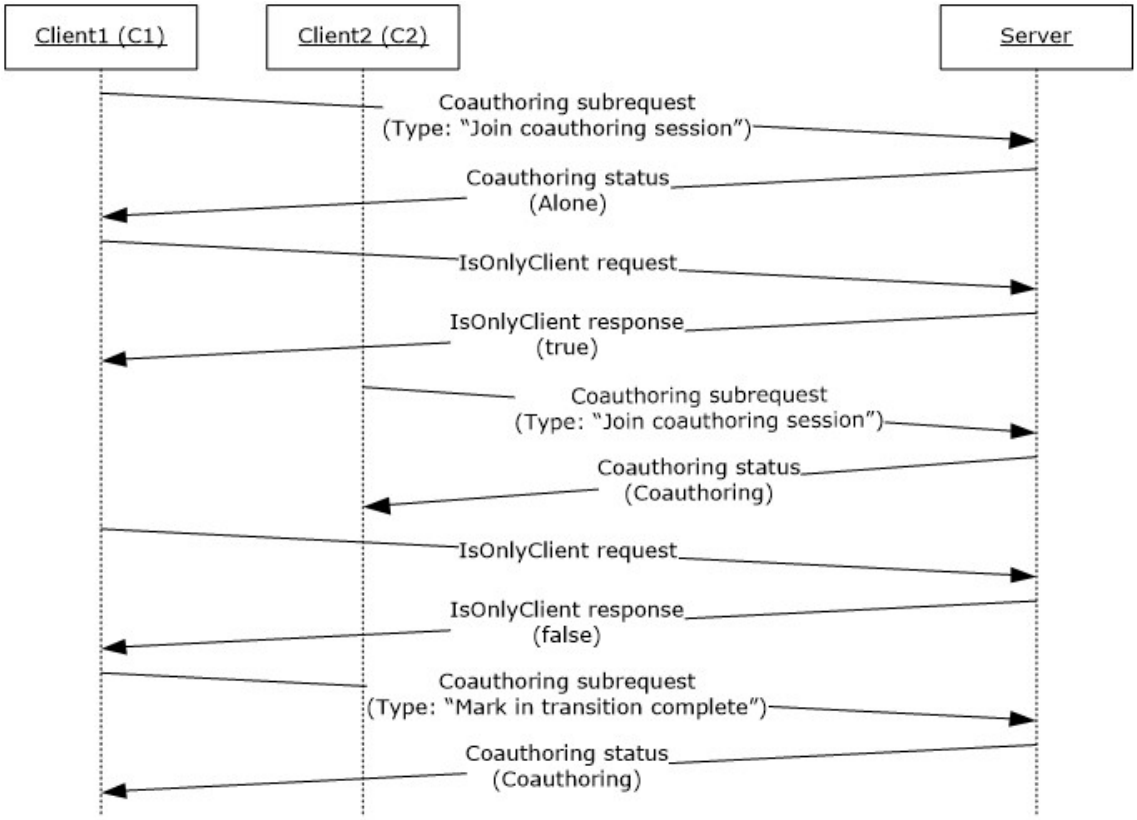

Figure 3: Sequence of coauthoring subrequest types

The coauthoring subrequest of type, "Join coauthoring session" is defined in section 3.1.4.3.1. The **IsOnlyClient** web service request sent by the client is as specified in [MS-SHDACCWS].

The **CoauthStatus** attribute is not set by the server in the subresponse returned for this subrequest. The client MUST set its local coauthoring status to "Coauthoring"—if the **ErrorCode** attribute in the subresponse is set to "Success" to indicate the successful processing of this subrequest.

The protocol server returns an error code value set to "InvalidCoauthSession" to indicate failure if any of the following conditions is true:

- There is no shared lock.<31>
- There is no coauthoring session for the file.<32>
- The current client is not present in the coauthoring session.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8, and generic error code types are defined in section 2.2.5.6. For other unknown error types, the protocol server returns an error code value set to "LockRequestFail".

3.1.4.3.7 Get Coauthoring Session

If the **CoauthRequestType** attribute is set to "GetCoauthoringStatus", the protocol server considers this coauthoring subrequest to be of type, "Get coauthoring status". The protocol server checks the coauthoring status of the coauthorable file. The protocol client keeps sending coauthoring subrequests of type, "Get coauthoring status" while it is actively coauthoring to check if it is the only client editing

the file. If the number of clients editing the file goes back to one, the protocol client stops sending the "Get coauthoring status" request and sends the **IsOnlyClient** web service request as specified in [MS-SHDACCWS].

If the coauthoring session does not exist or the current client is not present in the coauthoring session, the protocol server returns an error code value set to "InvalidCoauthSession"<33>. If the current client is the only client editing the coauthorable file, the protocol server MUST set the **CoauthStatus** attribute value to "Alone", indicating that no one else is editing the file. If the current client is the second, third, or later client trying to edit the document, the protocol server MUST return a **CoauthStatus** set to "Coauthoring", which indicates that the current client is coauthoring when editing the document.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8, and generic error code types are defined in section 2.2.5.6.

3.1.4.4 SchemaLock Subrequest

This operation is used to request a shared lock on a file or shared lock operations on a coauthorable file from the protocol server. The schema lock subrequest is conceptually a subset of the coauthoring subrequest and differs from the coauthoring subrequest by not keeping track of the clients currently in the coauthoring session that are sharing the lock on the file.

The term *schema lock* means that all clients with the same schema lock identifier share the lock and that clients with different schema lock identifiers are required to wait until the shared lock is released by all the clients having the same schema lock identifier. Depending on the **SchemaLockRequestType** attribute value, the protocol server interprets the request as one of the following types of lock operations:

- Get lock.
- Release lock.
- Refresh lock.
- Convert to exclusive lock.
- Check lock availability.

The **SchemaLockRequestType** attribute is defined in section 2.3.3.5. The **SubRequestData** element for the schema lock subrequest is of type **SchemaLockSubRequestDataType** and is defined in section 2.3.1.13.

The protocol client sends a schema lock **SubRequest** message, which is of type **SchemaLockSubRequestType** as specified in section 2.3.1.14. The protocol server responds with a schema lock **SubResponse** message, which is of type **SchemaLockSubResponseType** as specified in section 2.3.1.16. This is done as follows:

- The protocol client prepares a request containing a URL for the file, a unique **Request** token, and one or more **SubRequest** elements, as defined in section 2.2.3.2 and section 2.2.3.8. The **SubRequest** element is of type "SchemaLock" and the **SubRequestData** element contains attributes that are input parameters used by the protocol server when processing the schema lock subrequest. The **SubRequestData** element is of type, **SchemaLockSubRequestDataType** and is defined in section 2.3.1.13.

- The protocol server receives the request and parses the logic, and depending on the type of schema lock subrequest, the protocol sever process the request as specified in section 3.1.4.4.1, 3.1.4.4.2, 3.1.4.4.3, 3.1.4.4.4, or 3.1.4.4.5. The protocol server uses the **ClientID** attribute sent in a schema lock subrequest to uniquely identify each client and keep track of each client's timeout on the shared lock for the file. The protocol server also uses the **ClientID** sent in the schema lock subrequest to decide when to release the shared lock on the file. The protocol server does the following to decide when to release the shared lock on the file:

  - If the timeout expires for a client and no refresh on the timeout has been received, the protocol server MUST release that client's lock on the shared file. When the timeout for all the clients holding a shared lock on the file expire, the shared lock on the file is released by the protocol server.

- The **Response** element is as defined in section 2.2.3.5, and the **SubResponse** element is as defined in section 2.2.3.10. The **SchemaLockSubResponseDataType** defines the type of the **SubResponseData** element inside the schema lock **SubResponse** element. The **SchemaLockSubResponseDataType** is defined in section 2.3.1.15.

- If the schema lock subrequest is of type "Get lock" or "Refresh lock", the protocol server MUST return the lock type granted to the client as part of the response message to the client—if the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success". The lock type is sent as the **LockType** attribute in the schema lock **SubResponseData** element. The **SubResponseData** element returned for a schema lock subrequest is of type **SchemaLockSubResponseDataType** and is defined in section 2.3.1.15. The **LockType** attribute is specified in section 2.3.1.15. The **LockType** attribute values are specified in section 2.2.5.9.

- The protocol server returns results based on the following conditions:

  - Depending on the type of error, the **ErrorCode** value is returned as an attribute of the **SubResponse** element. The **ErrorCode** attribute that is part of the **SubResponse** element is updated with a specific error code as specified in section 2.2.5.4.

  - If the protocol server was unable to find the URL for the file specified in the **Url** attribute, the protocol server reports a failure by returning an error code value set to "FileNotExistsOrCannotBeCreated" in the **ErrorCode** attribute sent back in the **SubResponse** element.

  - The protocol server returns an error code value set to "CoauthRefblobConcurrencyViolation" when there is a concurrency violation and the schema lock subrequest is one of the following types:

    - Get lock

    - Release lock

    - Refresh lock

    - Convert to an exclusive lock

  - A concurrency violation happens when a current client's request to save the file coauthoring tracker fails because another client's request to edit and save the file coauthoring tracker is in progress on the server before the save is done by the current client. "CoauthRefblobConcurrencyViolation" is specified in section 2.2.5.7. The file coauthoring tracker is defined in section 3.1.1.

  - The protocol server returns an error code value set to "RequestNotSupported" if the server does not support this request type.

  - An **ErrorCode** value of "Success" indicates success in processing the schema lock request.

3.1.4.4.1 Get Lock

If the **SchemaLockRequestType** attribute is set to "GetLock", the protocol server considers this schema lock subrequest to be of type "Get lock". The protocol server processes the request to get a shared lock on the coauthorable file and joins the coauthoring session of the file by adding the client's associated **ClientID** and timeout to the file coauthoring tracker.

If the file already has a shared lock on the server with the given schema lock identifier and the client has already joined the coauthoring session, the protocol server does both of the following:

- Refresh the timeout value associated with the **ClientID** in the file coauthoring tracker.
- Return an error code value set to "Success".

If the coauthoring feature is disabled on the protocol server, the server does one of the following:

- If the **AllowFallbackToExclusive** attribute is set to **true**, the protocol server gets an exclusive lock on the file.
- If the **AllowFallbackToExclusive** attribute is set to **false**, the protocol server returns an error code value set to "FileNotLockedOnServerAsCoauthDisabled".<34>

The **AllowFallbackToExclusive** attribute is defined in section 2.3.1.13. The result of the lock type obtained by the server MUST be sent as the **LockType** attribute in the **SchemaLockSubResponseDataType**. The **SchemaLockSubResponseDataType** is defined in section 2.3.1.15. The **LockType** attribute values are defined in section 2.2.5.9.

The protocol server returns an error code value set to "NumberOfCoauthorsReachedMax" when all of the following conditions are true:

- The maximum number of coauthorable clients allowed to join a coauthoring session to edit a coauthorable file has been reached.
- The current client is not allowed to edit the file because the limit has been reached.

If there is a current exclusive lock on the file or if there is a shared lock on the file with a different schema lock identifier, the protocol server returns an error code value set to "FileAlreadyLockedOnServer". If the coauthorable file is checked out on the server and it is checked out by a client with a different user name than the current client, the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer".

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8, and generic error code types are defined in section 2.2.5.6. For other unknown error types, the protocol server returns an error code value set to "LockRequestFail".

3.1.4.4.2 Release Lock

If the **SchemaLockRequestType** attribute is set to "ReleaseLock", the protocol server considers this schema lock subrequest to be of type, "Release lock". The protocol server processes the request to exit the coauthoring session and checks the number of clients editing the document at that instant in time. When the protocol server receives a schema lock subrequest of type "ReleaseLock" from the last and only client editing the document, the protocol server does both of the following:

- Delete the client identifier entry associated with the client in the file coauthoring tracker. The file coauthoring tracker is defined in section 3.1.1.

- Delete the coauthoring session and the shared lock on the file if the client that sent the subrequest of type "Release lock" is the last client in the file coauthoring tracker.

If the current client is not already present in the coauthoring session, the protocol server does one of the following:

- Return an error code of "InvalidCoauthSession" if there are other clients present in the coauthoring session.<35>
- Return "Success" if no clients are present in the coauthoring session.

The protocol server SHOULD<36> return an error code of "FileAlreadyLockedOnServer" if there is a current exclusive lock on the file or if there is a shared lock on the file with a different schema lock identifier.

If the coauthoring session has already been deleted, the protocol server returns an error code value set to "Success" as defined in section 2.2.5.6.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8, and generic error code types are defined in section 2.2.5.6. For other unknown error types, the protocol server returns an error code value set to "LockRequestFail".

3.1.4.4.3 Refresh Lock

If the **SchemaLockRequestType** attribute is set to "RefreshLock", the protocol server considers this schema lock subrequest to be of type, "Refresh lock". The protocol server refreshes the client's timeout of the shared lock on the coauthorable file.

If the refresh of the shared lock on the file for that specific client fails because the file is no longer locked since the timeout value expired on the lock, the protocol server does one of the following:

- It the coauthoring feature is enabled on the protocol server, the server considers this a schema lock subrequest of type "Get lock" and gets a new shared lock on the file.
- If the coauthoring feature is disabled the protocol server returns an error code value set to "FileNotLockedOnServerAsCoauthDisabled".<37>

After the protocol server has ensured that there is a shared lock with the same schema lock identifier used by the current client and that the client is sharing that shared lock on the file, the protocol server MUST update the client's timeout on the shared lock on the file. The new timeout value is the timeout value sent as part of the schema lock subrequest. If the client is sending a request to refresh the shared lock with a timeout value less than the current timeout on the shared lock, the protocol server considers the coauthoring subrequest of type "Refresh lock" as a no-operation instruction. The **Timeout** attribute is defined in section 2.3.1.13.

If there is a current exclusive lock on the file or if there is a shared lock on the file with a different schema lock identifier, the protocol server returns an error code value set to "FileAlreadyLockedOnServer". If the coauthorable file is checked out on the server and it is checked out by a client with a different user name than the current client, the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer".

Depending on the other types of errors, an implementation-dependent error code is returned by the protocol server. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8, and generic error code types are defined in section 2.2.5.6. For other unknown error types, the protocol server returns an error code value set to "LockRequestFail".

3.1.4.4.4 Convert to Exclusive Lock

If the **SchemaLockRequestType** attribute is set to "ConvertToExclusive", the protocol server considers this schema lock subrequest to be of type, "Convert to exclusive lock". The protocol server process the request to convert a shared lock on the coauthorable file to an exclusive lock on the file.

The protocol server performs the following operations:

- The conversion of the shared lock to an exclusive lock.
- The deletion of the coauthoring session.

The protocol server returns an error code value set to "InvalidCoauthSession" to indicate failure if any one of the following conditions is true:

- There is no shared lock.
- There is no coauthoring session for the file. The current client is not present in the coauthoring session.
- There is a current exclusive lock on the file or a shared lock on the file from another client with a different schema lock identifier.

The shared lock is converted to an exclusive lock only if one client is currently editing the document. If the shared lock is successfully converted to an exclusive lock, the protocol server MUST use the unique value of the **ExclusiveLockID** attribute sent by the client to identify the lock. The **ExclusiveLockID** attribute is specified in section 2.3.1.13.

If there is more than one client currently editing the file and the **ReleaseLockOnConversionToExclusiveFailure** attribute is set to **false**, the protocol server returns an error code value set to "MultipleClientsInCoauthSession" to indicate the failure to convert to an exclusive lock. The protocol server returns an error code value set to "ExitCoauthSessionAsConvertToExclusiveFailed" when the following conditions are both true:

- **The ReleaseLockOnConversionToExclusiveFailure** attribute is set to **true** in the coauthoring subrequest.
- Multiple clients are in the coauthoring session.

When the **ReleaseLockOnConversionToExclusiveFailure** attribute is set to **true** and the conversion to an exclusive lock failed, the protocol server removes the client from the coauthoring session on the file, and it removes the current client's **ClientID** from the file coauthoring tracker that was associated with that file. The **ReleaseLockOnConversionToExclusiveFailure** attribute is specified in section 2.3.1.13.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8, and generic error code types are defined in section 2.2.5.6. For other unknown error types, the protocol server returns an error code value set to "LockRequestFail".

3.1.4.4.5 Check Lock Availability

If the **SchemaLockRequestType** attribute is set to "CheckLockAvailability", the protocol server considers this schema lock subrequest to be of type "Check lock availability". The protocol server checks to see if a file is available to take a shared lock or exclusive lock.

If there is a current exclusive lock on the file or if there is a shared lock on the file with a different schema lock identifier, the protocol server returns an error code value set to "FileAlreadyLockedOnServer". If the coauthorable file is checked out on the server and it is checked out by a client with a different user name than the current client, the protocol server returns an error

code value set to "FileAlreadyCheckedOutOnServer". In all other cases, the protocol server returns an error code value set to "Success" to indicate the availability of the file for locking.

3.1.4.5 ExclusiveLock Subrequest

This operation is used to request an exclusive lock on the file or different types of exclusive lock operations on a file from the protocol server. Depending on the **ExclusiveLockRequestType** attribute value, the protocol server interprets the request as one of the following types of lock operations:

- Get lock
- Release lock
- Refresh lock
- Convert to schema lock with coauthoring transition
- Convert to schema lock
- Check lock availability

The **ExclusiveLockRequestType** attribute is defined in section 2.3.3.4. The **SubRequestData** element for an exclusive lock subrequest is of type **ExclusiveLockSubRequestDataType** and is defined in section 2.3.1.9.

The protocol client sends an exclusive lock **SubRequest** message, which is of type **ExclusiveLockSubRequestType** as specified in section 2.3.1.10. The protocol server responds with an exclusive lock **SubResponse** message, which is of type **ExclusiveLockSubResponseType** as specified in section 2.3.1.12. This is done as follows:

- The protocol client prepares a request containing a URL for the file, a unique **Request** token, and one or more **SubRequest** elements, as defined in section 2.2.3.2 and section 2.2.3.8. The **SubRequest** element is of type "ExclusiveLock" and the **SubRequestData** element contains attributes that are input parameters used by the protocol server when processing the exclusive lock subrequest. The **SubRequestData** element is of type **ExclusiveLockSubRequestDataType** and is defined in section 2.3.1.9.
- The protocol server receives the request, parses the logic, and depending on the type of exclusive lock subrequest, processes the request as specified in section 3.1.4.5.1, 3.1.4.5.2, 3.1.4.5.3, 3.1.4.5.4, 3.1.4.5.5, or 3.1.4.5.6.
- The **Response** element is defined in section 2.2.3.5, and the **SubResponse** element is defined in section 2.2.3.10. The **ExclusiveLockSubResponseDataType** defines the type of the **SubResponseData** element inside the exclusive lock **SubResponse** element. The **ExclusiveLockSubResponseDataType** is defined in section 2.3.1.11.
- The protocol returns results based on the following conditions:
  - Depending on the type of error, the **ErrorCode** is returned as an attribute of the **SubResponse** element. The **ErrorCode** attribute that is part of the **SubResponse** element is updated with a specific error code as specified in section 2.2.5.4.
  - If the protocol server was unable to find the URL for the file specified in the **Url** attribute, the protocol server reports a failure by returning an error code value set to "LockRequestFail" or "Unknown" or "FileNotExistsOrCannotBeCreated" in the **ErrorCode** attribute sent back in the **SubResponse** element.

- If the protocol server gets an exclusive lock subrequest of type "Convert to schema lock with coauthoring transition tracked" or "Convert to schema lock" for a file, and the conversion fails because the file is checked out by the current client, the protocol server returns an error code value set to "ConvertToSchemaFailedFileCheckedOutByCurrentUser".

- A concurrency violation happens when a current client's request to save the file coauthoring tracker fails because another client's request to edit and save the file coauthoring tracker is in progress on the server before the save is done by the current client. "CoauthRefblobConcurrencyViolation" is specified in section 2.2.5.8. The file coauthoring tracker is defined in section 3.1.1. The protocol server returns an error code value set to "CoauthRefblobConcurrencyViolation" when there is a concurrency violation and the exclusive lock subrequest is one of the following types:

  - Convert to schema lock with coauthoring transition tracked

  - Convert to schema lock

- An **ErrorCode** value of "Success" indicates success in processing the exclusive lock request.

3.1.4.5.1 Get Lock

If the **ExclusiveLockRequestType** attribute is set to "GetLock", the protocol server considers this exclusive lock subrequest to be of type "Get lock". The protocol server process the request to get an exclusive lock on the file.

The protocol server returns an error code value set to "FileAlreadyLockedOnServer" if one of the following conditions is true:
- The file is already locked with an exclusive lock with a different exclusive lock identifier.
- The file is already locked with a shared lock.

If the file is locked with the same exclusive lock identifier that is sent in the exclusive lock subrequest of type "Get lock", the protocol server refreshes the existing exclusive lock and returns an error code value set to "Success".

If the protocol server encounters an error because the document is not being checked out on the server and the file is saved to a document library that requires checking out files, the protocol server returns an error code value set to "DocumentCheckoutRequired". The "DocumentCheckoutRequired" error code value indicates to the protocol client that a checkout needs to be done before the file can be locked. The checkout of the file is done by the client using the **CheckoutFile** web service call as specified in [MS-LISTSWS].

If the protocol server encounters an issue in locking the file because a checkout has already been done by another client, the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer". If the checkout of the file has been done by the current client, the protocol server MUST allow an exclusive lock on the file. If the protocol server encounters unknown exceptions or failures when trying to get a lock on the file, the protocol server returns an error code value that is set to "LockRequestFail" to indicate an unknown failure.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific.
DependencyCheckRelatedErrorCodeTypes is defined in section 2.2.5.2.
LockCoauthRelatedErrorCodeTypes is defined in section 2.2.5.8.

3.1.4.5.2 Release Lock

If the **ExclusiveLockRequestType** attribute is set to "ReleaseLock", the protocol server considers this exclusive lock subrequest of to be of type "Release lock". The protocol server releases the exclusive lock session on the file.

If the protocol server encounters an error because no lock currently exists on the file, the protocol server returns an error code value set to "FileNotLockedOnServer". The protocol server returns an error code value set to "FileAlreadyLockedOnServer" if any one of the following conditions is true:

- The file is already locked with an exclusive lock that has a different exclusive lock identifier.
- The file is already locked with a shared lock.

If the protocol server encounters unknown exceptions or failures when trying to release a lock on the file, the protocol server returns an error code value that is set to "LockRequestFail" to indicate an unknown failure.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **GenericErrorCodeTypes** is defined in section 2.2.5.6. **DependencyCheckRelatedErrorCodeTypes** is defined in section 2.2.5.2. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8.

3.1.4.5.3 Refresh Lock

If the **ExclusiveLockRequestType** attribute is set to "RefreshLock", the protocol server considers this exclusive lock subrequest to be of type "Refresh lock". The protocol server refreshes the exclusive lock on the file.

If the refresh of the exclusive lock fails because no exclusive lock exists on the file, the protocol server gets a new exclusive lock on the file. The protocol server returns an error code value set to "FileAlreadyLockedOnServer" if any one of the following conditions is true:

- The protocol server is unable to refresh the lock on the file because a shared lock already exists on the file.
- The protocol server is unable to refresh the lock on the file because an exclusive lock with a different exclusive lock identifier exists on the file.

If the protocol server encounters an error because the document is not checked out on the server and the file is saved to a document library that requires checking out files, the protocol server returns an error code value set to "DocumentCheckoutRequired". The "DocumentCheckoutRequired" error code value indicates to the protocol client that a checkout needs to be done before the file can be locked and the lock refreshed. The checkout of the file MUST be done by the client using the **CheckoutFile** web service call as specified in [MS-LISTSWS].

If the protocol server encounters an error in locking the file because a checkout has already been done by another client, the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer". If the protocol server encounters unknown exceptions or failures when trying to refresh the lock on the file, the protocol server returns an error code value that is set to "LockRequestFail" to indicate an unknown failure.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **GenericErrorCodeTypes** is defined in section 2.2.5.6. **DependencyCheckRelatedErrorCodeTypes** is defined in section 2.2.5.2. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8.

3.1.4.5.4 Convert to Schema Lock with Coauthoring Transition Tracked

If the **ExclusiveLockRequestType** attribute is set to "ConvertToSchemaJoinCoauth", the protocol server considers this exclusive lock subrequest to be of type "Convert to schema lock with coauthoring transition tracked". When the protocol server receives this subrequest, it does all of the following:

- Converts the exclusive lock on the file to a shared lock.
- Starts a coauthoring session for the file if one is not already present and adds the client to that session.

After the request to convert the exclusive lock to a shared lock is processed successfully, the protocol server gets the coauthoring status and returns the status to the client.

The protocol server uses the **ClientID** attribute sent in an exclusive lock subrequest of type "Convert to schema lock with coauthoring transition tracked" to uniquely identify each client and keep track of each client and its timeout on the shared lock for the file. The protocol server also uses the **ClientID** sent in the exclusive lock subrequest to decide when to release the shared lock on the file. The protocol server uses the **SchemaLockID** attribute sent in an exclusive lock subrequest of type "Convert to schema lock with coauthoring transition tracked" to ensure that after the exclusive lock on the file is converted to a shared lock, the protocol server MUST allow only other clients with the same schema lock identifier to share the lock on the file. The **SchemaLockID** and **ClientID** attributes are defined in section 2.3.1.9.

The protocol server returns error codes according to the following rules:

- If the feature of transitioning a file that currently has an exclusive lock to one that has a shared lock is not supported by the protocol server, the protocol server returns an error code value set to "RequestNotSupported".
- If the coauthoring feature is disabled by the protocol server, the protocol server returns an error code value set to "LockNotConvertedAsCoauthDisabled".
- If the protocol server is unable to convert the exclusive lock to a shared lock on the file because the file is checked out by the current user, the protocol server returns an error code value set to "ConvertToSchemaFailedFileCheckedOutByCurrentUser".
- If the protocol server is unable to convert the lock because either there is an exclusive lock with a different exclusive lock identifier or there is a shared lock already present on the file, the protocol server returns an error code value set to "FileAlreadyLockedOnServer".
- If the protocol server is unable to convert the lock on the file because no lock exists on the server, the protocol server returns an error code value set to "FileNotLockedOnServer".
- If the protocol server is unable to convert the lock because the document is saved to a document library that requires checking out files and the document is not checked out on the server, the protocol server returns an error code value set to "DocumentCheckoutRequired". The "DocumentCheckoutRequired" error code value indicates to the protocol client that a checkout needs to be done before the exclusive lock is converted to a shared lock. The checkout of the file is done by the client using the **CheckoutFile** web service call as specified in [MS-LISTSWS].
- If the protocol server encounters unknown exceptions or failures when converting the lock on the file, the protocol server returns an error code value set to "LockRequestFail".

To get the coauthoring status, the protocol server checks the number of clients editing the document at that instant in time. If the current client is the only client editing the file, the protocol server MUST return a **CoauthStatus** attribute set to "Alone", which indicates that no one else is editing the file. If the current client is not the only client editing the document, the protocol server MUST return a **CoauthStatus** attribute set to "Coauthoring", which indicates that the current client is coauthoring when editing the document. The "Coauthoring" status is possible because the operations of converting the exclusive lock on the file to a shared lock and then adding the client to the coauthoring session are not executed atomically, and another client could add itself to the coauthoring session after the lock was converted but before the current client was added to the session. The **CoauthStatus** attribute is sent as part of the **SubResponseData** element and is defined in section 2.2.8.2.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are

directly returned by the protocol server are implementation-specific. **GenericErrorCodeTypes** is defined in section 2.2.5.6. **DependencyCheckRelatedErrorCodeTypes** is defined in section 2.2.5.2. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8.

3.1.4.5.5 Convert to Schema Lock

If the **ExclusiveLockRequestType** attribute is set to "ConvertToSchema", the protocol server considers this exclusive lock subrequest to be of type "Convert to schema lock". The protocol server process the request by converting an exclusive lock on the file to a shared lock. The "Convert to schema lock" type of exclusive lock subrequest is a subset of the "Convert to schema lock with coauthoring transition tracked" type of exclusive lock subrequest. The "Convert to schema lock with coauthoring transition tracked" type of subrequest converts the exclusive lock on the file to a shared lock and keeps track of all the clients' user information for those that are editing the file concurrently. The "Convert to schema lock" type of subrequest converts the exclusive lock on the file to a shared lock and does not keep track of the coauthoring transition. The "Convert to schema lock" type and "Convert to schema lock with coauthoring transition tracked" types of subrequests both add the client identifiers to the file coauthoring tracker.

The protocol server uses the **ClientID** attribute sent in an exclusive lock subrequest of type "Convert to schema lock" to uniquely identify each client and keep track of each client's timeout on the shared lock for the file. The **ClientID** attribute is defined in section 2.3.1.9.

The protocol server returns error codes according to the following rules:

- If the feature of transitioning a file that currently has an exclusive lock to one that has a shared lock is not supported by the protocol server, the protocol server returns an error code value set to "RequestNotSupported".
- If the feature of coauthoring is not completely supported by the protocol server, the protocol server returns an error code value set to "LockNotConvertedAsCoauthDisabled".
- If the protocol server is unable to convert the exclusive lock to a shared lock on the file because the file is checked out by the current user, the protocol server returns an error code value set to "ConvertToSchemaFailedFileCheckedOutByCurrentUser".
- If the protocol server is unable to convert the lock because either there is an exclusive lock with a different exclusive lock identifier or there is a shared lock already present on the file, the protocol server returns an error code value set to "FileAlreadyLockedOnServer".
- If the protocol server is unable to convert the lock on the file because no lock exists on the server, the protocol server returns an error code value set to "FileNotLockedOnServer".
- If the protocol server is unable to convert the lock because the document is saved to a document library that requires checking out files and the document is not checked out on the server, the protocol server returns an error code value set to "DocumentCheckoutRequired". The "DocumentCheckoutRequired" error code value indicates to the protocol client that a checkout needs to be done before the exclusive lock is converted to a shared lock. The checkout of the file is done by the client using the **CheckoutFile** web service call as specified in [MS-LISTSWS].
- If the protocol server encounters unknown exceptions or failures when converting the lock on the file, the protocol server returns an error code value set to "LockRequestFail".

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an error. The error that the protocol server returns is implementation-specific. Errors that are directly returned by the protocol server are implementation-specific. **GenericErrorCodeTypes** is defined in section 2.2.5.6. **DependencyCheckRelatedErrorCodeTypes** is defined in section 2.2.5.2. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8.

3.1.4.5.6 Check Lock Availability

If the **ExclusiveLockRequestType** attribute is set to "CheckLockAvailability", the protocol server considers this exclusive lock subrequest to be of type "Check lock availability". The protocol server checks if a file is available to take a shared lock or exclusive lock.

The protocol server returns error codes according to the following rules:

- If there is a current exclusive lock on the file with a different exclusive lock identifier than the one specified by the current client or if there is a shared lock on the file, the protocol server returns an error code value set to "FileAlreadyLockedOnServer".
- If the file is checked out on the server, but it is checked out by a client with a different user name than that of the current client, the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer".
- In all other cases, the protocol server returns an error code value set to "Success" to indicate the availability of the file for locking.
- If the protocol server encounters unknown exceptions or failures in processing the exclusive lock subrequest of type "Check lock availability", the protocol server returns an error code value that is set to "LockRequestFail".

3.1.4.6 WhoAmI Subrequest

This operation is used to retrieve the current client's user information, which helps in showing a client's friendly name when a coauthorable file is being edited by more than one client.

The protocol client sends a **WhoAmI SubRequest** message, which is of type **WhoAmISubRequestType** as specified in section 2.3.1.20. The protocol server responds with a **WhoAmI SubResponse** message, which is of type **WhoAmISubResponseType** as specified in section 2.3.1.22. This is done as follows:

1. The protocol client prepares a request containing a URL for the file, a unique request token, and one or more **SubRequest** elements, as defined in section 2.2.3.2 and section 2.2.3.8. The **SubRequest** element is of type "WhoAmI".
2. The protocol server receives the request and parses the logic to request the client-specific user information. The requested client-specific user information is prepared as a response and sent back to the protocol client.

The **Response** element is defined in section 2.2.3.5 and the **SubResponse** element is defined in section 2.2.3.10. The **WhoAmISubResponseDataType** defines the type of the **SubResponseData** element inside the **WhoAmI SubResponse** element. The **WhoAmISubResponseDataType** is defined in section 2.3.1.21. The protocol server sends the requested client-specific user information as attributes in the **WhoAmI SubResponseData** element. The attributes sent by the protocol server are defined in section 2.3.3.6.

The protocol server returns results based on the following conditions:

- If the processing of the **WhoAmI** subrequest by the protocol server failed to get the client-specific user information or encountered an unknown exception, the protocol server returns an error code value set to "SubRequestFail".
- Otherwise, the protocol server sets the error code value to "Success" to indicate success in processing the **WhoAmI** subrequest.

3.1.4.7 ServerTime Subrequest

This operation is used to retrieve the server time, which is expressed as the number of ticks (that is, the number of 100-nanosecond intervals) that have elapsed since 00:00:00, January 1, 0001. The server time SHOULD<38> be expressed in Coordinated Universal Time (UTC).

The protocol client sends a **ServerTime SubRequest** message, which is of **ServerTimeSubRequestType** as specified in section 2.3.1.17. The protocol server responds with a **ServerTime SubResponse** message, which is of type **ServerTimeSubResponseType** as specified in section 2.3.1.19. This is done as follows:

1. The protocol client prepares a request containing a URL for the file, a unique request token, and one or more **SubRequest** elements, as defined in section 2.2.3.2 and section 2.2.3.8. The **SubRequest** element is of type "ServerTime".

2. The protocol server receives the request and gets the server time information from the server. The requested server time information is prepared as a response and sent back to the protocol client.

The **Response** element is defined in section 2.2.3.5, and the **SubResponse** element is defined in section 2.2.3.10. The **ServerTimeSubResponseDataType** defines the type of the **SubResponseData** element that is sent in a server time **SubResponse** element. The **ServerTimeSubResponseDataType** is defined in section 2.3.1.18. The protocol server sends the server time as a **ServerTime** attribute in the **ServerTime SubResponseData** element. The **ServerTime** attribute is defined in section 2.3.1.18.

The protocol returns results based on the following conditions:

- If the processing of the **ServerTime** subrequest by the server fails to get the server time or encountered an unknown exception, the protocol server returns an error code value set to "SubRequestFail".

- Otherwise, the protocol server sets the error code value to "Success" to indicate success in processing the **ServerTime** subrequest.

3.1.4.8 EditorsTable Subrequest

This operation is used to request that the protocol server store an editors table entry on the file for the client or perform other editors table operations on a file.<39><40> Depending on the **EditorsTableRequestType** attribute value, the protocol server interprets the subrequest as one of the following types of editors table operations:

- Join editing session

- Leave editing session

- Refresh editing session

- Update editor metadata

- Remove editor metadata

The **EditorsTableRequestType** attribute is defined in section 2.3.3.7. The **EditorsTableRequestType** attribute is one of the attributes for the **EditorsTableSubRequestData** element, which is of type **EditorsTableSubRequestDataType**. The **EditorsTableSubRequestDataType** is defined in section 2.3.1.23.

The protocol client sends an editors table **SubRequest** message, which is of type **EditorsTableSubRequestType** as specified in section 2.3.1.24. The protocol server responds with an editors table **SubResponse** message, which is of type **EditorsTableSubResponseType** as specified in section 2.3.1.25. This is done as follows:

- The protocol client prepares a request containing a URL for the file, a unique request token, and one or more **SubRequest** elements, as defined in section 2.2.3.2 and section 2.2.3.8. The **SubRequest** element is of type "EditorsTable", and the **SubRequestData** element contains attributes that are input parameters used by the server when processing the editors table

subrequest. The **SubRequestData** element is of type **EditorsTableSubRequestDataType** and is defined in section 2.3.1.23.

- The protocol server receives the request and parses the logic. Depending on the type of editors table request, the protocol server processes the subrequest as specified in section 3.1.4.8.1, 3.1.4.8.2, 3.1.4.8.3, 3.1.4.8.4 or 3.1.4.8.5. (The **Response** element is defined in section 2.2.3.5, and the **SubResponse** element is defined in section 2.2.3.10.) The protocol server uses the **ClientID** attribute sent in an editors table subrequest to do the following:
  - Uniquely identify each client
  - Keep track of each client and its timeout on its entry in the editors table
  - Optionally store information about the user, such as a user name or email address

The protocol server returns results based on the following conditions:

- Depending on the type of error, the **ErrorCode** is returned as an attribute of the **SubResponse** element. The **ErrorCode** attribute that is part of the **SubResponse** element is updated with a specific error code as specified in section 2.2.5.4.
- If the protocol server was unable to find the URL for the file specified in the **Url** attribute, the protocol server reports a failure by returning an error code value set to "FileNotExistsOrCannotBeCreated" in the **ErrorCode** attribute sent back in the **SubResponse** element.
- If the **ClientID** does not currently exist in the editors table, the protocol server returns an error code value set to "EditorClientIdNotFound" for the **Update editor metadata** request.
- The protocol server returns an error code value set to "EditorMetadataQuotaReached" for an "Update editor metadata" request if the client has already exceeded its quota for key/value pairs.<41>
- The protocol server returns an error code value set to "EditorMetadataStringExceedsLengthLimit" for an "Update editor metadata" request if the key exceeds the server's length limit.
- The protocol server returns an error code value set to "InvalidSubRequest" if server does not support this request type.
- The protocol server returns an error code value set to "Success" to indicate success in processing the **EditorsTable** request.

To retrieve an editors table, a protocol client must send a **QueryChanges** request as specified in [MS-FSSHTTPB], with the **PartitionID** attribute set to the corresponding editors table partition on the server.<42>

3.1.4.8.1 Join Editing Session

If the **EditorsTableRequestType** attribute is set to "JoinEditingSession", the protocol server considers the editors table subrequest to be of type "Join editing session". The protocol server processes this request to add an entry to the editors table associated with the coauthorable file by adding the client's associated **ClientID**, **Timeout**, and **AsEditor** status in an entry.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an implementation-specific error. Errors that are directly returned by the protocol server are implementation-specific. Generic error code types are defined in section 2.2.5.6.

3.1.4.8.2 Leave Editing Session

If the **EditorsTableRequestType** attribute is set to "LeaveEditingSession", the protocol server considers the editors table subrequest to be of type "Leave editing session." The protocol server processes this request to remove the entry in the editors table associated with the coauthorable file corresponding to the client with the given **ClientID**.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an implementation-specific error.

3.1.4.8.3 Refresh Editing Session

If the **EditorsTableRequestType** attribute is set to "RefreshEditingSession", the protocol server considers this editors table subrequest to be of type "Refresh editing session." The protocol server processes this request to refresh the **Timeout** value in the entry in the editors table associated with the coauthorable file corresponding to the client with the given **ClientID**.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an implementation-specific error.

3.1.4.8.4 Update Editor Metadata

If the **EditorsTableRequestType** attribute is set to "UpdateEditorMetadata", the protocol server considers this editors table subrequest to be of type "Update editor metadata." The protocol server processes this request to add or update the client-supplied key/value pair in the entry in the editors table associated with the coauthorable file corresponding to the client with the given **ClientID**.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an implementation-specific error.

3.1.4.8.5 Remove Editor Metadata

If the **EditorsTableRequestType** attribute is set to "RemoveEditorMetadata", the protocol server considers this editors table subrequest to be of type "Remove editor metadata." The protocol server processes this request to remove the client-supplied key/value pair for the given key in the entry in the editors table associated with the coauthorable file corresponding to the client with the given **ClientID**.

If any failure occurs such that the subrequest cannot be processed successfully, the protocol server returns an implementation-specific error.

3.1.4.9 GetDocMetaInfo Subrequest

This operation is used to retrieve server metadata properties pertaining to the server file and its parent folder<43>. Valid metadata properties are as specified in [MS-FPSE] section 2.2.4.

The protocol client MUST send the **GetDocMetaInfo SubRequest** message only if the **X-MSFSSHTTP** header, as described in [MS-OCPROTO] section 2.1.2.1.2, has a value of 1.1 or greater. The protocol client sends a **GetDocMetaInfo SubRequest** message, which is of type **GetDocMetaInfoSubRequestType** as specified in section 2.3.1.26. The protocol server responds with a **GetDocMetaInfo SubResponse** message, which is of type **GetDocMetaInfoSubResponseType** as specified in section 2.3.1.30. This is done as follows:

1. The protocol client prepares a request containing a URL for the file, a unique request token, and one or more **SubRequest** elements, as defined in section 2.2.3.2 and section 2.2.3.8. The **SubRequest** element is of type "GetDocMetaInfo".

2. The protocol server receives the request and parses the logic to request the metadata. The requested document and parent directory metadata are prepared as a response and sent back to the protocol client.

The **Response** element is defined in section 2.2.3.5, and the **SubResponse** element is defined in section 2.2.3.10. **GetDocMetaInfoSubResponseDataType** defines the type of the **SubResponseData** element inside the **GetDocMetaInfo SubResponse** element. **GetDocMetaInfoSubResponseDataType** is defined in section 2.3.1.27. The protocol server sends the requested metadata as child elements of the **GetDocMetaInfo SubResponseData** element. Document metadata and directory metadata are specified in the **DocProps** and the **FolderProps** element, respectively, both of which are of type **GetDocMetaInfoPropertySetType** as defined in section 2.3.1.28. The **GetDocMetaInfoPropertySetType** elements have one **Property** element per metadata item of type **GetDocMetaInfoPropertyType** as defined in section 2.3.1.29.

The protocol returns results based on the following conditions:

- If the processing of the **GetDocMetaInfo** subrequest by the protocol server failed to get the requested metadata or encountered an unknown exception, the protocol server returns an error code value set to "SubRequestFail".
- Otherwise, the protocol server sets the error code value to "Success" to indicate success in processing the **GetDocMetaInfo** subrequest.

3.1.4.10 GetVersions Subrequest

This operation is used to retrieve information about a file's versions<44>.

The protocol client MUST send the **GetVersions SubRequest** message only if the **X-MSFSSHTTP** header, as described in [MS-OCPROTO] section 2.1.2.1.2, has a value of 1.1 or greater. The protocol client sends a **GetVersions SubRequest** message, which is of type **GetVersionsSubRequestType** as specified in section 2.3.1.31. The protocol server responds with a **GetVersions SubResponse** message, which is of type **GetVersionsSubResponseType** as specified in section 2.3.1.32. This is done as follows:

1. The protocol client prepares a request containing a URL for the file, a unique request token, and one or more **SubRequest** elements, as defined in section 2.2.3.2 and section 2.2.3.8. The **SubRequest** element is of type "GetVersions".
2. The protocol server receives the request and parses the logic to request information about a file's versions. The requested file version data is prepared as a response and sent back to the protocol client.

The **Response** element is defined in section 2.2.3.5, and the **SubResponse** element is defined in section 2.2.3.10. The **Results** element, as specified in [MS-VERSS] section 2.2.4.1, is a complex type that specifies information about the file's versions.

The protocol returns results based on the following conditions:

- If the processing of the **GetVersions** subrequest by the protocol server failed to get the requested versions information or encountered an unknown exception, the protocol server returns an error code value set to "SubRequestFail".
- Otherwise, the protocol server sets the error code value to "Success" to indicate success in processing the **GetVersions** subrequest.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

4 Protocol Examples

This section provides common scenarios involving the use of the different types of subrequest operations for synchronizing a file's contents or metadata contents.

While key elements and attributes are described in detail in each of the scenarios in the following subsections, some elements and attributes are not described completely for the sake of brevity and readability.

4.1 Successful File Open of a Coauthorable Document

A client wants to open a file on a protocol server. This file is a coauthorable document. The client successfully opens the coauthorable document by sending a series of subrequests to initiate a download of the file contents for shared editing.

The protocol server is named Example.

The source file to be opened is http://Example/shared%20documents/test1.docx.

4.1.1 Request


```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <RequestVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
    <RequestCollection CorrelationId="{A2FFBFA0-50BA-47EC-81CB-D5627A458768}"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <Request Url="http://Example/shared%20documents/test1.docx" RequestToken="1">
        <SubRequest Type="Coauth" SubRequestToken="1">
          <SubRequestData CoauthRequestType="JoinCoauthoring" SchemaLockID=" 29358EC1-E813-
4793-8E70-ED0344E7B73C" ClientID="{BE07F85A-0CD1-4862-BDFC-F6CC3C8588A4}" Timeout="3600"
AllowFallbackToExclusive="true" ExclusiveLockID="{BE07F85A-0CD1-4862-BDFC-F6CC3C8588A4}"/>
        </SubRequest>
        <SubRequest Type="SchemaLock" SubRequestToken="2" DependsOn="1"
DependencyType="OnNotSupported">
          <SubRequestData SchemaLockRequestType="GetLock" SchemaLockID="29358EC1-E813-4793-
8E70-ED0344E7B73C" ClientID="{BE07F85A-0CD1-4862-BDFC-F6CC3C8588A4}" Timeout="3600"
AllowFallbackToExclusive="true" ExclusiveLockID="{BE07F85A-0CD1-4862-BDFC-F6CC3C8588A4}"/>
        </SubRequest>
        <SubRequest Type="Cell" SubRequestToken="6" DependsOn="2" DependencyType="OnExecute">
          <SubRequestData PartitionID="7808f4dd-2385-49d6-b7ce-37aca5e43602"
BinaryDataSize="88">DAALAJzPKfM5lAabBgIAAO4CAACqAiAAfrgx50XdqkSrgAx1+9FTDnoCCAD0J0UfdwEWAgYAA
wUAigICAADaAgYAAwAAygIIAAgAgAOEAEELAawCAFUDAQ==</SubRequestData>
        </SubRequest>
        <SubRequest Type="Cell" SubRequestToken="4" DependsOn="2" DependencyType="OnExecute">
          <SubRequestData GetFileProps="true"
BinaryDataSize="248">DAALAJzPKfM5lAabBgIAAO4CAACqAiAAfrgx50XdqkSrgAx1+9FTDnoCCAD0J0UfdwEWAgYA
AwUAigICAADaAgYAAwAAygIIAAgAgAOEACYCIAD2NXoyYQcURJaGUekAZnpNpAB4JE1n0+yUtD6/lXGrhF54W34Ac3gks
pgsE2tLwUCVcauEXnhbfgBrURMBJgIgABMfCRCCyPtAmIZlM/k0wh1sAXAtDPkLQTdv0ZlEpsMnIy7cpxEJMgAAALUTAS
YCIAAO6XY6MoAMTbnd88ZQKUM+TAEgJgyymCwTa0vBQJVxq4ReeFt+awClEwFBCwGsAgBVAwE=</SubRequestData>
        </SubRequest>
        <SubRequest Type="Cell" SubRequestToken="3" DependsOn="2" DependencyType="OnExecute">
          <SubRequestData PartitionID="383adc0b-e66e-4438-95e6-e39ef9720122"
BinaryDataSize="88">DAALAJzPKfM5lAabBgIAAO4CAACqAiAAfrgx50XdqkSrgAx1+9FTDnoCCAD0J0UfdwEWAgYAA
wUAigICAADaAgYAAwAAygIIAAgAgAOEAEELAawCAFUDAQ==</SubRequestData>
        </SubRequest>
        <SubRequest Type="ServerTime" SubRequestToken="5"/>
        <SubRequest Type="WhoAmI" SubRequestToken="7"/>
      </Request>
    </RequestCollection>
  </s:Body>
</s:Envelope>
```

The protocol client sends seven **SubRequest** elements as part of the **Request** element in the cell storage service request message for opening the file contents. The file whose contents need to be opened is uniquely identified by the URL for the file. The URL for the file is specified as part of the **Url** attribute of the **Request** element, as specified in section 2.2.3.2. Each **SubRequest** element specifies a type of subrequest to the protocol server and is uniquely identified by the **SubRequestToken** attribute. Details about the **SubRequestToken** attribute are specified in section 2.2.4.3. The type of the subrequest is specified as part of the **Type** attribute for each **SubRequest** element, as specified in section 2.2.4.2.

The first **SubRequest** element is a coauthoring subrequest of type "Join coauthoring session" that requests a shared lock on the file and the coauthoring status. Details about the coauthoring subrequest of type "Join coauthoring session" are specified in section 3.1.4.3.1.

The second **SubRequest** element is a schema lock subrequest of type "Get lock" that also requests a shared lock on the file. Unlike the first **SubRequest** element, the coauthoring subrequest, the schema lock subrequest does not get the coauthoring status. This schema lock subrequest is executed only if the first **SubRequest** element, the coauthoring subrequest, is not supported by the protocol server. The dependency of the schema lock subrequest on the first **SubRequest** element is defined by the **DependsOn** attribute and the **DependencyType** attribute of the schema lock subrequest. In this case, the **DependsOn** value of 1 specifies the **SubRequestToken** of the first **SubRequest** element, which the second element is dependent on. The **DependencyType** value of "OnNotSupported" specifies that the second schema lock subrequest gets called only if the first **SubRequest** element is not supported. Details about the **DependsOn** and **DependencyType** attributes are specified in section 2.2.4.3. Details about the schema lock subrequest of type "Get lock" are specified in section 3.1.4.4.1.

The third, fourth, and fifth **SubRequest** elements are cell subrequests that request the download of file contents or file metadata contents. All three cell subrequests are dependent on the second **SubRequest** element, as defined by the **DependsOn** attribute with a value of 2, which specifies the **SubRequestToken** of the second **SubRequest** element, and the **DependencyType** attribute with a value of "OnExecute", which specifies that the cell subrequest gets executed only after the second **SubRequest** element is executed.

The third and fifth **SubRequest** elements specify the **PartitionID** attribute, while the fourth **SubRequest** element specifies the **GetFileProps** attribute. The **PartitionID** attribute specifies the **Partition-block** subfile contents or file metadata contents that are downloaded. The binary data sent in the **SubRequestData** element of a cell subrequest indicates if this is a file content upload or download request. The cell subrequest for a download of file contents is processed as described in [MS-FSSHTTPB] section 3.1.4.2. In the fourth **SubRequest** element, the **GetFileProps** attribute is set to "true" to request the properties of the file. Details about the **PartitionID** attribute and **GetFileProps** attribute are specified in section 2.3.3.1. Details about the cell subrequest are specified in section 3.1.4.2.

The sixth **SubRequest** element is a **ServerTime** subrequest that gets server time information, as specified in section 3.1.4.7.

The seventh **SubRequest** element is a **WhoAmI** subrequest that requests the current client user information. This client user information helps in showing a client's friendly name when a coauthorable file is edited by more than one client. Details about the **WhoAmI** subrequest are specified in section 3.1.4.6.

In **SubRequest** elements where the **SchemaLockID** attribute string is present, the string is "29358EC1-E813-4793-8E70-ED0344E7B73C".

4.1.2 Response

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
```

```
  <s:Body>
    <ResponseVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
    <ResponseCollection WebUrl="http://Example"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <Response Url="http://Example/shared%20documents/test1.docx" RequestToken="1"
HealthScore="0">
        <SubResponse SubRequestToken="1" ErrorCode="Success" HResult="0">
          <SubResponseData LockType="SchemaLock" CoauthStatus="Alone" TransitionID="600ce272-
068f-4bd7-a1fb-4ac10c54386c"/>
        </SubResponse>
        91
        <SubResponse SubRequestToken="2"
ErrorCode="DependentOnlyOnNotSupportedRequestGetSupported" HResult="2147500037">
          <SubResponseData/>
        </SubResponse>
        1bc
        <SubResponse SubRequestToken="6" ErrorCode="Success" HResult="0">
          <SubResponseData CoalesceHResult="0"
ContainsHotboxData="True">DAALAJ3PKfM5lAabFgMCAACsAgAMVgy19aRX2CXZQo8Tj3agbNI7gHJo8x4AR9pJiSD
iCMP7d3ABAAAAAAAAAAOIVAz7/6ntX2CcT4gqqc6gT8RPgGskUtRR7xZFtwAl/eVQjrACAAAAAAAAAAUMVgz7/6ntX2Cc
T4gqqc6gT8RPgGskUtRR7xZFtwAl/eVQjrABAAAAAAAAAAVgIAAAAAAAAAAAAAAAAAAAAAAFVQ4CBgADBQD6AiQADLX1p
FfYJdlCjxOPdqBs0jsAhABBBwGLAQ==</SubResponseData>
        </SubResponse>
        29d
        <SubResponse SubRequestToken="4" ErrorCode="Success" HResult="0">
          <SubResponseData Etag="&quot;{600CE272-068F-4BD7-A1FB-4AC10C54386C},1&quot;"
CoalesceHResult="0" ContainsHotboxData="False" CreateTime="129092725940000000"
LastModifiedTime="129092725940000000" ModifiedBy="Jayne
Darcy">DAALAJ3PKfM5lAabFgMCAACsAgAMVgxdgv0sL6qXQpGLEnn1TeJGgEXxEmphk/RDliumkvzDIGwBAAAAAAAAAA
MFVQ4CBgADBQD6AiQADF2C/SwvqpdCkYsSefVN4kYAhAAmAiAA9jV6MmEHFESWhlHpAGZ6TaQAeCRNZ9PslLQ+v5Vxq4R
eeFt+AHN4JLKYLBNrS8FAlXGrhF54W34Aa1ETASYCIAATHwkQgsj7QJiGZTP5NMIdbAFwLQz5C0E3b9GZRKbDJyMu3KcR
CTIAAAClEwEmAiAADul2OjKADE253fPGUClDPkwBICYMspgsE2tLwUCVcauEXnhbfmsApRMBQQcBiwE=</SubResponse
Data>
        </SubResponse>
        1bc
        <SubResponse SubRequestToken="3" ErrorCode="Success" HResult="0">
          <SubResponseData CoalesceHResult="0"
ContainsHotboxData="True">DAALAJ3PKfM5lAabFgMCAACsAgAMVgx6KmdPAwAaQpf2Pb9Nlj2NgCminTMNs9JDuty
BtjY1HeoBAAAAAAAAAAOIVAxZA/FHJtJ3R6iV1MCVAWthgNcfQoujYe9GnWDh5152y9MCAAAAAAAAAAUMVgxZA/FHJtJ3
R6iV1MCVAWthgNcfQoujYe9GnWDh5152y9MBAAAAAAAAAAVgIAAAAAAAAAAAAAAAAAAAAAAFVQ4CBgADBQD6AiQADHoqZ
08DABpCl/Y9v02WPY0AhABBBwGLAQ==</SubResponseData>
        </SubResponse>
        81
        <SubResponse SubRequestToken="5" ErrorCode="Success" HResult="0">
          <SubResponseData ServerTime="634004247240000000"/>
        </SubResponse>
        ed
        <SubResponse SubRequestToken="7" ErrorCode="Success" HResult="0">
          <SubResponseData UserName="Jayne Darcy" UserLogin="EXAMPLE\jdarcy"
UserEmailAddress="jdarcy@x.example.com" UserSIPAddress="jdarcy@example.com"/>
        </SubResponse>
        36
      </Response>
    </ResponseCollection>
  </s:Body>
</s:Envelope>
```


For each **SubRequest** element that is in a **Request** element of a cell storage service request message, the protocol server sends a corresponding **SubResponse** element in a **Response** element as part of the cell storage service response message. Each **SubResponse** element contains the **SubRequestToken** attribute and the **ErrorCode** attribute. The **SubRequestToken** attribute identifies the **SubRequest** element for which the **SubResponse** was generated by the protocol server. The **ErrorCode** attribute specifies the error code result for the specific subrequest. The

SubRequestToken attribute and the **ErrorCode** attribute that are part of a **SubResponse** element are specified in section 2.2.4.6.

The **SubResponse** element with a **SubRequestToken** set to 1 in this example is the response from the protocol server for the coauthoring subrequest of type "Join coauthoring session". The **ErrorCode** attribute of "Success" indicates that the coauthoring subrequest was successfully processed, as specified in section 2.2.5.6. The **SubResponseData** element of the first **SubResponse** element specifies the type of lock granted in the **LockType** attribute and the coauthoring status in the **CoauthStatus** attribute. In this **SubResponse** element, the **LockType** attribute of "SchemaLock" indicates a shared lock on the file, and the **CoauthStatus** attribute of "Alone" indicates that only the current client is in the coauthoring session and editing the file. Details about the **LockType** attribute and **CoauthStatus** attribute are specified in section 2.3.1.7.

The **SubResponse** element with a **SubRequestToken** set to 2 is the response for the schema lock subrequest of type "Get Lock". The **ErrorCode** attribute of "DependentOnlyOnNotSupportedRequestGetSupported", defined in section 2.2.5.2, indicates that the coauthoring subrequest on which this schema lock subrequest is dependent was supported by the protocol server, so the schema lock subrequest was not executed.

The **SubResponse** elements with **SubRequestToken** values of 3, 4, and 5 are the responses from the protocol server for the corresponding cell subrequests. The **ErrorCode** attributes in each **SubResponse** element of "Success" indicate that all three cell subrequests were successfully processed. The third and fifth **SubResponse** elements contain the requested file contents or file metadata contents. The fourth **SubResponse** element contains the file properties requested as part of the corresponding cell subrequest. The **Etag** attribute in the fourth **SubResponse** element specifies the file version so that the protocol client knows which version of the file contents it is receiving. Details about the **Etag** attribute are specified in section 2.3.3.2.

The format of the binary data values of the **SubResponseData** elements is described in [MS-FSSHTTPB] section 2.2.3.1.3.

The **SubResponse** element with a **SubRequestToken** set to 6 is the response for the **ServerTime** subrequest. The **ErrorCode** attribute of "Success" indicates that the **ServerTime** subrequest was successfully processed. The **SubResponseData** element specifies the server time with the **ServerTime** attribute, as specified in section 2.3.1.18.

The seventh **SubResponse** element is the response for the **WhoAmI** subrequest. The **ErrorCode** attribute of "Success" indicates that the **WhoAmI** subrequest was successfully processed. The **SubResponseData** element specifies the requested client specific information in the **UserName**, **UserLogin**, **UserEmailAddress**, and **UserSIPAddress** attributes, which are specified in section 2.3.3.6.

4.2 Successful File Save of a Coauthorable Document

A client wants to save a file that has been edited back to the protocol server. This file is a coauthorable document. The client successfully saves the coauthorable document by sending a series of subrequests to initiate an upload of the file contents.

The protocol server is named Example.

The source file to be saved is http://Example/shared%20documents/test1.docx.

4.2.1 Request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <RequestVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
```

```
    <RequestCollection CorrelationId="{83E78EC0-5BAE-4BC2-9517-E2747382569B}"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <Request Url="http://Example/Shared%20Documents/test1.docx" RequestToken="1">
        <SubRequest Type="Coauth" SubRequestToken="1">
          <SubRequestData CoauthRequestType="RefreshCoauthoring" SchemaLockID=" 29358EC1-
E813-4793-8E70-ED0344E7B73C" ClientID="{BE07F85A-0CD1-4862-BDFC-F6CC3C8588A4}"
Timeout="3600"/>
        </SubRequest>
        <SubRequest Type="SchemaLock" SubRequestToken="2" DependsOn="1"
DependencyType="OnNotSupported">
          <SubRequestData SchemaLockRequestType="RefreshLock" SchemaLockID=" 29358EC1-E813-
4793-8E70-ED0344E7B73C" ClientID="{BE07F85A-0CD1-4862-BDFC-F6CC3C8588A4}" Timeout="3600"/>
        </SubRequest>
        <SubRequest Type="Cell" SubRequestToken="3" DependsOn="2"
DependencyType="OnSuccessOrNotSupported">
          <SubRequestData Coalesce="true" CoauthVersioning="true" BypassLockID=" 29358EC1-
E813-4793-8E70-ED0344E7B73C" SchemaLockID=" 29358EC1-E813-4793-8E70-ED0344E7B73C"
BinaryDataSize="17485">
            <i:Include xmlns:i="http://www.w3.org/2004/08/xop/include" href="cid:b2c67b53-
be27-4370-b214-6be0a48da399-0@tempuri.org"/>
          </SubRequestData>
        </SubRequest>
      </Request>
    </RequestCollection>
  </s:Body>
</s:Envelope>
```

The protocol client sends three **SubRequest** elements as part of the **Request** element in the cell storage service request message for uploading the file contents.

The first **SubRequest** element is a coauthoring subrequest of type "Refresh coauthoring session", which requests a refresh of the client's timeout of the shared lock and an update of the coauthoring status. Details about the coauthoring subrequest of type "Refresh coauthoring session" are specified in section 3.1.4.3.3.

The second **SubRequest** element is a schema lock subrequest of type "Refresh lock", which also requests a refresh of the client's timeout of the shared lock on the file. This schema lock subrequest is executed only if the first **SubRequest** element, the coauthoring subrequest, is not supported by the protocol server because the **DependencyType** attribute is set to "OnNotSupported". Details about the schema lock subrequest of type "Refresh lock" are specified in section 3.1.4.4.3.

The third **SubRequest** element is a cell subrequest that requests the upload of file contents or file metadata contents. Because the **DependencyType** attribute for this subrequest is set to "OnSuccessOrNotSupported", it is executed if either the coauthoring subrequest or the schema lock subrequest was successfully executed or if both these subrequests are not supported by the server. The **Coalesce** attribute of "true" requests that the protocol server persist all changes to the file with the underlying store. The **CoauthVersioning** attribute of "true" requests that the protocol server optimize the versioning of the file for coauthoring. Details about the **Coalesce** attribute and **CoauthVersioning** attribute are specified in section 2.3.3.1.

The **Include** element within the **SubRequestData** element of the cell subrequest is used for encapsulating and sending large amounts of binary data. Details about the **Include** element are specified in section 2.2.3.1. The cell subrequest for upload of file contents is processed as described in [MS-FSSHTTPB] section 3.1.4.3. Details about the cell subrequest are specified in section 3.1.4.2.

4.2.2 Response

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
```

```
    <ResponseVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
    <ResponseCollection WebUrl="http://Example"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <Response Url="http://Example/Shared%20Documents/test1.docx" RequestToken="1"
HealthScore="0">
        <SubResponse SubRequestToken="1" ErrorCode="Success" HResult="0">
          <SubResponseData LockType="SchemaLock" CoauthStatus="Alone"/>
        </SubResponse>
        91
        <SubResponse SubRequestToken="2"
ErrorCode="DependentOnlyOnNotSupportedRequestGetSupported" HResult="2147500037">
          <SubResponseData/>
        </SubResponse>
        1dd
        <SubResponse SubRequestToken="3" ErrorCode="Success" HResult="0">
          <SubResponseData Etag="&quot;{600CE272-068F-4BD7-A1FB-4AC10C54386C},2&quot;"
CoalesceHResult="0"
ContainsHotboxData="False">DAALAJ3PKfM5lAabFgMCAAAOAgYAAwsAhAAmAiAA9jV6MmEHFESWhlHpAGZ6TaQAeC
RNZ9PslLQ+v5Vxq4ReeFt+AMF4JLKYLBNrS8FAlXGrhF54W34At1ETASYCIAATHwkQgsj7QJiGZTP5NMIdbAFwLQz5C0E
3b9GZRKbDJyMu3KcRCTMAAAC1EwEmAiAADul2OjKADE253fPGUClDPkwBICYMspgsE2tLwUCVcauEXnhbfrcApRMBQQcB
iwE=</SubResponseData>
        </SubResponse>
        36
      </Response>
    </ResponseCollection>
  </s:Body>
</s:Envelope>
```


The first **SubResponse** element in this example is the response from the protocol server to the first subrequest from the protocol client, which is of type "Refresh coauthoring session". The **ErrorCode** attribute of "Success" indicates that the coauthoring subrequest was successfully processed, as specified in section 2.2.5.6. The **SubResponseData** element of the first **SubResponse** element specifies that the type of lock that was refreshed in the **LockType** attribute is "SchemaLock" and that the current coauthoring status in the **CoauthStatus** attribute is "Alone".

The second **SubResponse** element is the response to the schema lock subrequest of type "Refresh Lock".

The third **SubResponse** element is the response from the protocol server to the cell subrequest. The **ErrorCode** attribute of "Success" indicates that the cell subrequest for the uploading of file contents or file metadata contents was successfully processed. The other attributes and elements of this **SubResponse** element are described in section 4.2.1. The format of the value of the **SubResponseData** element is as described in [MS-FSSHTTPB] section 2.2.3.1.3.

4.3 Successful File Open of a Document that Is Not Coauthorable

A client wants to open a file on a protocol server. This file is not a coauthorable document. The client successfully opens the document by sending two subrequests to initiate a download of the file contents for exclusive editing.

The protocol server is named Example.

The source file to be opened is http://Example/shared%20documents/test2.xlsx.

4.3.1 Request

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
```

```
    <RequestVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
    <RequestCollection CorrelationId="{E07FCF1D-AD34-403F-9F77-D31B8225C8C5}"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <Request Url="http://Example/Shared%20Documents/test2.xlsx" RequestToken="1">
        <SubRequest Type="ExclusiveLock" SubRequestToken="1">
          <SubRequestData ExclusiveLockRequestType="GetLock" ExclusiveLockID="{9BCE3023-0F1F-
496B-A561-610144B54040}" Timeout="3600"/>
        </SubRequest>
        <SubRequest Type="Cell" SubRequestToken="2" DependsOn="1" DependencyType="OnExecute">
          <SubRequestData GetFileProps="true"
BinaryDataSize="88">DAALAJzPKfM5lAabBgIAAO4CAACqAiAAfrgx50XdqkSrgAx1+9FTDnoCCAD0J0UfdwEWAgYAA
wUAigICAADaAgYAAwAAygIIAAgAgAOEAEELAawCAFUDAQ==</SubRequestData>
        </SubRequest>
      </Request>
    </RequestCollection>
  </s:Body>
</s:Envelope>
```


The protocol client sends two **SubRequest** elements as part of the **Request** element in the cell storage service request message for opening the file contents.

The first **SubRequest** element is an exclusive lock subrequest of type "Get lock" that requests an exclusive lock on the file. The **ExclusiveLockID** attribute in the **SubRequestData** element specifies a unique identifier for the exclusive lock on the file. Details about the **ExclusiveLockID** attribute are specified in section 2.3.1.9. Details about the exclusive lock subrequest of type "Get lock" are specified in section 3.1.4.5.1.

The second **SubRequest** element is a cell subrequest that requests the download of file contents or file metadata contents and file properties. Details about the cell subrequest are specified in section 3.1.4.2.

4.3.2 Response


```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <ResponseVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
    <ResponseCollection WebUrl="http://Example"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <Response Url="http://Example/Shared%20Documents/test2.xlsx" RequestToken="1"
HealthScore="0">
        <SubResponse SubRequestToken="1" ErrorCode="Success" HResult="0">
          <SubResponseData/>
        </SubResponse>
        1ba
        <SubResponse SubRequestToken="2" ErrorCode="Success" HResult="0">
          <SubResponseData Etag="&quot;{901072DF-38C4-4211-BA16-0A882E036AB1},1&quot;"
CoalesceHResult="0" ContainsHotboxData="False" CreateTime="129092732530000000"
LastModifiedTime="129092732530000000" ModifiedBy="Jayne Darcy">
            <xop:Include href="cid:http%3A%2F%2Ftempuri.org%2F1%2F634003677517509709"
xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
          </SubResponseData>
        </SubResponse>
        36
      </Response>
    </ResponseCollection>
  </s:Body>
</s:Envelope>
```

The first **SubResponse** element in this example is the response from the protocol server to the exclusive lock subrequest of type "Get lock". The **ErrorCode** attribute of "Success" indicates that the coauthoring subrequest was successfully processed, as specified in section 2.2.5.6.

The second **SubResponse** element is the response from the protocol server for the cell subrequest. The **ErrorCode** of "Success" indicates that the cell subrequest for the downloading of file contents or file metadata contents and file properties was successfully processed. The **Include** element within the **SubResponseData** element of the response to the cell subrequest is used for encapsulating and sending large amounts of binary data. Details about the **Include** element are specified in section 2.2.3.1.

4.4 Unsuccessful File Open of a Document that Is Not Coauthorable

A client wants to open a file on the protocol server. This file is a not a coauthorable document. The client sends two subrequests to initiate a download of the file contents for exclusive editing. The client fails to open the document for exclusive editing because the file is already exclusively locked by another client.

The protocol server is named Example.

The source file to be opened is http://Example/shared%20documents/test2.xlsx.

4.4.1 Request


```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <RequestVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
    <RequestCollection CorrelationId="{930DC577-EB4B-455D-B186-CF31FDA04F0A}"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <Request Url="http://Example/Shared%20Documents/test2.xlsx" RequestToken="1">
        <SubRequest Type="ExclusiveLock" SubRequestToken="1">
          <SubRequestData ExclusiveLockRequestType="GetLock" ExclusiveLockID="{9BCE3023-0F1F-
496B-A561-610144B54040}" Timeout="3600"/>
        </SubRequest>
        <SubRequest Type="Cell" SubRequestToken="2" DependsOn="1" DependencyType="OnExecute">
          <SubRequestData GetFileProps="true"
BinaryDataSize="248">DAALAJzPKfM5lAabBgIAAO4CAACqAiAAfrgx50XdqkSrgAx1+9FTDnoCCAD0J0UfdwEWAgYA
AwUAigICAADaAgYAAwAAygIIAAgAgAOEACYCIAD2NXoyYQcURJaGUekAZnpNpAB4JET1xqOzzRdGqqho/p1URqkAr3gku
wo5XEwy6LmqqGj+nVRGqQC5URMBJgIgABMfCRCCyPtAmIZlM/k0wh1sAXAtDPkLQTdv0ZlEpsMnIy7cpxEJMwAAALUTAS
YCIAAO6XY6MoAMTbnd88ZQKUM+TAEgJgxE9cajs80XRqqoaP6dVEaprwClEwFBCwGsAgBVAwE=</SubRequestData>
        </SubRequest>
      </Request>
    </RequestCollection>
  </s:Body>
</s:Envelope>
```


The protocol client sends two **SubRequest** elements as part of the **Request** element in the cell storage service request message for opening the file contents.

The first **SubRequest** element is an exclusive lock subrequest of type "Get lock" that requests an exclusive lock on the file and provides an **ExclusiveLockID** of "{9BCE3023-0F1F-496B-A561-610144B54040}". Details about the exclusive lock subrequest of type "Get lock" are specified in section 3.1.4.5.1.

The second **SubRequest** element is a cell subrequest that requests the download of file contents or file metadata contents and file properties. Details about the cell subrequest are specified in section 3.1.4.2.

4.4.2 Response


```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <ResponseVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
    <ResponseCollection WebUrl="http://Example"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <Response Url="http://Example/Shared%20Documents/test2.xlsx" RequestToken="1"
HealthScore="0">
        <SubResponse SubRequestToken="1" ErrorCode="FileAlreadyLockedOnServer"
ErrorMessage="EXAMPLE\jdarcy" HResult="2147500037">
          <SubResponseData/>
        </SubResponse>
        29d
        <SubResponse SubRequestToken="2" ErrorCode="Success" HResult="0">
          <SubResponseData Etag="&quot;{901072DF-38C4-4211-BA16-0A882E036AB1},2&quot;"
CoalesceHResult="0" ContainsHotboxData="False" CreateTime="129092732530000000"
LastModifiedTime="129092735310000000" ModifiedBy="Jayne
Darcy">DAALAJ3PKfM5lAabFgMCAACsAgAMVgxpc1eRqqWLTIo5oB7fKT/AgPs2nIa+j3dFjynav98WLV0BAAAAAAAAAA
MFVQ4CBgADBQD6AiQADGlzV5GqpYtMijmgHt8pP8AAhAAmAiAA9jV6MmEHFESWhlHpAGZ6TaQAeCRE9cajs80XRqqoaP6
dVEapAK94JLsKOVxMMui5qqho/p1URqkAuVETASYCIAATHwkQgsj7QJiGZTP5NMIdbAFwLQz5C0E3b9GZRKbDJyMu3KcR
CTMAAAC1EwEmAiAADul2OjKADE253fPGUClDPkwBICYMRPXGo7PNF0aqqGj+nVRGqa8ApRMBQQcBiwE=</SubResponse
Data>
        </SubResponse>
        36
      </Response>
    </ResponseCollection>
  </s:Body>
</s:Envelope>
```


The first **SubResponse** element in this example is the response from the protocol server to the exclusive lock subrequest of type "Get lock". The **ErrorCode** attribute of "FileAlreadyLockedOnServer", defined in section 2.2.5.8, indicates that there is an existing exclusive lock on the targeted file or an existing schema lock on the targeted file but with a schema lock identifier different from the one provided in the client request. The **ErrorMessage** attribute of "EXAMPLE\jdarcy" specifies the identity of the user who is currently holding the lock on the file. The **HResult** attribute set to 2,147,500,037 specifies an error code specific to the exclusive lock subrequest that failed and gives more hints about the cause of the failure. Details about the **ErrorMessage** attribute and **HResult** attribute are specified in section 2.2.4.6.

Although the first subrequest failed, the second subrequest is executed by the server because the **DependencyType** attribute in the second subrequest is set to "OnExecute". The second **SubResponse** element is the response from the protocol server for the cell subrequest. The **ErrorCode** of "Success" indicates that the cell subrequest for the downloading of file contents and file properties was successfully processed. In this example, because the exclusive lock subrequest of type "Get lock" failed but the cell subrequest for download succeeded, the protocol client is allowed to open the file contents in read-only mode. The protocol client will not be allowed to make edits on the file.

4.5 Successful File Save of a Document that Is Not Coauthorable

A client wants to save a file that it has edited back to the protocol server. This file is a not a coauthorable document. The client successfully saves the document by sending two subrequests to initiate an upload of the file contents.

The protocol server is named Example.

The source file to be saved is http://Example/shared%20documents/test2.xlsx .

4.5.1 Request


```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <RequestVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
    <RequestCollection CorrelationId="{0E50BDEA-C991-495E-A574-B2BECAD97074}"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <Request Url="http://Example/Shared%20Documents/test2.xlsx" RequestToken="1">
        <SubRequest Type="ExclusiveLock" SubRequestToken="1">
          <SubRequestData ExclusiveLockRequestType="RefreshLock" ExclusiveLockID="{9BCE3023-
0F1F-496B-A561-610144B54040}" Timeout="3600"/>
        </SubRequest>
        <SubRequest Type="Cell" SubRequestToken="2" DependsOn="1"
DependencyType="OnSuccessOrNotSupported">
          <SubRequestData Coalesce="true" CoauthVersioning="true" BypassLockID="{9BCE3023-
0F1F-496B-A561-610144B54040}" BinaryDataSize="14972">
            <i:Include xmlns:i="http://www.w3.org/2004/08/xop/include" href="cid:4faa9924-
dbf5-4566-88a7-92c3adcce4fc-0@tempuri.org"/>
          </SubRequestData>
        </SubRequest>
      </Request>
    </RequestCollection>
  </s:Body>
</s:Envelope>
```


The protocol client sends two **SubRequest** elements as part of the **Request** element in the cell storage service request message for uploading the file contents.

The first **SubRequest** element is an exclusive lock subrequest of type "Refresh lock" that requests a refresh of the client's timeout of the exclusive lock on the file. Details about the exclusive lock subrequest of type "Refresh lock" are specified in section 3.1.4.5.3.

The second **SubRequest** element is a cell subrequest that requests the upload of file contents or file metadata contents. This cell subrequest is executed only if the first **SubRequest** element, the exclusive lock subrequest, succeeded or is not supported by the protocol server. The dependency of the cell subrequest on the first **SubRequest** element is defined by the **DependsOn** attribute and the **DependencyType** attribute of the cell subrequest. In this case, the **DependsOn** value of 1 specifies the **SubRequestToken** of the first **SubRequest** element, which the second element is dependent on. The **DependencyType** value of "OnSuccessOrNotSupported" specifies that the second cell subrequest gets called only if the first **SubRequest** element succeeded or is not supported, but not if it failed for a different reason. Details about the cell subrequest are specified in section 3.1.4.2.

4.5.2 Response


```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <ResponseVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
    <ResponseCollection WebUrl="http://Example"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <Response Url="http://Example/Shared%20Documents/test2.xlsx" RequestToken="1"
HealthScore="0">
        <SubResponse SubRequestToken="1" ErrorCode="Success" HResult="0">
          <SubResponseData/>
        </SubResponse>
        1dd
        <SubResponse SubRequestToken="2" ErrorCode="Success" HResult="0">
          <SubResponseData Etag="&quot;{901072DF-38C4-4211-BA16-0A882E036AB1},2&quot;"
CoalesceHResult="0"
ContainsHotboxData="False">DAALAJ3PKfM5lAabFgMCAAAOAgYAAwsAhAAmAiAA9jV6MmEHFESWhlHpAGZ6TaQAeC
RE9cajs80XRqqoaP6dVEapAK94JLsKOVxMMui5qqho/p1URqkAuVETASYCIAATHwkQgsj7QJiGZTP5NMIdbAFwLQz5C0E
```

```
3b9GZRKbDJyMu3KcRCTMAAAC1EwEmAiAADul2OjKADE253fPGUClDPkwBICYMRPXGo7PNF0aqqGj+nVRGqa8ApRMBQQcB
iwE=</SubResponseData>
        </SubResponse>
        36
      </Response>
    </ResponseCollection>
  </s:Body>
</s:Envelope>
```


The first **SubResponse** element in this example is the response from the protocol server to the exclusive lock subrequest of type "Refresh lock". The **ErrorCode** attribute of "Success" indicates that the exclusive lock subrequest was successfully processed, as specified in section 2.2.5.6.

The second **SubResponse** element is the response from the protocol server for the cell subrequest. The **ErrorCode** attribute of "Success" indicates that the cell subrequest for the uploading of file contents or file metadata contents was successfully processed. Other attributes and elements of this **SubResponse** are described in previous examples.

4.6 Unsuccessful File Open of a Coauthorable Document

A client wants to open a coauthorable document for coauthoring from a protocol server. The client fails to open the document for coauthoring because the number of coauthors has already reached the maximum. This section is almost identical to section 4.1, with the exception that the first subrequest ("Coauth") fails because the maximum number of coauthors has been reached.

The protocol server is named Example.

The source file to be opened is http://Example/Shared%20Documents/hello.docx.

4.6.1 Request


```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
    <s:Body>
        <RequestVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
        <RequestCollection CorrelationId="{006F42FA-D024-42C2-9A22-46BADDA853FD}"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
            <Request Url="http://Example/Shared%20Documents/hello.docx" RequestToken="1">
                <SubRequest Type="Coauth" SubRequestToken="1">
                    <SubRequestData CoauthRequestType="JoinCoauthoring"
SchemaLockID="29358EC1-E813-4793-8E70-ED0344E7B73C" ClientID="{09A77B2E-A135-480A-B4D4-
DD4E37F21E74}" Timeout="3600" AllowFallbackToExclusive="true" ExclusiveLockID="{09A77B2E-
A135-480A-B4D4-DD4E37F21E74}"/>
                </SubRequest>
                <SubRequest Type="SchemaLock" SubRequestToken="3" DependsOn="1"
DependencyType="OnNotSupported">
                    <SubRequestData SchemaLockRequestType="GetLock" SchemaLockID="29358EC1-
E813-4793-8E70-ED0344E7B73C" ClientID="{09A77B2E-A135-480A-B4D4-DD4E37F21E74}" Timeout="3600"
AllowFallbackToExclusive="true" ExclusiveLockID="{09A77B2E-A135-480A-B4D4-DD4E37F21E74}"/>
                </SubRequest>
                <SubRequest Type="Cell" SubRequestToken="6" DependsOn="3"
DependencyType="OnExecute">
                    <SubRequestData GetFileProps="true"
BinaryDataSize="88">DAALAJzPKfM5lAabBgIAAO4CAACqAiAAfrgx50XdqkSrgAx1+9FTDnoCCAB0OOUudwEWAgYAA
wUAigICAADaAgYAAwAAygIIAAgAgAOEAEELAawCAFUDAQ==</SubRequestData>
                </SubRequest>
                <SubRequest Type="Cell" SubRequestToken="5" DependsOn="3"
DependencyType="OnExecute">
                    <SubRequestData PartitionID="7808f4dd-2385-49d6-b7ce-37aca5e43602"
BinaryDataSize="88">DAALAJzPKfM5lAabBgIAAO4CAACqAiAAfrgx50XdqkSrgAx1+9FTDnoCCAB0OOUudwEWAgYAA
wUAigICAADaAgYAAwAAygIIAAgAgAOEAEELAawCAFUDAQ==</SubRequestData>
                </SubRequest>
```

```
                <SubRequest Type="Cell" SubRequestToken="4" DependsOn="3"
DependencyType="OnExecute">
                    <SubRequestData PartitionID="383adc0b-e66e-4438-95e6-e39ef9720122"
BinaryDataSize="88">DAALAJzPKfM5lAabBgIAAO4CAACqAiAAfrgx50XdqkSrgAx1+9FTDnoCCAB0OOUudwEWAgYAA
wUAigICAADaAgYAAwAAygIIAAgAgAOEAEELAawCAFUDAQ==</SubRequestData>
                </SubRequest>
                <SubRequest Type="WhoAmI" SubRequestToken="2"/>
                <SubRequest Type="ServerTime" SubRequestToken="7"/>
            </Request>
        </RequestCollection>
    </s:Body>
</s:Envelope>
```


The protocol client sends seven **SubRequest** elements as part of the **Request** element in the cell storage service request message for opening the document. The seven subrequests are identical to those in section 4.1.1.

4.6.2 Response


```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
    <s:Body>
        <ResponseVersion Version="2" MinorVersion="0"
xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
        <ResponseCollection WebUrl="http://Example"
xmlns="http://schemas.microsoft.com/sharepoint/soap/">
            <Response Url="http://Example/Shared%20Documents/hello.docx" RequestToken="1"
HealthScore="0">
                <SubResponse SubRequestToken="1" ErrorCode="NumberOfCoauthorsReachedMax"
HResult="2147500037"/>
                91
                <SubResponse SubRequestToken="3"
ErrorCode="DependentOnlyOnNotSupportedRequestGetSupported" HResult="2147500037">
                    <SubResponseData/>
                </SubResponse>
                1d4
                <SubResponse SubRequestToken="6" ErrorCode="Success" HResult="0">
                    <SubResponseData Etag="&quot;{18EA7896-87E5-4FF2-8B56-
D6932D37A920},2&quot;" CoalesceHResult="0" ContainsHotboxData="False"
HaveOnlyDemotionChanges="False" CreateTime="129217115870000000"
LastModifiedTime="129217115970000000" ModifiedBy="User"><xop:Include
href="cid:http%3A%2F%2Ftempuri.org%2F1%2F634128096252642638"
xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
                    </SubResponseData>
                </SubResponse>
                13d
                <SubResponse SubRequestToken="5" ErrorCode="Success" HResult="0">
                    <SubResponseData CoalesceHResult="0" ContainsHotboxData="True"
HaveOnlyDemotionChanges="False"><xop:Include
href="cid:http%3A%2F%2Ftempuri.org%2F2%2F634128096252722638"
xmlns:xop="http://www.w3.org/2004/08/xop/include"/>
                    </SubResponseData>
                </SubResponse>
                170
                <SubResponse SubRequestToken="4" ErrorCode="Success" HResult="0">


                        <SubResponseData CoalesceHResult="0" ContainsHotboxData="True"
HaveOnlyDemotionChanges="False">DAALAJ3PKfM5lAabFgMCAACsAgAMVgwiNmq4KENARZbfdZYMxE0zgNV311V
ftJZJo4z7N5ngOMcBAAAAAAAAAAOINACAAYc8L30CEECd7/lCoJY2CAEAAAAAAAAABVUOAgYAAwUA+gIkAAwiNmq4KE
NARZbfdZYMxE0zAIQAQQcBiwE=</SubResponseData>


                </SubResponse>
                e9
                <SubResponse SubRequestToken="2" ErrorCode="Success" HResult="0">
```

```
                    <SubResponseData UserName="The Automation Net Client"
UserLogin="DOMAIN\User" UserEmailAddress="user@example.com"
UserSIPAddress="user@example.com"/>
                </SubResponse>
                81
                <SubResponse SubRequestToken="7" ErrorCode="Success" HResult="0">
                    <SubResponseData ServerTime="634128600250000000"/>
                </SubResponse>
                36
            </Response>
        </ResponseCollection>
    </s:Body>
</s:Envelope>
```

With one exception, the **SubResponse** elements in the **Response** element are identical to those of section 4.1.2. The one exception is the **SubResponse** element for the "Coauth" subrequest. It shows that the "Coauth" subrequest failed with an **ErrorCode** of "NumberOfCoauthorsReachedMax", defined in section 2.2.5.8, indicating that no more coauthors are allowed because the maximum number of coauthors has already been reached on the server.

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

6 Appendix A: Full XML Schema

For ease of implementation, the following sections provide the full XML schemas for this protocol.

Schema name	Prefix	Section
Request message	None	6.1
Response message	None	6.2

6.1 Request Message Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/sharepoint/soap/"
attributeFormDefault="unqualified" elementFormDefault="qualified"
           targetNamespace=" http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:i="http://www.w3.org/2004/08/xop/include">
<xs:import namespace="http://www.w3.org/2004/08/xop/include" />
 <xs:element name="Envelope">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Body">
          <xs:complexType>
            <xs:sequence>
              <xs:element ref="tns:RequestVersion" minOccurs="1" maxOccurs="1" />
              <xs:element ref="tns:RequestCollection" minOccurs="1" maxOccurs="1"/>
             </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
 </xs:element>
</xs:schema>
```

The referenced child elements of the **Body** element are specified in the following schema:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/sharepoint/soap/"
attributeFormDefault="unqualified" elementFormDefault="qualified"
           targetNamespace="http://schemas.microsoft.com/sharepoint/soap/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:i="http://www.w3.org/2004/08/xop/include">
  <xs:import namespace="http://www.w3.org/2004/08/xop/include" />

  <!--common types between request and response schemas-->
  <!--
*******************************************************************************************
***********-->
  <!-- definition of simple types-->
  <xs:simpleType name="guid">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-
fA-F]{12}" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="LockTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="None" />
      <xs:enumeration value="SchemaLock" />
      <xs:enumeration value="ExclusiveLock" />
    </xs:restriction>
  </xs:simpleType>
```

```
  <xs:simpleType name="VersionNumberType">
    <xs:restriction base="xs:unsignedShort">
      <xs:minInclusive value="2"/>
      <xs:maxInclusive value="2"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="MinorVersionNumberType">
    <xs:restriction base="xs:unsignedShort">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="2"/>
    </xs:restriction>
  </xs:simpleType>

  <!-- definition of attributes-->

  <!-- definition of attribute groups-->

  <!--definition of complex types-->
  <xs:complexType name="VersionType">
    <xs:attribute name="Version" type="tns:VersionNumberType" use="required" />
    <xs:attribute name="MinorVersion" type="tns:MinorVersionNumberType" use="required" />
  </xs:complexType>

  <!--
*********************************************************************************************
******************-->
  <!-- definition of simple types-->
  <xs:simpleType name="CoauthRequestTypes">
    <xs:restriction base="xs:string">
      <!--JoinCoauthoring-->
      <xs:enumeration value="JoinCoauthoring"/>
      <!--ExitCoauthoring-->
      <xs:enumeration value="ExitCoauthoring"/>
      <!--RefreshCoauthoring-->
      <xs:enumeration value="RefreshCoauthoring"/>
      <!-- ConvertToExclusive-->
      <xs:enumeration value="ConvertToExclusive"/>
      <!--CheckLockAvailability-->
      <xs:enumeration value="CheckLockAvailability"/>
      <!--MarkTransitionComplete-->
      <xs:enumeration value="MarkTransitionComplete"/>
      <!-- GetCoauthoringStatus-->
      <xs:enumeration value="GetCoauthoringStatus"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="SchemaLockRequestTypes">
    <xs:restriction base="xs:string">
      <!--GetLock-->
      <xs:enumeration value="GetLock"/>
      <!--ReleaseLock-->
      <xs:enumeration value="ReleaseLock"/>
      <!--RefreshLock-->
      <xs:enumeration value="RefreshLock"/>
      <!--ConvertToExclusiveLock,-->
      <xs:enumeration value="ConvertToExclusive"/>
      <!--CheckLockAvailability-->
      <xs:enumeration value="CheckLockAvailability"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="ExclusiveLockRequestTypes">
    <xs:restriction base="xs:string">
      <!--GetLock-->
      <xs:enumeration value="GetLock"/>
      <!--ReleaseLock-->
```

```
        <xs:enumeration value="ReleaseLock"/>
        <!--RefreshLock-->
        <xs:enumeration value="RefreshLock"/>
        <!--ConvertToSchemaJoinCoauth-->
        <xs:enumeration value="ConvertToSchemaJoinCoauth"/>
        <!--ConvertToSchemaLock-->
        <xs:enumeration value="ConvertToSchema"/>
        <!--CheckLockAvailability-->
        <xs:enumeration value="CheckLockAvailability"/>
      </xs:restriction>
    </xs:simpleType>

    <xs:simpleType name="SubRequestAttributeType">
      <xs:restriction base="xs:string">
        <xs:enumeration value="Cell" />
        <xs:enumeration value="Coauth" />
        <xs:enumeration value="SchemaLock" />
        <xs:enumeration value="WhoAmI" />
        <xs:enumeration value="ServerTime" />
        <xs:enumeration value="ExclusiveLock" />
        <xs:enumeration value="GetDocMetaInfo" />
        <xs:enumeration value="GetVersions" />
        <xs:enumeration value="EditorsTable" />
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="DependencyTypes">
      <xs:restriction base="xs:string">
        <xs:enumeration value="OnExecute"/>
        <xs:enumeration value="OnSuccess"/>
        <xs:enumeration value="OnFail"/>
        <xs:enumeration value="OnNotSupported"/>
        <xs:enumeration value="OnSuccessOrNotSupported"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="EditorsTableRequestTypes">
      <xs:restriction base="xs:string">
        <!--JoinEditingSession-->
        <xs:enumeration value="JoinEditingSession"/>
        <!--LeaveEditingSession-->
        <xs:enumeration value="LeaveEditingSession"/>
        <!--RefreshEditingSession-->
        <xs:enumeration value="RefreshEditingSession"/>
        <!--UpdateEditorMetadata-->
        <xs:enumeration value="UpdateEditorMetadata"/>
        <!--RemoveEditorMetadata-->
       <xs:enumeration value="RemoveEditorMetadata"/>
     </xs:restriction>
    </xs:simpleType>


    <!-- definition of attributes-->

    <!-- definition of attribute groups-->
    <xs:attributeGroup name="SubRequestDataOptionalAttributes">
      <xs:attributeGroup ref="tns:CellSubRequestDataOptionalAttributes"/>
      <xs:attributeGroup ref="tns:CoauthSubRequestDataOptionalAttributes"/>
      <xs:attributeGroup ref="tns:SchemaLockSubRequestDataOptionalAttributes"/>
      <xs:attributeGroup ref="tns:ExclusiveLockSubRequestDataOptionalAttributes"/>
      <xs:attributeGroup ref="tns:EditorsTableSubRequestDataOptionalAttributes"/>
      <xs:attribute name="ClientID" type="xs:string" use="optional"/>
      <xs:attribute name="AllowFallbackToExclusive" type="xs:boolean" use="optional" />
      <xs:attribute name="ReleaseLockOnConversionToExclusiveFailure" type="xs:boolean"
  use="optional"/>
      <xs:attribute name="SchemaLockID" type="xs:string" use="optional" />
      <xs:attribute name="Timeout" type="xs:integer" use="optional" />
      <xs:attribute name="ExclusiveLockID" type="xs:string" use="optional"/>
      <xs:attribute name="BinaryDataSize" type="xs:long" use="optional" />
      <xs:attribute name="AsEditor" type="xs:boolean" use="optional" />
```

```
    <xs:attribute name="Key" type="xs:string" use="optional" />
    <xs:attribute name="Value" type="xs:binary" use="optional" />
  </xs:attributeGroup>

  <xs:attributeGroup name="CellSubRequestDataOptionalAttributes">
    <xs:attribute name="Coalesce" type="xs:boolean" use="optional" />
    <xs:attribute name="GetFileProps" type="xs:boolean" use="optional" />
    <xs:attribute name="CoauthVersioning" type="xs:boolean" use="optional" />
    <xs:attribute name="Etag" type="xs:string" use="optional" />
    <xs:attribute name="ContentChangeUnit" type="xs:string" use="optional" />

    <xs:attribute name="ClientFileID" type="xs:string" use="optional" />
    <xs:attribute name="PartitionID" type="tns:guid" use="optional" />
    <xs:attribute name="ExpectNoFileExists" type="xs:boolean" use="optional" />
    <xs:attribute name="BypassLockID" type="xs:string" use="optional" />
  </xs:attributeGroup>

  <xs:attributeGroup name="CoauthSubRequestDataOptionalAttributes">
    <xs:attribute name="CoauthRequestType" type="tns:CoauthRequestTypes" use="optional"/>
  </xs:attributeGroup>

  <xs:attributeGroup name="SchemaLockSubRequestDataOptionalAttributes">
    <xs:attribute name="SchemaLockRequestType" type="tns:SchemaLockRequestTypes"
use="optional"/>
  </xs:attributeGroup>

  <xs:attributeGroup name="ExclusiveLockSubRequestDataOptionalAttributes">
    <xs:attribute name="ExclusiveLockRequestType" type="tns:ExclusiveLockRequestTypes"
use="optional"/>
  </xs:attributeGroup>
  <xs:attributeGroup name="EditorsTableSubRequestDataOptionalAttributes">
    <xs:attribute name="EditorsTableRequestType" type="tns:EditorsTableRequestTypes"
use="optional"/>
  </xs:attributeGroup>

  <!--definition of complex types-->

  <xs:complexType name="SubRequestDataType">
    <xs:simpleContent>
      <xs:extension base="xs:string" />
    </xs:simpleContent>
  </xs:complexType>

  <xs:complexType name="CellSubRequestDataType" mixed="true">
    <xs:all>
      <xs:element ref="i:Include" minOccurs="0" maxOccurs="1" />
    </xs:all>
    <xs:attributeGroup ref="tns:CellSubRequestDataOptionalAttributes" />
    <xs:attribute name="SchemaLockID" type="xs:string" use="optional" />
    <xs:attribute name="ExclusiveLockID" type="xs:string" use="optional"/>
    <xs:attribute name="Timeout" type="xs:integer" use="optional" />
    <xs:attribute name="BinaryDataSize" type="xs:long" use="required" />
  </xs:complexType>

  <xs:complexType name="CoauthSubRequestDataType">
    <xs:attributeGroup ref="tns:CoauthSubRequestDataOptionalAttributes" />
    <xs:attribute name="ClientID" type="xs:string" use="required"/>
    <xs:attribute name="AllowFallbackToExclusive" type="xs:boolean" use="optional" />
    <xs:attribute name="ReleaseLockOnConversionToExclusiveFailure" type="xs:boolean"
use="optional"/>
    <xs:attribute name="SchemaLockID" type="xs:string" use="required" />
    <xs:attribute name="Timeout" type="xs:integer" use="optional" />
    <xs:attribute name="ExclusiveLockID" type="xs:string" use="optional"/>
  </xs:complexType>

  <xs:complexType name="SchemaLockSubRequestDataType">
    <xs:attributeGroup ref="tns:SchemaLockSubRequestDataOptionalAttributes"/>
    <xs:attribute name="ClientID" type="xs:string" use="optional"/>
```

```
    <xs:attribute name="AllowFallbackToExclusive" type="xs:boolean" use="optional" />
    <xs:attribute name="ReleaseLockOnConversionToExclusiveFailure" type="xs:boolean"
use="optional"/>
    <xs:attribute name="SchemaLockID" type="xs:string" use="required" />
    <xs:attribute name="Timeout" type="xs:integer" use="optional" />
    <xs:attribute name="ExclusiveLockID" type="xs:string" use="optional"/>
  </xs:complexType>

  <xs:complexType name="ExclusiveLockSubRequestDataType">
    <xs:attributeGroup ref="tns:ExclusiveLockSubRequestDataOptionalAttributes"/>
    <xs:attribute name="ClientID" type="xs:string" use="optional"/>
    <xs:attribute name="SchemaLockID" type="xs:string" use="optional" />
    <xs:attribute name="Timeout" type="xs:integer" use="optional" />
    <xs:attribute name="ExclusiveLockID" type="xs:string" use="required"/>
  </xs:complexType>

  <xs:complexType name="EditorsTableSubRequestDataType" mixed="true">
    <xs:attributeGroup ref="tns:EditorsTableSubRequestDataOptionalAttributes"/>
    <xs:attribute name="ClientID" type="xs:string" use="required"/>
    <xs:attribute name="AsEditor" type="xs:boolean" use="optional" />
    <xs:attribute name="Timeout" type="xs:integer" use="optional" />
    <xs:attribute name="Key" type="xs:string" use="optional" />
    <xs:attribute name="Value" type="xs:binary" use="optional" />
  </xs:complexType>

  <xs:complexType name="SubRequestDataGenericType" mixed="true">
    <xs:choice>
      <xs:element ref="i:Include" minOccurs="0" maxOccurs="1" />
    </xs:choice>
    <xs:attributeGroup ref="tns:SubRequestDataOptionalAttributes" />
  </xs:complexType>

  <xs:complexType name="SubRequestType">
    <xs:attribute name="SubRequestToken" type="xs:nonNegativeInteger" use="required" />
    <xs:attribute name="DependsOn" type="xs:nonNegativeInteger" use="optional" />
    <xs:attribute name="DependencyType" type="tns:DependencyTypes" use="optional" />
  </xs:complexType>

  <xs:complexType name="WhoAmISubRequestType">
    <xs:complexContent>
      <xs:extension base="tns:SubRequestType">
        <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="WhoAmI" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="ServerTimeSubRequestType">
    <xs:complexContent>
      <xs:extension base="tns:SubRequestType">
        <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="ServerTime" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="CellSubRequestType">
    <xs:complexContent>
      <xs:extension base="tns:SubRequestType">
        <xs:sequence minOccurs="0" maxOccurs="1">
          <xs:element name="SubRequestData" type="tns:CellSubRequestDataType" />
        </xs:sequence>
        <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="Cell" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

```
  <xs:complexType name="CoauthSubRequestType">
    <xs:complexContent>
      <xs:extension base="tns:SubRequestType">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="SubRequestData" type="tns:CoauthSubRequestDataType" />
        </xs:sequence>
        <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="Coauth" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="SchemaLockSubRequestType">
    <xs:complexContent>
      <xs:extension base="tns:SubRequestType">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="SubRequestData" type="tns:SchemaLockSubRequestDataType" />
        </xs:sequence>
        <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="SchemaLock" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="ExclusiveLockSubRequestType">
    <xs:complexContent>
      <xs:extension base="tns:SubRequestType">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="SubRequestData" type="tns:ExclusiveLockSubRequestDataType" />
        </xs:sequence>
        <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="ExclusiveLock" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="GetDocMetaInfoSubRequestType">
    <xs:complexContent>
      <xs:extension base="tns:SubRequestType">
        <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="GetDocMetaInfo" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>


  <xs:complexType name="GetVersionsSubRequestType">
    <xs:complexContent>
      <xs:extension base="tns:SubRequestType">
        <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="GetVersions" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>


<xs:complexType name="EditorsTableSubRequestType">
 <xs:complexContent>
  <xs:extension base="tns:SubRequestType">
   <xs:sequence minOccurs="1" maxOccurs="1">
     <xs:element name="SubRequestData" type="tns:EditorsTableSubRequestDataType" />
   </xs:sequence>
   <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="EditorsTable" />
  </xs:extension>
 </xs:complexContent>
</xs:complexType>
```

```
  <!--One SubrequestElement type that encapsulates the definition of all Subrequest types.  -
->
  <xs:complexType name="SubRequestElementGenericType" mixed="true">
    <xs:complexContent>
      <xs:extension base="tns:SubRequestType">
        <xs:all>
          <xs:element name="SubRequestData" minOccurs="0" maxOccurs="1"
type="tns:SubRequestDataGenericType" />
        </xs:all>
        <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required" />
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!--definition of simple elements-->

  <!-- definition of complex elements-->

  <!--definition of complex elements for Requests-->
  <xs:element name="RequestVersion" type="tns:VersionType" />

  <xs:element name="Request">
  <xs:complexType>
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:element name="SubRequest" type="tns:SubRequestElementGenericType" />
    </xs:sequence>
    <xs:attribute name="Url" type="xs:string" use="required"/>
    <xs:attribute name="Interval" type="xs:nonNegativeInteger" use="optional"/>
    <xs:attribute name="MetaData" type="xs:integer" use="optional"/>
    <xs:attribute name="RequestToken" type="xs:nonNegativeInteger" use="required" />
  </xs:complexType>
  </xs:element>

  <xs:element name="RequestCollection">
    <xs:complexType>
      <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element ref="tns:Request" />
      </xs:sequence>
      <xs:attribute name="CorrelationId" type="tns:guid" use="required" />
    </xs:complexType>
  </xs:element>

</xs:schema>
```

6.2 Response Message Schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/sharepoint/soap/"
attributeFormDefault="unqualified" elementFormDefault="qualified"
           targetNamespace=" http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:i="http://www.w3.org/2004/08/xop/include">
<xs:import namespace="http://www.w3.org/2004/08/xop/include" />

<xs:element name="Envelope">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="Body">
    <xs:complexType>
     <xs:sequence>
      <xs:element ref="tns:ResponseVersion" minOccurs="1" maxOccurs="1" />
      <xs:element ref="tns:ResponseCollection" minOccurs="0" maxOccurs="1"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
```

```
 </xs:complexType>
</xs:element>
</xs:schema>
```

The referenced child elements of the **Body** element are specified in the following schema:

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/sharepoint/soap/"
attributeFormDefault="unqualified" elementFormDefault="qualified"
           targetNamespace="http://schemas.microsoft.com/sharepoint/soap/"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:i="http://www.w3.org/2004/08/xop/include">
  <xs:import namespace="http://www.w3.org/2004/08/xop/include" />

  <!--common datatypes between Cell storage service request and response schemas-->
  <!-- definition of simple types-->
  <xs:simpleType name="guid">
    <xs:restriction base="xs:string">
      <xs:pattern value="[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-
fA-F]{12}" />
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="TRUEFALSE">
    <xs:restriction base="xs:string">
      <xs:pattern value="[Tt][Rr][Uu][Ee]|[Ff][Aa][Ll][Ss][Ee]"/>
      </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="LockTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="None" />
      <xs:enumeration value="SchemaLock" />
      <xs:enumeration value="ExclusiveLock" />
    </xs:restriction>
  </xs:simpleType>


  <xs:simpleType name="VersionNumberType">
    <xs:restriction base="xs:unsignedShort">
      <xs:minInclusive value="2"/>
      <xs:maxInclusive value="2"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="MinorVersionNumberType">
    <xs:restriction base="xs:unsignedShort">
      <xs:minInclusive value="0"/>
      <xs:maxInclusive value="2"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="ExclusiveLockReturnReasonTypes">
   <xs:restriction base="xs:string">
      <xs:enumeration value="CoauthoringDisabled" />
      <xs:enumeration value="CheckedOutByCurrentUser" />
      <xs:enumeration value="CurrentUserHasExclusiveLock" />
   </xs:restriction>
  </xs:simpleType>

  <!-- definition of attributes-->

  <!-- definition of attribute groups-->

  <!--definition of complex types-->
  <xs:complexType name="VersionType">
```

```
    <xs:attribute name="Version" type="tns:VersionNumberType" use="required" />
    <xs:attribute name="MinorVersion" type="tns:MinorVersionNumberType" use="required" />
  </xs:complexType>
  <!--
*****************************************************************************************-->

  <!--definition of simple types-->
  <xs:simpleType name="ErrorCodeTypes">
    <xs:union memberTypes="tns:GenericErrorCodeTypes
      tns:CellRequestErrorCodeTypes tns:DependencyCheckRelatedErrorCodeTypes
tns:LockAndCoauthRelatedErrorCodeTypes tns:NewEditorsTableCategoryErrorCodeTypes"/>
  </xs:simpleType>

  <xs:simpleType name="GenericErrorCodeTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Success"/>
      <xs:enumeration value="IncompatibleVersion"/>
      <xs:enumeration value="InvalidUrl"/>
      <xs:enumeration value="FileNotExistsOrCannotBeCreated"/>
      <xs:enumeration value="FileUnauthorizedAccess"/>
      <xs:enumeration value="InvalidSubRequest"/>
      <xs:enumeration value="SubRequestFail"/>
      <xs:enumeration value="BlockedFileType"/>
      <xs:enumeration value="DocumentCheckoutRequired"/>
      <xs:enumeration value="InvalidArgument"/>
      <xs:enumeration value="RequestNotSupported"/>
      <xs:enumeration value="InvalidWebUrl"/>
      <xs:enumeration value="WebServiceTurnedOff"/>
      <xs:enumeration value="ColdStoreConcurrencyViolation"/>
      <xs:enumeration value="HighLevelExceptionThrown"/>
      <xs:enumeration value="Unknown"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="CellRequestErrorCodeTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="CellRequestFail"/>
      <xs:enumeration value="IRMDocLibarysOnlySupportWebDAV"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="DependencyCheckRelatedErrorCodeTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="DependentRequestNotExecuted"/>
      <xs:enumeration value="DependentOnlyOnSuccessRequestFailed"/>
      <xs:enumeration value="DependentOnlyOnFailRequestSucceeded"/>
      <xs:enumeration value="DependentOnlyOnNotSupportedRequestGetSupported"/>
      <xs:enumeration value="InvalidRequestDependencyType"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="LockAndCoauthRelatedErrorCodeTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="LockRequestFail"/>
      <xs:enumeration value="FileAlreadyLockedOnServer"/>
      <xs:enumeration value="FileNotLockedOnServer"/>
      <xs:enumeration value="FileNotLockedOnServerAsCoauthDisabled"/>
      <xs:enumeration value="LockNotConvertedAsCoauthDisabled"/>
      <xs:enumeration value="FileAlreadyCheckedOutOnServer"/>
      <xs:enumeration value="ConvertToSchemaFailedFileCheckedOutByCurrentUser"/>
      <xs:enumeration value="CoauthRefblobConcurrencyViolation"/>
      <xs:enumeration value="MultipleClientsInCoauthSession"/>
      <xs:enumeration value="InvalidCoauthSession"/>
      <xs:enumeration value="NumberOfCoauthorsReachedMax"/>
      <xs:enumeration value="ExitCoauthSessionAsConvertToExclusiveFailed"/>
    </xs:restriction>
  </xs:simpleType>
```

```
  <xs:simpleType name="NewEditorsTableCategoryErrorCodeTypes">
     <xs:restriction base="xs:string">
        <xs:enumeration value="EditorMetadataQuotaReached"/>
        <xs:enumeration value="EditorMetadataStringExceedsLengthLimit"/>
        <xs:enumeration value="EditorClientIdNotFound"/>
     </xs:restriction>
  </xs:simpleType>


  <xs:simpleType name="CoauthStatusType">
    <xs:restriction base="xs:string">
      <!--None-->
      <xs:enumeration value="None"/>
      <!--Alone -->
      <xs:enumeration value="Alone"/>
      <!--Coauthoring-->
      <xs:enumeration value="Coauthoring"/>
    </xs:restriction>
  </xs:simpleType>



  <xs:simpleType name="UserNameType">
    <xs:restriction base="xs:string">
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="UserLoginType">
    <xs:restriction base="xs:string">
    </xs:restriction>
  </xs:simpleType>

  <!--definition of attribute groups-->
  <xs:attributeGroup name="SubResponseDataOptionalAttributes">
    <xs:attributeGroup ref="tns:CellSubResponseDataOptionalAttributes"/>
    <xs:attributeGroup ref="tns:WhoAmISubResponseDataOptionalAttributes"/>
    <xs:attribute name="ServerTime" type="xs:positiveInteger" use="optional"/>
    <xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
    <xs:attribute name="CoauthStatus" type="tns:CoauthStatusType" use="optional"/>
    <xs:attribute name="TransitionID" type="tns:guid" use="optional"/>
    <xs:attribute name="ExclusiveLockReturnReason" type="tns:ExclusiveLockReturnReasonTypes"
use="optional" />
   </xs:attributeGroup>

  <xs:attributeGroup name="CellSubResponseDataOptionalAttributes">
    <xs:attribute name="Etag" type="xs:string" use="optional" />
    <xs:attribute name="CreateTime" type="xs:integer" use="optional"/>
    <xs:attribute name="LastModifiedTime" type="xs:integer" use="optional"/>
    <xs:attribute name="ModifiedBy" type="tns:UserNameType" use="optional" />
    <xs:attribute name="CoalesceErrorMessage" type="xs:string" use="optional"/>
    <xs:attribute name="CoalesceHResult" type="xs:integer" use="optional"/>
    <xs:attribute name="ContainsHotboxData" type="tns:TRUEFALSE" use="optional"/>
    <xs:attribute name="HaveOnlyDemotionChanges" type="tns:TRUEFALSE" use="optional"/>
  </xs:attributeGroup>

  <xs:attributeGroup name="WhoAmISubResponseDataOptionalAttributes">
    <xs:attribute name="UserName" type="tns:UserNameType" use="optional"/>
    <xs:attribute name="UserEmailAddress" type="xs:string" use="optional"/>
    <xs:attribute name="UserSIPAddress" type="xs:string" use="optional" />
    <xs:attribute name="UserIsAnonymous" type="xs:boolean" use="optional" />
    <xs:attribute name="UserLogin" type="tns:UserLoginType" use="required"/>
  </xs:attributeGroup>

  <!--definition of complex types-->
  <xs:complexType name="CellSubResponseDataType" mixed="true">
    <xs:all>
      <xs:element ref="i:Include" minOccurs="0" maxOccurs="1" />
    </xs:all>
```

```
    <xs:attributeGroup ref="tns:CellSubResponseDataOptionalAttributes" />

    <xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
       </xs:complexType>

  <!--There is no text in this element-->
  <xs:complexType name="CoauthSubResponseDataType">
    <xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
    <xs:attribute name="CoauthStatus" type="tns:CoauthStatusType" use="optional"/>
    <xs:attribute name="TransitionID" type="tns:guid" use="optional"/>
    <xs:attribute name="ExclusiveLockReturnReason" type="tns:ExclusiveLockReturnReasonTypes"
use="optional" />
  </xs:complexType>

  <xs:complexType name="SchemaLockSubResponseDataType">
    <xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
    <xs:attribute name="ExclusiveLockReturnReason" type="tns:ExclusiveLockReturnReasonTypes"
use="optional" />
  </xs:complexType>

  <xs:complexType name="ExclusiveLockSubResponseDataType">
    <xs:attribute name="CoauthStatus" type="tns:CoauthStatusType" use="optional"/>
    <xs:attribute name="TransitionID" type="tns:guid" use="optional"/>
      </xs:complexType>

  <xs:complexType name="WhoAmISubResponseDataType">
    <xs:attributeGroup ref="tns:WhoAmISubResponseDataOptionalAttributes"/>

  </xs:complexType>

  <xs:complexType name="ServerTimeSubResponseDataType">
    <xs:attribute name="ServerTime" type="xs:positiveInteger" use="optional"/>
  </xs:complexType>

  <xs:complexType name="GetDocMetaInfoSubResponseDataType">
    <xs:sequence>
      <xs:element name="DocProps" type="tns:GetDocMetaInfoPropertySetType"/>
      <xs:element name="FolderProps" type="tns:GetDocMetaInfoPropertySetType"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="GetDocMetaInfoPropertyType">
      <xs:attribute name="Key" type="xs:string" use="required"/>
      <xs:attribute name="Value" type="xs:string" use="required"/>
  </xs:complexType>

  <xs:complexType name="GetDocMetaInfoPropertySetType">
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="Property" type="tns:GetDocMetaInfoPropertyType"/>
      </xs:sequence>
  </xs:complexType>

  <xs:complexType name="SubResponseDataGenericType" mixed="true">
     <xs:all>
      <xs:element ref="i:Include" minOccurs="0" maxOccurs="1" />
      <xs:element name="DocProps" minOccurs="0" maxOccurs="1"
type="tns:GetDocMetaInfoPropertySetType"/>
      <xs:element name="FolderProps" minOccurs="0" maxOccurs="1"
type="tns:GetDocMetaInfoPropertySetType"/>
     </xs:all>
     <xs:attributeGroup ref="tns:SubResponseDataOptionalAttributes" />
  </xs:complexType>

  <xs:complexType name="SubResponseType">
    <xs:attribute name="SubRequestToken" type="xs:nonNegativeInteger" use="required" />
    <xs:attribute name="ErrorCode" type="tns:ErrorCodeTypes" use="required" />
    <xs:attribute name="HResult" type="xs:integer" use="required" />
    <xs:attribute name="ErrorMessage" type="xs:string" use="optional"/>
```

```
  </xs:complexType>

  <xs:complexType name="WhoAmISubResponseType">
    <xs:complexContent>
      <xs:extension base="tns:SubResponseType">
        <xs:sequence minOccurs="0" maxOccurs="1">
          <xs:element name="SubResponseData" type="tns:WhoAmISubResponseDataType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="ServerTimeSubResponseType">
    <xs:complexContent>
      <xs:extension base="tns:SubResponseType">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="SubResponseData" type="tns:ServerTimeSubResponseDataType"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="CellSubResponseType">
    <xs:complexContent>
      <xs:extension base="tns:SubResponseType">
        <xs:sequence>
          <xs:element name="SubResponseData" type="tns:CellSubResponseDataType" minOccurs="0"
maxOccurs="1"/>
          <xs:element name="SubResponseStreamInvalid" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="CoauthSubResponseType">
    <xs:complexContent>
      <xs:extension base="tns:SubResponseType">
        <xs:sequence minOccurs="0" maxOccurs="1">
          <xs:element name="SubResponseData" type="tns:CoauthSubResponseDataType" />
        </xs:sequence>
       </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="SchemaLockSubResponseType">
    <xs:complexContent>
      <xs:extension base="tns:SubResponseType">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="SubResponseData" type="tns:SchemaLockSubResponseDataType" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="ExclusiveLockSubResponseType">
    <xs:complexContent>
      <xs:extension base="tns:SubResponseType">
        <xs:sequence minOccurs="1" maxOccurs="1">
          <xs:element name="SubResponseData" type="tns:ExclusiveLockSubResponseDataType" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="GetDocMetaInfoSubResponseType">
    <xs:complexContent>
      <xs:extension base="tns:SubResponseType">
        <xs:sequence minOccurs="0" maxOccurs="1">
```

```
            <xs:element name="SubResponseData" type="tns:GetDocMetaInfoSubResponseDataType"/>


          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="GetVersionsSubResponseType">
      <xs:complexContent>
        <xs:extension base="tns:SubResponseType">
          <xs:sequence minOccurs="0" maxOccurs="1">
             <xs:element ref="tns:GetVersionsResponse"/>
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  <xs:element name="GetVersionsResponse">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="1" maxOccurs="1" name="GetVersionsResult">
          <xs:complexType>
            <xs:sequence>
              <xs:element name="results" minOccurs="1" maxOccurs="1" type="tns:Results" />
            </xs:sequence>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>

  <xs:complexType name="Results">
    <xs:sequence>
      <xs:element name="list" maxOccurs="1" minOccurs="1">
        <xs:complexType>
          <xs:attribute name="id" type="xs:string" use="required" />
        </xs:complexType>
      </xs:element>
      <xs:element name="versioning" maxOccurs="1" minOccurs="1">
        <xs:complexType>
          <xs:attribute name="enabled" type="xs:unsignedByte" use="required" />
        </xs:complexType>
      </xs:element>
      <xs:element name="settings" maxOccurs="1" minOccurs="1">
        <xs:complexType>
          <xs:attribute name="url" type="xs:string" use="required" />
        </xs:complexType>
      </xs:element>
      <xs:element name="result" maxOccurs="unbounded" minOccurs="1" type="tns:VersionData"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="VersionData">
    <xs:attribute name="version" type="xs:string" use="required" />
    <xs:attribute name="url" type="xs:string" use="required" />
    <xs:attribute name="created" type="xs:string" use="required" />
    <xs:attribute name="createdRaw" type="xs:string" use="required" />
    <xs:attribute name="createdBy" type="xs:string" use="required" />
    <xs:attribute name="createdByName" type="xs:string" use="optional" />
    <xs:attribute name="size" type="xs:unsignedLong" use="required" />
    <xs:attribute name="comments" type="xs:string" use="required" />
  </xs:complexType>




    <xs:complexType name="EditorsTableSubResponseType">
      <xs:complexContent>
```

```
      <xs:extension base="tns:SubResponseType">
      <xs:sequence minOccurs="0" maxOccurs="1">
      <xs:element name="SubResponseData">
      <xs:complexType>
      <xs:complexContent>
      <xs:restriction base="xs:anyType"/>
      </xs:complexContent>
      </xs:complexType>
      </xs:element>
      </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!--One SubrequestElement type that encapsulates the defintion of all Subrequest types. -->
  <xs:complexType name="SubResponseElementGenericType">
    <xs:complexContent>
      <xs:extension base="tns:SubResponseType">
        <xs:sequence>
          <xs:element name="SubResponseData" minOccurs="0" maxOccurs="1"
type="tns:SubResponseDataGenericType" />
          <xs:element name="SubResponseStreamInvalid" minOccurs="0" maxOccurs="1" />
          <xs:element ref="GetVersionsResponse" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <!--definition of simple elements-->

  <!-- definition of complex elements-->

  <!--definition of complex elements for Responses-->
  <xs:element name="ResponseVersion">
    <xs:complexType>
    <xs:complexContent>
      <xs:extension base="tns:VersionType">
        <xs:attribute name="ErrorCode" type="tns:GenericErrorCodeTypes" use="optional" />
        <xs:attribute name="ErrorMessage" type="xs:string" use="optional" />
      </xs:extension>
    </xs:complexContent>
    </xs:complexType>
  </xs:element>

  <xs:element name="Response">
  <!--Allows for the numbers to be displayed between the SubResponse elements-->
  <xs:complexType mixed="true">
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:element name="SubResponse" type="tns:SubResponseElementGenericType" />
    </xs:sequence>
    <xs:attribute name="Url" type="xs:string" use="required"/>
    <xs:attribute name="RequestToken" type="xs:nonNegativeInteger" use="required" />
    <xs:attribute name="HealthScore" type="xs:integer" use="required"/>
    <xs:attribute name="ErrorCode" type="tns:GenericErrorCodeTypes" use="optional" />
    <xs:attribute name="ErrorMessage" type="xs:string" use="optional"/>
  </xs:complexType>
  </xs:element>

  <xs:element name="ResponseCollection">
    <xs:complexType>
      <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element ref="tns:Response" />
      </xs:sequence>
      <xs:attribute name="WebUrl" type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
```

```
</xs:schema>
```

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 Office 2013
- Microsoft SharePoint Foundation 2010
- Microsoft SharePoint Foundation 2013
- Microsoft SharePoint Server 2010
- Microsoft SharePoint Server 2013
- Microsoft SharePoint Workspace 2010
- Windows 8.1 Update
- Microsoft Office 2016 Preview

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 2.2.3.2: Attribute not supported by s.

<2> Section 2.2.3.2: Attribute not supported by Office 2010.

<3> Section 2.2.3.5: Microsoft SharePoint Server 2010 will return 2 **ErrorCode** attributes in **Response** element. SharePoint Server 2013 will not return **Response** element.

<4> Section 2.2.3.5: The **RequestToken** attribute is ignored by SharePoint Foundation 2013 and SharePoint Server 2013.

<5> Section 2.2.5.5: SharePoint Server 2010 and SharePoint Server 2013 will never return this value to the client.

<6> Section 2.2.5.6: SharePoint Server 2010 does not support this error code type.

<7> Section 2.2.5.14: SharePoint Server 2010 does not support this simple type.

<8> Section 2.2.8.1: Microsoft Word 2010 and Microsoft PowerPoint 2010 use the string "29358EC1-E813-4793-8E70-ED0344E7B73C" for the **SchemaLockID** attribute sent in the different subrequests of type **Coauthoring**, **ExclusiveLock**, and **SchemaLock**.

<9> Section 2.2.8.2: Servers running Microsoft Office 2010 suites return a time value that is not the current time. The protocol client does not change its behavior because the protocol client uses the difference between **ServerTime** values, not the difference between **ServerTime** and the time at the client.

<10> Section 2.2.8.2: SharePoint Server 2010 will not return **CoauthStatus** attribute when client tries to join the coauthoring session and the subrequest falls back to an exclusive lock subrequest.

<11> Section 2.3.1.1: Word 2010 and PowerPoint 2010 use string "29358EC1-E813-4793-8E70-ED0344E7B73C" for the **SchemaLockID** attribute sent in the different subrequests of type **Coauthoring**, **ExclusiveLock** and **SchemaLock**.

<12> Section 2.3.1.5: In SharePoint Server 2013, if the **CoauthRequestType** attribute is not provided, a "HighLevelExceptionThrown" error code MUST be returned as part of the **ResponseVersion** element.

<13> Section 2.3.1.5: Word 2010 and PowerPoint 2010 use the string "29358EC1-E813-4793-8E70-ED0344E7B73C" for the **SchemaLockID** attribute sent in the different subrequests of type **Coauthoring**, **ExclusiveLock**, and **SchemaLock**.

<14> Section 2.3.1.7: SharePoint Server 2010 will not return **CoauthStatus** attribute when client tries to join the coauthoring session and the subrequest falls back to an exclusive lock subrequest.

<15> Section 2.3.1.9: In SharePoint Server 2013, if the **ExclusiveLockRequestType** attribute is not provided, a "HighLevelExceptionThrown" error code MUST be returned as part of the **ResponseVersion** element.

<16> Section 2.3.1.9: Word 2010 and PowerPoint 2010 use the string "29358EC1-E813-4793-8E70-ED0344E7B73C" for the **SchemaLockID** attribute sent in the different subrequests of type **Coauthoring**, **ExclusiveLock**, and **SchemaLock**.

<17> Section 2.3.1.13: In SharePoint Server 2013, if the **SchemaLockRequestType** attributes is not provided, a "HighLevelExceptionThrown" error code MUST be returned as part of the **ResponseVersion** element.

<18> Section 2.3.1.13: Word 2010 and PowerPoint 2010 use string "29358EC1-E813-4793-8E70-ED0344E7B73C" for the **SchemaLockID** attribute sent in the different subrequests of type **Coauthoring**, **ExclusiveLock**, and **SchemaLock**.

<19> Section 2.3.1.18: Servers running Office 2010 return a time value that is not the current time. The protocol client does not change its behavior because the protocol client uses the difference between **ServerTime** values, not the difference between the **ServerTime** and the time at the client.

<20> Section 2.3.2.6: There is an additional authentication prefix if **claim-based authentication mode** is enabled. For example: " i:0#.w|contoso\user01".

<21> Section 2.3.3.1: Servers running Office 2010 save all changes to the underlying store at the end of processing a subrequest, with the following four exceptions:

1. The subrequest is a **Put Changes** subrequest, as described in [MS-FSSHTTPB] section 2.2.2.1.4, and the **Partial** bit, as described in [MS-FSSHTTPB] section 2.2.2.1.4, of the **Put Changes** request is 1. In this case, servers running Office 2010 do not save changes to the underlying store. Instead, the changes are stored in a write cache.

2. Servers running Office 2010 write to an intermediate write cache while processing a subrequest if the subrequest is a **Put Changes** subrequest, as described in [MS-FSSHTTPB] section 2.2.2.1.4. If writing to the write cache fails, servers running Office 2010 do not save changes to the underlying store.

3. If prior to the current attempt to save changes to the underlying store, there have been from 26 through 73 consecutive failed attempts immediately preceding the current attempt, and the most recent failed attempt is within 60 minutes of the current subrequest, servers running Office 2010 do not save changes to the underlying store. Instead, the changes are stored in a write cache.

4. If prior to the current attempt to save changes to the underlying store there have been more than 73 consecutive failed attempts immediately preceding and the most recent failed attempt is within 24 hours of the current subrequest then servers running Office 2010 will not save changes to the underlying store. Instead the changes will be stored in a write cache.

<22> Section 2.3.3.1: Attribute not supported by SharePoint Foundation 2013 and SharePoint Server 2013 in **Put Changes** subrequest, as described in [MS-FSSHTTPB] section 2.2.1.1.4.

<23> Section 2.3.3.2: Attribute not supported by SharePoint Server 2013.

<24> Section 2.3.3.2: Attribute not supported by SharePoint Server 2013.

<25> Section 2.3.3.6: Attribute not supported by Office 2010.

<26> Section 3.1.4.1: Office 2010 applications set the lock timeout interval to 3,600 seconds and send a request to refresh the timeout at a regular interval of 2,700 seconds.

<27> Section 3.1.4.3.1: SharePoint Foundation 2013 and SharePoint Server 2013 will return an error code value "FileNotLockedOnServer" if the **AllowFallbackToExclusive** attribute is set to false.

<28> Section 3.1.4.3.2: SharePoint Server 2013 will return an error code of "InvalidCoauthSession" if there are other clients present in the coauthoring session.

<29> Section 3.1.4.3.2: SharePoint Server 2013 and SharePoint Server 2010 return an error code of "Success" if there is an exclusive lock on the file or if there is a shared lock with a different shared lock identifier and a valid coauthoring session containing more than one clients. SharePoint Server 2013 and SharePoint Server 2010 return an error code of "FileAlreadyLockedOnServer" if there is a shared lock with a different shared lock identifier and a coauthoring session containing one client.

<30> Section 3.1.4.3.3: SharePoint Foundation 2013 and SharePoint Server 2013 will return an error code value "FileNotLockedOnServer".

<31> Section 3.1.4.3.6: SharePoint Server 2010 returns an error code value set to "LockRequestFail" if there is no shared lock.

<32> Section 3.1.4.3.6: SharePoint Server 2010 returns an error code value set to "LockRequestFail" if there is no coauthoring session for the file.

<33> Section 3.1.4.3.7: In SharePoint Server 2013, the protocol server returns an error code value set to "Success".

<34> Section 3.1.4.4.1: SharePoint Server 2013 will return an error code value "FileNotLockedOnServer" if the **AllowFallbackToExclusive** attribute is set to false.

<35> Section 3.1.4.4.2: SharePoint Server 2010 will return an error code of "Success" if there are other clients present in the coauthoring session.

<36> Section 3.1.4.4.2: SharePoint Server 2013 and SharePoint Server 2010 return error code "Success" if there is an exclusive lock on the file or there is a shared lock with a different shared lock identifier and valid coauthoring session with more than one clients in it. SharePoint Server 2013 and SharePoint Server 2010 return an error code "FileAlreadyLockedOnServer" if there is a shared lock with a different shared lock identifier and a coauthoring session with one client in it.

<37> Section 3.1.4.4.3: SharePoint Foundation 2013 and SharePoint Server 2013 will return an error code value "FileNotLockedOnServer".

<38> Section 3.1.4.7: Servers running Office 2010 return a time value that is not the current time. The protocol client does not change its behavior because the protocol client uses the difference between **ServerTime** values, not the difference between **ServerTime** and the time at the client.

<39> Section 3.1.4.8: SharePoint Server 2010 does not support this operation.

<40> Section 3.1.4.8: Servers running Office 2013 automatically add a client to the editors table when it takes a coauthoring or schema lock—if the client protocol version is 2.2 or higher as described in section 2.2.5.10. Editors table subrequests are used by clients running Office 2013 that do not take locks on documents before editing them.

<41> Section 3.1.4.8: Only 4 key/value pairs can be associated with an editor on servers running Office 2013.

<42> Section 3.1.4.8: On servers running Office 2013, the editors table partition identifier is the GUID "7808F4DD-2385-49d6-B7CE-37AC-A5E4-3602", and the editors table is represented as a compressed XML fragment.

<43> Section 3.1.4.9: SharePoint Server 2010 does not support this operation.

<44> Section 3.1.4.10: SharePoint Server 2010 does not support this operation.

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
7 Appendix B: Product Behavior	Updated list of supported products.	Y	Content updated due to protocol revision.

9 Index

F

G

I

L

M

N

O

P

R

S

T

U

V

W

X