[MS-FSSHTTP]:
File Synchronization via SOAP over HTTP Protocol Specification

---

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

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

Table of Contents

1 Introduction

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

1.1 Glossary

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

Coordinated Universal Time (UTC)
GUID
HRESULT
Hypertext Transfer Protocol (HTTP)

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

absolute URL
friendly name
Information Rights Management (IRM)
Session Initiation Protocol (SIP) address
Simple Object Access Protocol (SOAP)
SOAP fault
Uniform Resource Locator (URL)
XML namespace
XML namespace prefix
XML schema

The following terms are specific to this document:

chunked transfer encoding: A mechanism that enables an HTTP server to transmit data to a protocol client as multiple parts, as described in [RFC2616].

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

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].

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].

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

1.2 References

1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact dochelp@microsoft.com. We will assist you in finding the relevant information. Please check the archive site, http://msdn2.microsoft.com/en-us/library/E4BD6494-06AD-4aed-9823-445E921C9624, as an additional source.

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

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

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

[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.ietf.org/rfc/rfc2616.txt

[RFC2822] Resnick, P., Ed., "Internet Message Format", STD 11, 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] World Wide Web Consortium, "XML Information Set (Second Edition)", 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.S., Ed., Beech, D., Ed., Maloney, M., Ed., and Mendelsohn, N., Ed., "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

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

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

[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

1.3 Protocol Overview (Synopsis)

This protocol enables a protocol client to call request that allows for 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 information about the different types of locking operations, see section 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 a Uniform Resource Locator (**URL**). Each file has one or more partitions associated with it. These partitions can be empty, contain binary file contents, file co-authoring related information or file format specific contents. Data in each partition can be synchronized independently by using this protocol. For details about the abstract data model used for synchronization, see [MS-FSSHTTPB] section 3.1.1.

A user on the protocol client or 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 contents a specific partition of a file specified by a URL. If the file exists on the protocol server, then 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 on **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 have changed that needs to be uploaded. The protocol client can also send upload request for changes done in the partitions associated with a file at a given URL. The server responds with a 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 a success in locking the file for edit. For more details on 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 co-authorable file and responds with a success in sharing the lock on the co-authorable 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 co-authoring status. For more details on co-authoring status, see section 2.2.5.1. For more details on 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 a shared lock request, the protocol server responds with an error code value indicating the type of error. For more details on error code types, see section 2.2.5.4.

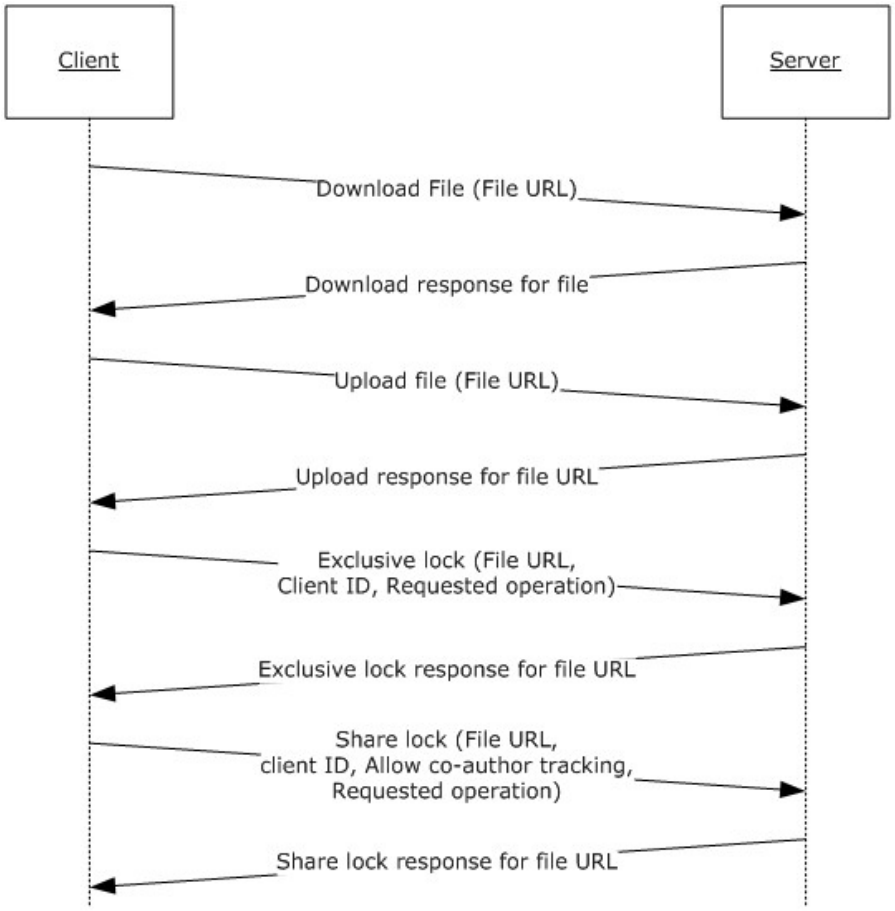

Figure 1: File upload/download to/from a server and lock requests to a server

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 binary requests for file synchronization binary protocol that allows for incremental file synching and the client local cache. Details on the binary requests for file synchronization binary protocol are described in [MS-FSSHTTPB].

Because multiple clients can co-author 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 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 is a word processing application that enables coauthoring and multi-user editing of files that are stored in a single protocol server.

1.4 Relationship to Other Protocols

This protocol uses the SOAP (Simple Object Access Protocol) 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 following diagram shows the underlying messaging and transport stack used by the protocol:

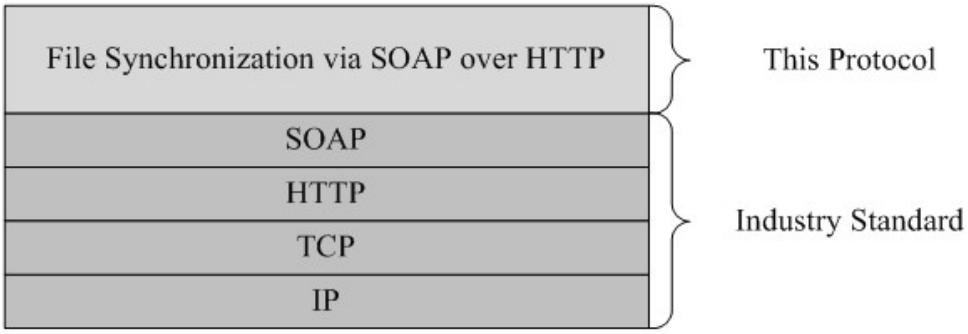

Figure 2: This protocol in relation to other protocols

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. For example: http://www.contoso.com/_vti_bin/cellstorage.svc.

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

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 an incremental update of the file. The protocol provides means that allows for this type of specification in the request.

The advantages of this protocol can be seen when used in conjunction with the Binary Requests for File Synchronization protocol described in [MS-FSSHTTPB] that allows for upload requests of incremental updates to the content s or partition data associated with the file.

The protocol is advantageous when used for upload/download of files that require coauthoring and are stored in 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

- Protocol servers MUST support SOAP over HTTP, as specified in [RFC2616].
- Protocol messages MUST be formatted as specified in [SOAP1.1] section 4 (SOAP Envelope).
- Protocol server MUST use MTOM encoding as specified in [SOAP1.2-MTOM].
- Protocol server faults MUST be returned either using HTTP Status codes as specified in [RFC2616], section 10 (Status Code Definitions) or using **SOAP faults** as specified in [SOAP1.1], section 4.4 (SOAP Fault).

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

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

2.2 Common Message Syntax

The section contains common definitions used by this protocol. The syntax of the definitions use **XML schema** as defined in [XMLSCHEMA1] and [XMLSCHEMA2].

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

The following specifies the protocol message schemas.

2.2.2.1 Request Message Schema

The following specifies the protocol request 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" />
```

```
 <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 of the **RequestVersion** element are specified in section 2.2.3.4 and details of the **RequestCollection** element are specified in section 2.2.3.3.

2.2.2.2 Response Message Schema

The following specifies the protocol response 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" />

<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 of the **ResponseVersion** element are specified in section 2.2.3.7 and details of the **ResponseCollection** element are specified in section 2.2.3.6.

2.2.2.3 SOAP Fault Message

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 standards fault codes and are 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 in the messages sub-section for the operations that return SOAP faults.

Request or Sub Request 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 specified with the operation.

Element name	Description
Include	The **Include** element is used for encapsulating and sending large amounts of binary data. The details of when an **Include** element is sent as part of a cell storage service request message and details of 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 cell storage service request, specific to a URL for the file with a unique identifier that identifies the request 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 cell storage service response specific to a URL

Element name	Description
	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 sub requests within a **Request** element.
SubRequestData	The **SubRequestData** element specifies data required for processing the sub request.
SubResponse	The **SubResponse** element specifies the corresponding response for the sub request.
SubResponseData	The **SubResponseData** element contains any data requested as part of the sub request.

2.2.3.1 Include Element

The **Include** element is used for encapsulating and sending large amounts of binary data. The **Include** element MUST only be sent 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** then the server MUST ignore it.

SubRequestData element is specified in section 2.2.3.9. **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 section 2.1 of [XOP10]. XML-binary Optimized Packaging (XOP) provides a means for more efficiently serializing **XML Information Set (Infoset)** that have certain types of content as specified in [XMLINFOSET].

2.2.3.2 Request Element

Each **Request** element maps to 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 sub-requests. 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="RequestToken" type="xs:nonNegativeInteger" use="required" />
  </xs:complexType>
</xs:element>
```

SubRequest: A **SubRequestElementGenericType** that specifies the type of sub request on the URL for the file and input parameters needed by a protocol server for processing the sub request.

The **SubRequest** element is defined in section 2.2.3.8. **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 is 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.

RequestToken: A non-negative integer that specifies the request token that uniquely identifies the **Request** element in a cell storage service request. **RequestToken** MUST be set to a non-negative integer value with a minimum allowed value of 0 and maximum allowed value of 4294967295. For each new **Request** element in a cell storage service request that needs to be sent to the protocol server, the **RequestToken** gets incremented by the protocol client. The **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 using the **RequestToken**. The **RequestToken** MUST be specified for each **Request** element.

Errors that occur during parsing of the **Request** element causes the error code value to be set to "InvalidArgument". The protocol server MUST send the error code as 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 appropriate error code MUST be returned by the protocol server. Generic error code types are defined in section 2.2.5.6.

2.2.3.3 RequestCollection Element

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. Logging of events with the **CorrelationId** helps in 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 parsing of the **RequestCollection** element MUST return a SOAP fault message. SOAP fault message is defined in section 2.2.2.3.

2.2.3.4 RequestVersion Element

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

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

The **VersionType** is specified in section 2.2.4.7.

Errors that occur because a version is not supported causes an **IncompatibleVersion** error code value to be set and sent as part of the **ResponseVersion** element. The **IncompatibleVersion** error code is defined in section 2.2.5.6. The **ResponseVersion** element is defined in section 2.2.3.7.

2.2.3.5 Response Element

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 sub request 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. The **SubResponse** element is not sent in a corresponding **Response** element as part of a cell storage service response if no **SubRequest** element was sent in the corresponding **Request** element as part of the cell storage service request by the protocol client.

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 non-negative 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 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 between 0 and 10, wherein a score of 0 specifies excellent server health and a score of 10 specifies very poor server health. Health score provides hints that help the protocol client throttle 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 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. The **Url** attribute and the **Request** element are defined in section 2.2.3.2.
- The **RequestToken** attribute of the corresponding **Request** element does not exist or is an empty string. The **RequestToken** attribute and the **Request** element are defined in section 2.2.3.2.
- An exception occurred during the processing of a sub request that was not entirely handled by the sub request processing logic.

ErrorMessage: A string that specifies a description of the error message and also specifies information of what was expected by the server. This attribute MUST be present exactly when the **ErrorCode** attribute is present.

2.2.3.6 ResponseCollection Element

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. The **Response** element is not sent in a corresponding **ResponseCollection** element as part of a cell storage service response if no **Request** element was sent in the corresponding **RequestCollection** element as part of the cell storage service request by the protocol client.

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 Element

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

VersionType 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 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 **RequestVersion** element is missing from the **Body** element of the SOAP request message. The **RequestVersion** element is defined in section 2.2.3.7.
- The **Version** attribute of the **RequestVersion** element of the request message 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 doesn't exist or isn't available. The **WebUrl** attribute and the **ResponseCollection** element are defined in section 2.2.3.6.
- The user doesn't 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 Element

The **SubRequest** element defines the sub request 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 Element

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 in which a **SubRequestData** element MUST be sent as part of the **SubRequest** element. 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 Element

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 case of success, **SubResponse** element specifies the information

requested as part of the **SubRequest** element. In case of failure, the error code attribute in a **SubResponse** element specifies the error code result returned during processing of the SubRequest element. 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** element. **SubResponseData** element is defined in section 2.2.3.11.

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

2.2.3.11 SubResponseData Element

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 in 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.

Complex type	Description
SubRequestDataGenericType	Generic type definition of data that is part of a cell storage service sub request.
SubRequestElementGenericType	Generic type definition of a cell storage service sub request.
SubRequestType	Base type definition of a cell storage service sub request. **SubRequestType** is used to extend **SubRequestElementGenericType**.
SubResponseDataGenericType	Generic type definition of data that is part of a cell storage service sub response.
SubResponseElementGenericType	Generic type definition of a cell storage service sub response.
SubResponseType	Base type definition of a cell storage service sub response. **SubResponseType** is used to extend **SubResponseElementGenericType**.
VersionType	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 sub request. **SubRequestDataGenericType** provides a generic sub request 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>
```

i:Include: A complex type, specified in [XOP10], section 2.1, that is used for encapsulating and sending large amounts of binary data. The **i: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 sub request, **SubRequestDataGenericType** MUST take one of the forms described in the following table.

Complex type	Description
CellSubRequestDataType	Type definition for cell sub request data.
CoauthSubRequestDataType	Type definition for co-authoring sub request data.
ExclusiveLockSubRequestDataType	Type definition for exclusive lock sub request data.
SchemaLockSubRequestDataType	Type definition for schema lock sub request 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.

The referenced **i: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 sub request is for the upload of file's binary contents or file's 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 sub request in a cell storage service request message. **SubRequestElementGenericType** provides a generic sub request type definition. The **SubRequestType** definition from which the **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 sub request 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 sub request to be processed. The sub request 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 sub request, **SubRequestElementGenericType** MUST take one of the forms described in the following table.

Complex type	Description
CellSubRequestType	Type definition for cell sub request when Type attribute is set to "Cell".
CoauthSubRequestType	Type definition for co-authoring sub request when Type attribute is set to "Coauth".
ExclusiveLockSubRequestType	Type definition for exclusive lock sub request when Type attribute is set to "ExclusiveLock".
SchemaLockSubRequestType	Type definition for schema lock sub request when Type attribute is set to "SchemaLock".
ServerTimeSubRequestType	Type definition for server time sub request when Type attribute is set to "ServerTime".
WhoAmISubRequestType	Type definition for Who Am I sub request when Type attribute is set to "WhoAmI".

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.

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 "SchemaLock"
- The **Type** attribute that is specified in the **SubRequest** element is set to a value of "Cell" and the cell sub request is for uploading or downloading of file's binary contents or file's 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"

2.2.4.3 SubRequestType

The **SubRequestType** complex type contains information about a basic cell storage service sub request. The **SubRequestType** is used as the base complex type to extend the **SubRequestElementGenericType**. The **SubRequestElementGenericType** takes one of the following forms, namely, **CellSubRequestType**, **CoauthSubRequestType**, **SchemaLockSubRequestType**, **ExclusiveLockSubRequestType**, **WhoAmISubRequestType** or **ServerTimeSubRequestType**. **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.

```
<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 non-negative 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 4294967295. For each new **SubRequest** element in a cell storage service request that needs to be sent to the protocol server, the **SubRequestToken** gets incremented by the protocol client. The **SubRequestToken** is reset to 1 by the protocol client for a new cell storage service request. The mapping sub response that gets generated for the sub request would reference the **SubRequestToken** to indicate that it is the response for that sub request. The **SubRequestToken** attribute MUST be specified for any type of SubRequest element.

DependsOn: A non-negative integer that specifies the **SubRequestToken** of the **SubRequest** element on which this specific sub request 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 sub request. **SubResponseDataGenericType** provides a generic sub response data type definition.

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

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

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 sub request for which this cell storage service **SubResponseData** is sent by the protocol server.

Complex type	Description
CellSubResponseDataType	Type definition for cell sub response data.
CoauthSubResponseDataType	Type definition for co-authoring sub response data.
ExclusiveLockSubResponseDataType	Type definition for exclusive lock sub response data.
SchemaLockSubResponseDataType	Type definition for schema lock sub response data.
ServerTimeSubResponseDataType	Type definition for server time sub response data.
WhoAmISubResponseDataType	Type definition for Who Am I sub response 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.

The referenced **i: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 sub request is for the download of file's binary contents or file's 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 sub request. In case of success, it contains information requested as part of the sub request. In case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for the sub request. **ErrorCode** attribute is specified in section 2.2.4.6. **SubResponseElementGenericType** provides a generic sub response type definition. **SubResponseType** definition from which the **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: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 sub request. **SubResponseDataGenericType** is defined in section 2.2.4.4.

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	Type definition for cell sub response.
CoauthSubResponseType	Type definition for co-authoring sub response.
ExclusiveLockSubResponseType	Type definition for ExclusiveLock sub response.
SchemaLockSubResponseType	Type definition for SchemaLock sub response.
ServerTimeSubResponseType	Type definition for server time sub response.
WhoAmISubResponseType	Type definition for Who Am I sub response.

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.

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 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 sub request. **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 sub response. The **SubResponseType** is used as the base complex type to extend **SubResponseElementGenericType**. The **SubResponseElementGenericType** takes one of the following forms, namely, **CellSubResponseType**, **CoauthSubResponseType**, **SchemaLockSubResponseType**, **ExclusiveLockSubResponseType**, **WhoAmISubResponseType** or **ServerTimeSubResponseType**. **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.

```
<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 non-negative integer that specifies a number that uniquely identifies the **SubRequest** element whose sub response is being generated as part of the **SubResponse** element. The mapping sub response that gets generated for the sub request would reference the **SubRequestToken** to indicate that it is the response for that sub request. 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 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 sub request that failed and gives more hints on 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 username 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 for future use. **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.

Simple type	Description
CoauthStatusType	Type of the **CoauthStatus** attribute of a co-authoring sub request or exclusive lock sub request. **CoauthStatusType** is an enumeration of co-authoring statuses of a file.
DependencyCheckRelatedErrorCodeTypes	Enumeration of error codes that occurs during sub request dependency checks.
DependencyTypes	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 dependency conditions supported for execution of a sub request.
ErrorCodeTypes	Type of the **ErrorCode** attribute of the **SubResponse** element. **ErrorCodeTypes** is a union of all possible error codes for the cell storage service response message, including success.
GenericErrorCodeTypes	A sub set of error codes returned as part of a cell storage service response message. **GenericErrorCodeTypes** is an enumeration of all 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 co-authoring sub request or schema lock sub request or an exclusive lock sub request.
LockTypes	Type of the **LockType** attribute, which is part of a **SubResponse** and **SubRequest** element. **LockTypes** is an enumeration of all file lock types.
MinorVersionNumberType	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	Type of the **Type** attribute of the **SubRequest** element. **SubRequestAttributeType** is an enumeration of all types of cell storage service sub request.
VersionNumberType	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 co-authoring status of a targeted URL for the file as part of processing a co-authoring sub request or exclusive lock sub request of type,

"Convert to schema lock with co-authoring transition tracked". The exclusive lock sub request of type, "Convert to schema lock with co-authoring transition tracked" is defined in section 2.3.3.4.

```
<xs:simpleType name="CoauthStatusType">
    <xs:restriction base="xs:string">
      <!--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
Alone	String value of "Alone", indicating a co-authoring status of alone. Alone status specifies that there is only one user in the co-authoring session who is editing the file.
Coauthoring	String value of "Coauthoring", indicating a co-authoring status of co-authoring. Co-authoring status specifies that the targeted URL for the file has more than one user in the co-authoring session and 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 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 sub request on which this specific sub request is dependent on has not been executed and the DependencyType attribute in this sub request is set to "OnExecute".
DependentOnlyOnSuccessRequestFailed	Indicates an error when the sub request on which this specific sub request is dependent on has failed and the DependencyType attribute in this sub request is set to "OnSuccess" or "OnSuccessOrNotSupported".

Value	Meaning
DependentOnlyOnFailRequestSucceeded	Indicates an error when the sub request on which this specific sub request is dependent on has succeeded and the DependencyType attribute in this sub request is set to "OnFail".
DependentOnlyOnNotSupportedRequestGetSupported	Indicates an error when the sub request on which this specific sub request is dependent on is supported and the DependencyType attribute in this sub request is set to "OnNotSupported" or "OnSuccessOrOnNotSupported".
InvalidRequestDependencyType	Indicates an error when a sub request dependency type that is not valid is specified.

DependencyTypes are 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 sub request has on another cell storage service sub request. The other cell storage service sub request 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 sub request is processed or not processed.

```
<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 sub request MUST be processed only on execution of the other sub request.
OnSuccess	Indicates that the sub request MUST be processed only on successful execution of the other sub request.
OnFail	Indicates that the sub request MUST be processed only on failed execution of the other sub request.
OnNotSupported	Indicates that the sub request MUST be processed only if the other sub request is not supported.
OnSuccessOrNotSupported	Indicates that the sub request MUST be processed only when one of the following conditions is true: ▪ On successful execution of the other sub request

Value	Meaning
	▪ If the other sub request is not supported.

2.2.5.4 ErrorCodeTypes

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

```
<xs:simpleType name="ErrorCodeTypes">
 <xs:union memberTypes="tns:GenericErrorCodeTypes tns:CellRequestErrorCodeTypes
tns:DependencyCheckRelatedErrorCodeTypes tns:LockAndCoauthRelatedErrorCodeTypes"/>
</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 2.2.5.8.

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	String value "CoauthoringDisabled", indicating that an exclusive lock is granted on a file because co-authoring is disabled.
CheckedOutByCurrentUser	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	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.

2.2.5.6 GenericErrorCodeTypes

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

```
<xs:simpleType name="GenericErrorCodeTypes">
    <xs:restriction base="xs:string">
      <xs:enumeration value="Success"/>
      <xs:enumeration value="IncompatibleVersion"/>
      <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="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 sub request 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.
FileNotExistsOrCannotBeCreated	Indicates an error when the targeted URL for the file specified as part of the **Request** element does not exists 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 that is 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.

Value	Meaning
InvalidArgument	Indicates an error when any of the cell storage service sub requests for the targeted URL for the file contains input parameters that are not valid.
RequestNotSupported	Indicates an error when the targeted cell storage service sub request is a valid sub request, but the server does not support that sub request.
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 processing of the cell storage service request.
ColdStoreConcurrencyViolation	Indicates an error when the file that is correctly stored on 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
Unknown	Indicates any undefined error that occurs during processing of the cell storage service request.

2.2.5.7 GUID

The **guid** type specifies 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 a co-authoring sub request or schema lock sub-request or exclusive lock sub request processing.

```
<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 processing of lock operations requested as part of a cell storage service sub request.
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 "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. **ErrorMessage** attribute and **ErrorCode** attribute is defined in section 2.2.4.6.
FileNotLockedOnServer	Indicates an error when no exclusive lock or shared 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 co-authoring of 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 co-authoring of file is disabled on the server.
FileAlreadyCheckedOutOnServer	Indicates an error when the file is checked out by another client that is preventing the file from being locked by the current client. When 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. **ErrorMessage** attribute and **ErrorCode** attribute is defined in section 2.2.4.6.
ConvertToSchemaFailedFileCheckedOutByCurrentUser	Indicates an error when convert to shared lock fails because the file is checked out by the current client.
CoauthRefblobConcurrencyViolation	Indicates an error when a save of the file co-authoring tracker, maintained by the protocol

Value	Meaning
	server fails after some other client edited the file co-authoring tracker before the save is done by the current client. The file co-authoring tracker is defined in section 3.1.1.
MultipleClientsInCoauthSession	Indicates an error when all of the following conditions are **true**: ▪ A co-authoring sub request of type, "Convert to exclusive lock" or schema lock sub request of type, "Convert to exclusive lock" is requested on a file. ▪ There is more than one client in the current co-authoring session for that file. ▪ **ReleaseLockOnConversionToExclusiveFailure** attribute specified as part of the sub request is set to **false**.
InvalidCoauthSession	Indicates an error when one of the following conditions is true when a co-authoring sub request or schema lock sub request is sent: ▪ No co-authoring session exists for the file. ▪ The current client does not exist in the co-authoring session for the file. ▪ The current client exists in the co-authoring session, but protocol server is unable to remove it from the co-authoring session for the file. A co-authoring session indicates a shared lock on the co-authorable file that is shared by one or more clients.
NumberOfCoauthorsReachedMax	Indicates an error when the number of users that co-author a file has reached the threshold limit. The threshold limit specifies the maximum number of users allowed to co-author a file at any instant of 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 co-authoring sub request or schema lock sub request 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 type of file lock string values. **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	String value "None", indicating no type of file lock on the file.
SchemaLock	String value "SchemaLock", indicating a shared lock on the file. In a cell storage service request message, shared lock indicates a request for sharing the lock on the file that allows for co-authoring of the file. In a cell storage service response message, shared lock indicates that the current client is granted a shared lock on the file that allows for co-authoring of the file along with other clients.
ExclusiveLock	String value "ExclusiveLock", indicating an exclusive lock on the file. In a cell storage service request message, exclusive lock indicates a request for exclusive access to the file. In a cell storage service response message, exclusive lock indicates that an exclusive lock is granted to the current client for that specific file. In a cell storage service response message, exclusive lock also indicates that all other clients requesting for 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 minor version number as unsigned short values.

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

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

Value	Meaning
0	Unsigned short value 0.

2.2.5.11 SubRequestAttributeType

The **SubRequestAttributeType** simple type is used to represent the type of cell storage service sub request. Depending on the type of sub request, the sub request is processed as a cell sub request operation or a co-authoring sub request operation or a schema lock sub request operation or exclusive lock sub request operation or WhoAmI sub request operation or server time sub request operation.

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

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

Value	Meaning
Cell	String value "Cell", indicating the cell storage service sub request is to be processed as a cell sub request operation.
Coauth	String value "Coauth", indicating the cell storage service sub request is to be processed as a co-authoring sub request operation.
SchemaLock	String value "SchemaLock", indicating the cell storage service sub request is to be processed as a schema lock sub request operation.
WhoAmI	String value "WhoAmI", indicating the cell storage service sub request is to be processed as a WhoAmI sub request operation.
ServerTime	String value "ServerTime", indicating the cell storage service sub request is to be processed as a server time sub request operation.
ExclusiveLock	String value "ExclusiveLock", indicating the cell storage service sub request is to be processed as a co-authoring exclusive lock sub request operation.

2.2.5.12 VersionNumberType

The **VersionNumberType** simple type is used to represent 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 one of the values in the following table.

Value	Meaning
2	Unsigned short value 2.

2.2.6 Attributes

None

2.2.7 Groups

None

2.2.8 Attribute Groups

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

Attribute	Description
SubRequestDataOptionalAttributes	Contains XML schema attribute used in all **SubRequestData** elements. It is a union of sub request data attributes used for all types of sub requests.
SubResponseDataOptionalAttributes	Contains XML schema attribute used in all **SubResponseData** elements. It is a union of sub response data attributes used for all types of sub responses.

2.2.8.1 SubRequestDataOptionalAttributes

The **SubRequestDataOptionalAttributes** attribute group contains attribute that are used in **SubRequestData** elements of all types of sub requests. **SubRequestDataOptionalAttributes** are used as input parameters in processing the data associated with sub requests. 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: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:attributeGroup>
```

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

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

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

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

ClientID: A string that specifies a string value that serves to uniquely identify each client that has access to a shared lock on a co-authorable file.

AllowFallbackToExclusive: A Boolean that specifies to a protocol server, whether a co-authoring sub request or a schema lock sub request is allowed to fallback to an exclusive lock sub request 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 co-authoring sub request or a schema lock sub request is allowed to fallback to an exclusive lock sub request.
- An **AllowFallbackToExclusive** attribute value set to **false** indicates that a co-authoring sub request or a schema lock sub request is not allowed to fallback to an exclusive lock sub request.

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

- The type of co-authoring sub request is "Convert to an exclusive lock" or the type of the schema lock sub request 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 co-authoring 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 co-authoring tracker.

SchemaLockID: A string that specifies a string value 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 a different schema identifier. Once a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST only allow 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 all clients having the same schema lock identifier lock the file and not protocol clients with different schema lock identifiers. Once the shared lock by all protocol clients for that file have released their lock, the protocol server MUST allow a protocol client with a different schema lock identifier to get a shared lock for that file. String "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use, for this attribute.<1>

Timeout: An integer that specifies the time in seconds, after which the shared lock or exclusive lock for that specific file expires for that 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 specifies a string value that serves as a unique identifier for the exclusive lock on the file.

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 sub response for any type of cell storage service sub request. **SubResponseDataOptionalAttributes** provide the data that was requested as part of the sub request. 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 only be used for **SubResponseData** elements associated with a sub response for a cell sub request. The **CellSubResponseDataOptionalAttributes** attribute group is defined in section 2.3.3.2.

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

ServerTime: A positive integer that specifies the server time, expressed in tick count. A single tick represents one hundred nanoseconds or one ten-millionth of a second. **ServerTime** specifies the number of 100-nanosecond intervals that have elapsed since 12:00:00 midnight, January 1st, 1601 and SHOULD <2> be **Coordinated Universal Time (UTC)**. The **ServerTime** attribute MUST be specified in a server time sub response that is generated in response to a server time sub request.

LockType: A **LockTypes** that specifies the type of lock granted in a co-authoring sub response or a schema lock sub response. The **LockTypes** is defined in section 2.2.5.9. The **LockType** attribute MUST be specified in a sub response that is generated in response to one of the following types of cell storage service sub request operation, if the **ErrorCode** attribute that is part of the SubResponse element is set to a value of "Success":

- Co-authoring sub request of type, "Join co-authoring session".
- Co-authoring sub request of type, "Refresh co-authoring session".
- Schema lock sub request of type, "Get lock".
- Schema lock sub request of type, "Refresh lock".

The types of co-authoring sub request are defined in section 2.3.3.3. The types of schema lock sub request are defined in section 2.3.3.5.

CoauthStatus: A **CoauthStatusType** that specifies the co-authoring status in a co-authoring sub response or an exclusive lock sub response. The **CoauthStatusType** is defined in section 2.2.5.1. **CoauthStatus** attribute MUST be specified in a sub response that is generated in response to one of the following types of cell storage service sub request operation, if the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success":

- Co-authoring sub request of type, "Join co-authoring session".
- Co-authoring sub request of type, "Refresh co-authoring session".
- Co-authoring sub request of type, "Get co-authoring status".
- Exclusive lock sub request of type, "Convert to schema lock with co-authoring transition tracked".

The types of co-authoring sub request are defined in section 2.3.3.3. The types of exclusive lock sub request are defined in section 2.3.3.4.

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

ExclusiveLockReturnReason: An **ExclusiveLockReturnReasonTypes** that specifies the reason for why an exclusive lock is granted in a co-authoring sub response or a schema lock sub response. The **ExclusiveLockReturnReasonTypes** is defined in section 2.2.5.5. The **ExclusiveLockReturnReason** attribute MUST be specified in a sub response that is generated in response to one of the following types of cell storage service sub request operation and the **LockType** attribute in the subresponse is set to "ExclusiveLock":

- Co-authoring sub request of type, "Join co-authoring session".
- Co-authoring sub request of type, "Refresh co-authoring session".
- Schema lock sub request of type, "Get lock".
- Schema lock sub request of type, "Refresh lock".

The types of co-authoring sub request are defined in section 2.3.3.3. The types of schema lock sub request are defined in section 2.3.3.5.

2.3 Other Message Syntax

The section contains definitions used by this protocol. The syntax of the definitions use the XML schema as defined in [XMLSCHEMA1] and [XMLSCHEMA2].

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	Type definition for cell sub request data.
CellSubRequestType	Type definition for cell sub request when Type attribute is set to "Cell".
CellSubResponseDataType	Type definition for cell sub response data.
CellSubResponseType	Type definition for cell sub response.
CoauthSubRequestDataType	Type definition for co-authoring sub request data.
CoauthSubRequestType	Type definition for co-authoring sub request when Type attribute is set to "Coauth".

Complex type	Description
CoauthSubResponseDataType	Type definition for co-authoring sub response data.
CoauthSubResponseType	Type definition for co-authoring sub response.
ExclusiveLockSubRequestDataType	Type definition for exclusive lock sub request data.
ExclusiveLockSubRequestType	Type definition for exclusive lock sub request when Type attribute is set to "ExclusiveLock".
ExclusiveLockSubResponseDataType	Type definition for exclusive lock sub response data.
ExclusiveLockSubResponseType	Type definition for ExclusiveLock sub response.
SchemaLockSubRequestDataType	Type definition for schema lock sub request data.
SchemaLockSubRequestType	Type definition for schema lock sub request when Type attribute is set to "SchemaLock".
SchemaLockSubResponseDataType	Type definition for schema lock sub response data.
SchemaLockSubResponseType	Type definition for SchemaLock sub response.
ServerTimeSubRequestType	Type definition for server time sub request when Type attribute is set to "ServerTime".
ServerTimeSubResponseDataType	Type definition for server time sub response data.
ServerTimeSubResponseType	Type definition for server time sub response.
WhoAmISubRequestType	Type definition for Who Am I sub request when Type attribute is set to "WhoAmI".
WhoAmISubResponseDataType	Type definition for Who Am I sub response data.
WhoAmISubResponseType	Type definition for Who Am I sub response.

2.3.1.1 CellSubRequestDataType

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

```
<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>
```

i:Include: A complex type that is specified in section 2.1 of [XOP10] and used for encapsulating and sending large amounts of binary data. The referenced **i:Include** element is specified in section 2.2.3.1. The referenced **i:Include** element MUST be sent as part of the **SubRequestData** element

in a cell storage service request message only if the cell sub request is for the upload of file's binary contents or file's metadata contents.

CellSubRequestDataOptionalAttributes: An attribute group that specifies the set of attributes that are 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 specifies a string value 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 a different schema lock identifier. Once a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST only allow 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 all clients having the same schema lock identifier lock the file and not protocol clients with different schema lock identifiers. Once the shared lock by all protocol clients for that file have released their lock, 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 sub request if one of the following conditions is **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 sub request is for uploading binary file contents in a partition.

String "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use, for this attribute.<3>

ExclusiveLockID: A string that specifies a string value that serves as a unique identifier for the type of exclusive lock on the file when a cell sub request with **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 sub request is for first time upload of file's binary contents.

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

If text is specified in the **SubRequestData** element, this is base64 binary encoded data and indicates if the cell sub request is for upload or download of data in a partition. This is passed on to the component on the protocol server responsible for implementing the protocol as specified in [MS-FSSHTTPB] section 3.1.4.2 and [MS-FSSHTTPB] section 3.1.4.5. The encoded data is opaque to the protocol.

BinaryDataSize: A long that specifies the numbers of bytes of data in the **SubRequestData** element of a cell sub request. It MUST be present in the **SubRequestData** element of a cell sub request. 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. **SubRequestData** element is defined in section 2.2.3.9.

2.3.1.2 CellSubRequestType

The **CellSubRequestType** complex type contains information about a cell sub request. **SubRequestType** definition from which the **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 sub request. If no **SubRequestData** element is specified in the cell sub request, the protocol server MUST process this as a no-operation. **CellSubRequestDataType** is defined in section 2.3.1.1.

Type: A **SubRequestAttributeType** that specifies the type of sub request. The **Type** attribute MUST be set to "Cell" for cell sub request. **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 sub request.

```
<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="CoalesceHResult" type="xs:integer" use="required" />
   <xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
   <xs:attribute name="ContainsHotboxData" type="xs:boolean" use="required"/>
   <xs:attribute name="HaveOnlyDemotionChanges" type="xs:boolean" use="required"/>
</xs:complexType>
```

i:Include: A complex type that is specified in section 2.1 of [XOP10] and used for encapsulating and sending large amounts of binary data. The referenced **i:Include** element is specified in section 2.2.3.1. As part of processing the cell sub request, the referenced **i:Include** element MUST be sent as part of the **SubResponseData** element in a cell storage service response message only if the cell sub request is for the download of file's binary contents or file's 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 sub request. **CellSubResponseDataOptionalAttributes** is defined in section 2.3.3.2.

CoalesceHResult: An integer. It MUST be 0 except when the protocol server attempts to fully save all changes in the underlying store. It specifies the **HRESULT** when the protocol server attempts to fully save all changes in the underlying store. The CoalesceHResult MUST be set to a value ranging from 0 to 2147483647. A **CoalesceHResult** of 0 indicates a success. If the **CoalesceHResult** is greater than 0, it indicates an exception or failure condition that occurred.

LockType: A **LockTypes** that specifies the type of lock granted in a cell sub response. The LockTypes is defined in section 2.2.5.9. The **LockType** attribute MUST be set to "ExclusiveLock" in

the cell sub response if the **ExclusiveLockID** attribute is sent in the cell sub request and the protocol server is successfully able to take an exclusive lock. The condition, under which **ExclusiveLockID** attribute is sent in the cell sub request, is specified in section 2.3.1.1.

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

HaveOnlyDemotionChanges: A Boolean that specifies whether the returned binary content only has protocol server property demotion changes compared to the client's last synced status. If the value is set to **false**, the returned binary content contains changes other than property demotion changes.

2.3.1.4 CellSubResponseType

The **CellSubResponseType** complex type contains information about the success or failure in processing the cell sub request. In case of success, it contains information requested as part of the cell sub request. **SubResponseType** definition from which the **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="1"
maxOccurs="1" />
        <xs:element name="SubResponseStreamInvalid" type="tns:CellSubResponseDataType"
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 sub request. **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 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 co-authoring sub request. The **SchemaLockID** attribute and the **CoauthRequestType** attribute specified in the **CoauthSubRequestDataOptionalAttributes** attribute group MUST both be specified for a co-authoring sub request. **SchemaLockID** attribute and the **CoauthRequestType** attribute is specified as part of the **SubRequestData** element associated with a co-authoring **SubRequest** element.
CoauthSubRequestDataOptionalAttributes is defined in section 2.3.3.3. If the specified attributes are not provided, then an "InvalidArgument" error code MUST be returned as part of the **SubResponseData** element associated with the co-authoring sub response.

```
<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 are 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 specifies a string value that serves to uniquely identify each client that has access to a shared lock on a co-authorable file. **ClientID** attribute MUST be specified on all types of co-authoring sub requests. The types of coauthoring sub request are defined in section 2.3.3.3.

AllowFallbackToExclusive: A Boolean that specifies to a protocol server, whether a co-authoring sub request is allowed to fallback to an exclusive lock sub request 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 co-authoring sub request is allowed to fallback to an exclusive lock sub request.
- An **AllowFallbackToExclusive** attribute value set to **false** indicates that a co-authoring sub request is not allowed to fallback to an exclusive lock sub request.

The **AllowFallbackToExclusive** attribute is specified as part of a co-authoring sub request of type, "Join co-authoring session". The types of coauthoring sub request are defined in section 2.3.3.3.

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

- The type of co-authoring sub request 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 co-authoring 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 co-authoring tracker.
- A **ReleaseLockOnConversionToExclusiveFailure** attribute MUST only be sent when the co-authoring sub request type is set to "Convert to exclusive lock". The types of coauthoring sub request are defined in section 2.3.3.3. The file co-authoring tracker is defined in section 3.1.1.

SchemaLockID: A string that specifies a string value 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 a different schema identifier. Once a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST only allow 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 all clients having the same schema lock identifier lock the file and not protocol clients with different schema lock identifiers. Once the shared lock by all protocol clients for that file have released their lock, 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 co-authoring sub requests. String "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use, for this attribute.<4>

Timeout: An integer that specifies the time in seconds, after which the shared lock for that particular file expires for that specific protocol client. The **Timeout** attribute MUST be set to a value ranging from 3600 seconds to 120,000 seconds. When more than one client is editing the file, the protocol server MUST maintain a separate timeout value for each client in the file co-authoring tracker. The file co-authoring tracker is defined in section 3.1.1. The client's timeout on a shared lock for a file is refreshed by sending a co-authoring sub request of type, "Refresh co-authoring session". The timeout attribute MUST be sent in the following types of co-authoring sub request:

- Join co-authoring session
- Refresh co-authoring session
- Convert to exclusive lock

The types of co-authoring sub requests are defined in section 2.3.3.3.

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

2.3.1.6 CoauthSubRequestType

The **CoauthSubRequestType** complex type contains information about a co-authoring sub request. **SubRequestType** definition from which the **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 co-authoring sub request. **CoauthSubRequestDataType** is defined in section 2.3.1.5.

Type: A **SubRequestAttributeType** that specifies the type of sub request. The **Type** attribute MUST be set to "Coauth" for co-authoring sub request. **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 co-authoring sub request.

```
<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 co-authoring sub response. The **LockTypes** is defined in section 2.2.5.9. The **LockType** attribute MUST be specified in a co-authoring sub response that is generated in response to all of the following types of co-authoring sub requests, if the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success":

- Join co-authoring session
- Refresh co-authoring session

The different types of co-authoring sub request are defined in section 2.3.3.3.

CoauthStatus: A **CoauthStatusType** that specifies the co-authoring status in a co-authoring sub response. The **CoauthStatusType** is defined in section 2.2.5.1. **CoauthStatus** attribute MUST be specified in a co-authoring sub response that is generated in response to all of the following types of co-authoring sub requests, if the **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success":

- Join co-authoring session
- Refresh co-authoring session
- Get co-authoring status.

The different types of co-authoring sub request are defined in section 2.3.3.3.

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

ExclusiveLockReturnReason: An **ExclusiveLockReturnReasonTypes** that specifies the reason for why an exclusive lock is granted in a co-authoring sub response. The **ExclusiveLockReturnReasonTypes** is defined in section 2.2.5.5. The **ExclusiveLockReturnReason** attribute MUST be specified in a co-authoring sub response that is generated in response to all of the following types of co-authoring sub requests and the **LockType** attribute in the subresponse is set to "ExclusiveLock":

- Join co-authoring session
- Refresh co-authoring session

The types of co-authoring sub request 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 co-authoring sub request. In case of success, it contains co-authoring information requested as part of the co-authoring sub request. In case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this sub request. **ErrorCode** attribute is specified in section 2.2.4.6. **SubResponseType** definition from which the **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 co-authoring related information provided by the protocol server that was requested as part of the co-authoring sub request. **CoauthSubResponseDataType** is defined in section 2.3.1.7. As part of processing the co-authoring sub request, 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 co-authoring sub request. **ErrorCode** attribute 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 sub request. The **ExclusiveLockID** attribute and the **ExclusiveLockRequestType** attribute specified in the **ExclusiveLockSubRequestDataOptionalAttributes** attribute group MUST both be specified for an exclusive lock sub request. 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, then an "InvalidArgument" error code MUST be returned as part of the **SubResponseData** element associated with the exclusive lock sub response.

```
<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 only provided 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 specifies a string value that serves to uniquely identify each client that has access to a shared a lock on a co-authorable file. **ClientID** attribute MUST be specified when the exclusive lock sub request has an **ExclusiveLockSubRequestType** attribute set to "ConvertToSchemaLock" or "ConvertToSchemaJoinCoauth". **ExclusiveLockSubRequestType** that specifies the different types of Exclusive lock sub request is defined in section 2.3.3.4.

SchemaLockID: A string that specifies a string value 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 a different schema identifier. Once a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST only allow 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 all clients having the same schema lock identifier lock the doc and not protocol clients with different schema lock identifiers. Once the shared lock by all protocol clients for that file have released their lock, 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 sub request has an **ExclusiveLockSubRequestType** attribute set to "ConvertToSchemaLock" or "ConvertToSchemaJoinCoauth". **ExclusiveLockSubRequestType** that specifies the different types of Exclusive lock sub request is defined in section 2.3.3.4. String "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use, for this attribute<5>.

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

- Get lock.
- Refresh lock.
- Convert to schema lock.
- Convert to schema lock with co-authoring transition tracked.

The types of exclusive lock sub request are defined in section 2.3.3.4.

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

2.3.1.10 ExclusiveLockSubRequestType

The **ExclusiveLockSubRequestType** complex type contains information about an exclusive lock sub request. **SubRequestType** definition from which the **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 sub request. **ExclusiveLockSubRequestDataType** is defined in section 2.3.1.9.

Type: A **SubRequestAttributeType** that specifies the type of sub request. The Type attribute MUST be set to "ExclusiveLock" for exclusive lock sub request. **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 sub request.

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

CoauthStatus: A **CoauthStatusType** that specifies the co-authoring status in an exclusive lock sub response. **CoauthStatusType** is defined in section 2.2.5.1. **CoauthStatus** attribute MUST only be specified in an exclusive lock sub response that is generated in response to an exclusive lock sub request of type, "Convert to schema lock with co-authoring 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 sub request 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 response for an exclusive lock sub request of type, "Convert to Schema lock with co-authoring 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 sub request. In case of success, it contains information requested as part of the exclusive lock sub request. In case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this sub request. **ErrorCode** attribute is specified in section 2.2.4.6. **SubResponseType** definition from which the **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 sub request. **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 sub request. The **SchemaLockID** attribute and the **SchemaLockRequestType** attribute specified in the **SchemaLockSubRequestDataOptionalAttributes** attribute group MUST both be specified for a schema lock sub request. **SchemaLockID** attribute and the **SchemaLockRequestType** attribute is 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, then an "InvalidArgument" error code MUST be returned as part of the **SubResponseData** element associated with the schema lock sub response.

```
<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 only provided 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 specifies a string value that serves to uniquely identify each client that has access to a shared lock on a co-authorable file. **ClientID** attribute MUST be specified on all types of schema lock sub requests. The different types of schema lock sub request are defined in section 2.3.3.5.

AllowFallbackToExclusive: A Boolean that specifies to a protocol server, whether a schema lock sub request is allowed to fallback to an exclusive lock sub request 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 schema lock sub request is allowed to fallback to an exclusive lock sub request.

- An **AllowFallbackToExclusive** attribute value set to **false** indicates that a schema lock sub request is not allowed to fallback to an exclusive lock sub request.

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

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

- The type of the schema lock sub request 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 co-authoring 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 co-authoring tracker.

ReleaseLockOnConversionToExclusiveFailure attribute MUST only be sent when the schema lock sub request type is set to "Convert to exclusive lock". The types of schema lock sub request are defined in section 2.3.3.5. The file co-authoring tracker is defined in section 3.1.1.

SchemaLockID: A string that specifies a string value 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 a different schema identifier. Once a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST only allow 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 all clients having the same schema lock identifier lock the file and not protocol clients with different schema lock identifiers. Once the shared lock by all protocol clients for that file have released their lock, 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 sub requests. String "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use, for this attribute<6>.

Timeout: An integer that specifies the time in seconds, after which the shared lock for that particular file expires for that specific protocol client. The **Timeout** attribute MUST be set to a value ranging from 3600 seconds to 120,000 seconds. 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 sub request of type, "Refresh lock". The timeout attribute MUST be specified in all of the following types of schema lock sub request:

- Get lock.
- Refresh lock.
- Convert to exclusive lock.

The types of schema lock sub requests are defined in section 2.3.3.5.

ExclusiveLockID: A string that specifies a string value that serves as a unique identifier for the exclusive lock on the file when a schema lock sub request of type, "Convert to exclusive lock" is

requested. **ExclusiveLockID** MUST be specified when the type of the schema lock sub request is "Convert to exclusive lock" or the type of schema lock sub request is "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 sub request. **SubRequestType** definition from which the **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 sub request. **SchemaLockSubRequestDataType** is defined in section 2.3.1.13.

Type: A **SubRequestAttributeType** that specifies the type of sub request. The **Type** attribute MUST be set to "SchemaLock" for schema lock sub request. **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 sub request.

```
<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 sub response. The **LockTypes** is defined in section 2.2.5.9. **LockType** attribute MUST be specified in a schema lock sub response that is generated in response to a schema lock sub request of type, "Get lock" and "Refresh lock", if the **ErrorCode** attribute that is part of the SubResponse element is set to a value of "Success". The types of schema lock sub request are defined in section 2.3.3.5.

ExclusiveLockReturnReason: An **ExclusiveLockReturnReasonTypes** that specifies the reason for why an exclusive lock is granted in a schema lock sub response. The **ExclusiveLockReturnReasonTypes** is defined in section 2.2.5.5. The **ExclusiveLockReturnReason** attribute MUST be specified in a schema lock sub response that is generated in response to a schema lock sub request of type, "Get lock" and "Refresh lock" when **LockType** attribute in the sub-response is set to "ExclusiveLock". The types of schema lock sub request 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 sub request. In case of success, it contains information requested as part of the schema lock sub request. In case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this sub request. **ErrorCode** attribute is specified in section 2.2.4.6. **SubResponseType** definition from which the **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 sub request. **SchemaLockSubResponseDataType** is defined in section 2.3.1.15.

2.3.1.17 ServerTimeSubRequestType

The **ServerTimeSubRequestType** complex type contains information about a server time sub request. **SubRequestType** definition from which the **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 sub request. The Type attribute MUST be set to "ServerTime" for server time sub request. **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 sub request.

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

ServerTime: A positive integer that specifies the server time, expressed in tick count. A single tick represents one hundred nanoseconds or one ten-millionth of a second. **ServerTime** specifies the

number of 100-nanosecond intervals that have elapsed since 12:00:00 midnight, January 1st, 0001 and SHOULD<7> be Coordinated Universal Time (UTC). The **ServerTime** attribute MUST be specified in a server time sub response that is generated in response to a server time sub request.

2.3.1.19 ServerTimeSubResponseType

The **ServerTimeSubResponseType** complex type contains information about the success or failure in processing the server time sub request. In case of success, it contains information requested as part of server time sub request. In case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this sub request. **ErrorCode** attribute is specified in section 2.2.4.6. **SubResponseType** definition from which the **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 sub request. **ServerTimeSubResponseDataType** is defined in section 2.3.1.18.

2.3.1.20 WhoAmISubRequestType

The **WhoAmISubRequestType** complex type contains information about Who Am I sub request. **SubRequestType** definition from which the **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 sub request. The **Type** attribute MUST be set to "WhoAmI" for WhoAmI sub request. **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** sub request.

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

```
    <xs: attribute name="UserLogin" type="tns:UserLoginType" use="required"/>
</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** sub request. **WhoAmISubResponseDataOptionalAttributes** is defined in section 2.3.3.6.

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** sub response that is generated in response to a **WhoAmI** sub request.

2.3.1.22 WhoAmISubResponseType

The **WhoAmISubResponseType** complex type contains information about the success or failure in processing **WhoAmI** sub request. In case of success, it contains information requested as part of **WhoAmI** sub request. In case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this sub request. **ErrorCode** attribute is specified in section 2.2.4.6. **SubResponseType** definition from which the **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** sub request. **WhoAmISubResponseDataType** is defined in section 2.3.1.21. As part of processing the **WhoAmI** sub request, 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** sub request. **ErrorCode** attribute is specified in section 2.2.4.6.

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 sub set of error codes returned for a cell sub request as part of a cell storage service response message. **CellRequestErrorCodeTypes** is an enumeration of error codes specific to a cell sub request.

Simple type	Description
CoauthRequestTypes	Type of the **CoauthRequestType** attribute which is part of a co-authoring sub request. **CoauthRequestTypes** is an enumeration of all co-authoring request types.
ExclusiveLockRequestTypes	Type of the **ExclusiveLockRequestType** attribute, which is part of an exclusive lock sub request. **ExclusiveLockRequestTypes** is an enumeration of all exclusive lock request types.
SchemaLockRequestTypes	Type of the **SchemaLockRequestType** attribute, which is part of a schema lock sub request. **SchemaLockRequestTypes** is an enumeration of all schema lock request types.
UserLoginType	Specifies user login value.
UserNameType	Specifies user name value.

2.3.2.1 CellRequestErrorCodeTypes

The **CellRequestErrorCodeTypes** simple type is used to represent error codes that occur during cell sub request 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 sub request 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 only supported through WebDAV (Web Distributed Authoring and Versioning Protocol).

2.3.2.2 CoauthRequestTypes

The **CoauthRequestTypes** simple type is used to represent the type of co-authoring sub request. **CoauthRequestTypes** is the type definition of the **CoauthRequestType** attribute, which is part of a co-authoring sub request 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	String value "JoinCoauthoring", specifying a co-authoring sub request of type, "Join co-authoring session".
ExitCoauthoring	String value "ExitCoauthoring", specifying a co-authoring sub request of type, "Exit co-authoring session".
RefreshCoauthoring	String value "RefreshCoauthoring", specifying a co-authoring sub request of type, "Refresh co-authoring session".
ConvertToExclusive	String value "ConvertToExclusive", specifying a co-authoring sub request of type, "Convert to exclusive lock".
CheckLockAvailability	String value "CheckLockAvailability", specifying a co-authoring sub request of type, "Check lock availability".
MarkTransitionComplete	String value "MarkTransitionComplete ", specifying a co-authoring sub request of type, "Mark transition to complete".
GetCoauthoringStatus	String value "GetCoauthoringStatus", specifying a co-authoring sub request of type, "Get co-authoring status".

2.3.2.3 ExclusiveLockRequestTypes

The **ExclusiveLockRequestTypes** simple type is used to represent the type of exclusive lock sub request. **ExclusiveLockRequestTypes** is the type definition of the **ExclusiveLockRequestType** attribute, which is part of an exclusive lock sub request 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	String value "GetLock", indicating an exclusive lock sub request of type, "Get lock".
ReleaseLock	String value "ReleaseLock", indicating an exclusive lock sub request of type, "Release lock".
RefreshLock	String value "RefreshLock", indicating an exclusive lock sub request of type, "Refresh lock".
ConvertToSchemaJoinCoauth	String value "ConvertToSchemaJoinCoauth", indicating an exclusive lock sub request of type, "Convert to schema lock with co-authoring transition tracked".
ConvertToSchema	String value "ConvertToSchema", indicating an exclusive lock sub request of type, "convert to schema lock".
CheckLockAvailability	String value "CheckLockAvailability", indicating an exclusive lock sub request of type, "Check lock availability".

2.3.2.4 SchemaLockRequestTypes

The **SchemaLockRequestTypes** simple type is used to represent the type of schema lock sub request. **SchemaLockRequestTypes** is the type definition of the **SchemaLockRequestType** attribute, which is part of a schema lock sub request 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	String value "GetLock", indicating a schema lock sub request of type, "Get lock".
ReleaseLock	String value "ReleaseLock", indicating a schema lock sub request of type, "Release lock".

Value	Meaning
RefreshLock	String value "RefreshLock ", indicating a schema lock sub request of type, "Refresh lock".
ConvertToExclusive	String value "ConvertToExclusive", indicating a schema lock sub request of type, "Convert to exclusive lock".
CheckLockAvailability	String value "CheckLockAvailability", indicating a schema lock sub request of type, "Check lock availability".

2.3.2.5 UserLoginType

The **UserLoginType** simple type specifies a representation of a user login value as specified in [RFC2822]. **UserLoginType** is the type definition of the **UserLogin** attribute, which is part of the sub response for a **WhoAmI** sub request.

```
<xs:simpleType name="UserLoginType">
   <xs:restriction base="xs:string">
      <xs:pattern value="^[a-zA-Z]([a-zA-Z0-9\-_])*\\[a-zA-Z]([a-zA-Z0-9])*"/>
   </xs:restriction>
</xs:simpleType>
```

2.3.2.6 UserNameType

The **UserNameType** simple type specifies representation of a user name value as specified in [RFC2822]. **UserNameType** is the type definition of the **UserName** attribute, which is part of the sub response for a **WhoAmI** sub request.

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

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 sub requests.
CellSubResponseDataOptionalAttributes	Contains XML schema attributes that are used in cell sub responses.
CoauthSubRequestDataOptionalAttributes	Contains XML schema attributes that are used in co-authoring sub requests.
ExclusiveLockSubRequestDataOptionalAttributes	Contains XML schema attributes that are used in exclusive lock sub requests.
SchemaLockSubRequestDataOptionalAttributes	Contains XML schema attributes that are used in

Attribute	Description
	schema lock sub requests.
WhoAmISubResponseDataOptionalAttributes	Contains XML schema attributes that are used in WhoAmI sub responses.

2.3.3.1 CellSubRequestDataOptionalAttributes

The **CellSubRequestDataOptionalAttributes** attribute group contains attributes that MUST only be used for **SubRequestData** elements associated with parent **SubRequest** element of cell type. **CellSubRequestDataOptionalAttributes** are used as input parameters in processing the data associated with a cell sub request. 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:attributeGroup>
```

Coalesce: A Boolean that specifies whether the protocol server SHOULD<8> fully save all changes to the underlying store without storing the changes in any intermediate write cache after processing the sub request. If **Coalesce** attribute is set to true in a cell sub request, the protocol server ensures to persist all changes to the file after processing the sub request. The **Coalesce** attribute is set to false in the cell sub request for updates to other partitions that do not contain binary file contents.

GetFileProps: A Boolean that specifies whether the file properties are requested or not. When set to true, file properties are requested as part of the cell sub request. When set to false, file properties are not requested as part of the cell sub request. When set to true, the protocol server MUST return **CreateTime** and **LastModifiedTime** as attributes in the cell **SubResponseData** element.

CoauthVersioning: A Boolean. If the co-authoring status of a client is "Coauthoring" then this value MUST **true** for upload requests. Otherwise it MUST **false**. For more details on Co-authoring status, see **CoauthStatusType** (section 2.2.5.1).

Etag: A string that specifies 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 co-authorable file. Anytime the protocol client specifies the **Etag** attribute in a cell sub request, 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 of **Etag** between the version specified by the client and version stored by the server, the protocol server MUST send an error code value set to "CellRequestFail" in the cell sub response message. The protocol server processes this value as specified in [RFC2616].

ContentChangeUnit: A string that specifies a string value that uniquely identifies the synchronization version of the file content. It is the value of the property **vti_contentchangeunit** as defined in [MS-LISTSWS].

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

PartitionID: A **guid** that specifies a unique identifier for the partition-block in which the file contents or file metadata contents is 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. The **PartitionID** of the file contents is "00000000-0000-0000-0000-000000000000".

ExpectNoFileExists: A Boolean 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 upload of file content.

BypassLockID: A string that specifies a unique string value. If a client wants to acquire an exclusive lock then it MUST be the same as **ExclusiveLockID** specified in section 2.3.1.1. If a client wants to acquire a shared lock then it MUST be the same as **SchemaLockID** specified in section 2.3.1.1.

2.3.3.2 CellSubResponseDataOptionalAttributes

The **CellSubResponseDataOptionalAttributes** attribute group contains attributes that MUST only be used in **SubResponseData** elements associated with a **SubResponse** for a cell sub request. **CellSubResponseDataOptionalAttributes** provide the data that was requested as part of the cell sub request. 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:attributeGroup>
```

Etag: A string that specifies 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 co-authorable file. **Etag** defines the file version and allows for the client to know the version of the file contents.

When the **Etag** attribute is specified as part of a response to a cell sub request, **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 sub request 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 section 14.19 of [RFC2616].

CreateTime: An integer that specifies the time at which the file was created and is expressed in tick count. A single tick represents one hundred nanoseconds or one ten-millionth of a second. **CreateTime** specifies the number of 100-nanosecond intervals that have elapsed since 12:00:00 midnight, January 1st, 1601 and 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 sub request.

LastModifiedTime: An integer that specifies the last modified time and is expressed in tick count. A single tick represents one hundred nanoseconds or one ten-millionth of a second. **LastModifiedTime** specifies the number of 100-nanosecond intervals that have elapsed since 12:00:00 midnight, January 1st, 1601 and 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 sub request.

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. The **CoalesceErrorMessage** also specifies information of what was expected by the protocol server. The **CoalesceErrorMessage** attribute MUST only be sent when the **CoalesceHResult** attribute is set to an integer value greater than 0.

2.3.3.3 CoauthSubRequestDataOptionalAttributes

The **CoauthSubRequestDataOptionalAttributes** attribute group contains attributes that MUST only be used for **SubRequestData** elements associated with parent **SubRequest** element for a co-authoring sub request. **CoauthSubRequestDataOptionalAttributes** are used as input parameters in processing the data associated with a co-authoring sub request. 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 co-authoring sub request. CoauthRequestTypes is defined in section 2.3.2.2.

Depending on the type of the co-authoring sub request, the following table shows a mapping between the type of co-authoring sub request and the attributes that MUST be specified for that **CoauthRequestType**.

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

Value of CoauthRequestType attribute	Timeout	AllowFallbackToExclusive	ExclusiveLockID	Release lock on conversion failure	ClientID	SchemalockID
JoinCoauthoring (Join co-authoring)	✓		If Allow fall back to exclusive is set to true, then the Exclusive Lock		✓	✓

Value of CoauthRequestType attribute	Timeout	AllowFallbackToExclusive	ExclusiveLockID	Release lock on conversion failure	Client ID	SchemalockID
			identifier attribute MUST be specified.			
ExitCoauthoring (Exit co-authoring)					✓	✓
RefreshCoauthoring (Refresh co-authoring)	✓				✓	✓
ConvertToExclusive (Convert to Exclusive lock co-authoring)	✓		✓	✓	✓	✓
CheckLockAvailability (Check lock availability)						✓
MarkTransitionComplete (Mark co-authoring transition complete)					✓	✓
GetCoauthoringStatus, (Get co-authoring status)					✓	✓

✓ **:** Specifies that the attribute MUST be specified as part of the SubRequestData element associated with the co-authoring sub request.

2.3.3.4 ExclusiveLockSubRequestDataOptionalAttributes

The **ExclusiveLockSubRequestDataOptionalAttributes** attribute group contains attributes that MUST only be used for **SubRequestData** elements associated with parent **SubRequest** element for an exclusive lock sub request. **ExclusiveLockSubRequestDataOptionalAttributes** are used as input parameters in processing the data associated with an exclusive lock sub request. 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 sub request. **ExclusiveLockRequestTypes** is defined in section 2.3.2.3.

Depending on the type of the exclusive lock sub request, the following table shows a mapping between the type of exclusive lock sub request 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 sub request depending on the type of exclusive lock sub request.

Value of ExclusiveLockRequestType attribute	Timeout	SchemaLockID	ClientID	ExclusivelockID
GetLock (Get lock)	✓			✓
ReleaseLock (Release lock)				✓
RefreshLock (Refresh lock)	✓			✓
ConvertToSchemaJoinCoauth (Convert to Schema lock with co-authoring transition tracked)	✓	✓	✓	✓
ConvertToSchema (Convert to Schema lock)	✓	✓	✓	✓
CheckLockAvailability (Check lock availability)				✓

✓ **:** Specifies that the attribute MUST be specified as part of the SubRequestData element associated with the exclusive lock sub request.

2.3.3.5 SchemaLockSubRequestDataOptionalAttributes

The **SchemaLockSubRequestDataOptionalAttributes** attribute group contains attributes that MUST only be used for **SubRequestData** elements associated with parent SubRequest element for a schema lock sub request. **SchemaLockSubRequestDataOptionalAttributes** are used as input parameters in processing the data associated with a schema lock sub request. 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 sub request. SchemaLockRequestTypes is defined in section 2.3.2.4.

Depending on the type of the schema lock sub request, the following table shows a mapping between the type of schema lock sub request and the attributes that MUST be specified for that **SchemaLockRequestType**.

In the following table, **Timeout**, **AllowFallbackToExclusive**, **ExclusiveLockID**, release lock on conversion failure, **ClientID**, and **SchemaLockID** are attributes that MUST be specified for the **SubRequestData** element associated with the co-authoring sub request depending on the type of co-authoring sub request

Value of SchemaLockRequestType attribute	Timeout	AllowFallbackToExclusive	ExclusiveLockID	Release lock on conversion failure	Client ID	SchemalockID
GetLock (Get lock)	✓		If Allow Fall back to exclusive is set to true, then the Exclusive Lock identifier attribute MUST be specified.		✓	✓
ReleaseLock (Release lock)					✓	✓
RefreshLock (Refresh lock)	✓				✓	✓
ConvertToExclusive (Convert to Exclusive lock)	✓		✓	✓	✓	✓
CheckLockAvailability (Check lock availability)						✓

✓ **:** Specifies that the attribute MUST be specified as part of the SubRequestData element associated with the schema lock sub request.

2.3.3.6 WhoAmISubResponseDataOptionalAttributes

The **WhoAmISubResponseDataOptionalAttributes** attribute group contains attributes that MUST only be used in **SubResponseData** elements associated with a sub response for a **WhoAmI** sub

request. **WhoAmISubResponseDataOptionalAttributes** provide the data that was requested as part of the **WhoAmI** sub request. 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: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 e-mail address associated with the client. Format of the e-mail addresses 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.

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] Status Code Definitions section 10.

This protocol allows protocol servers to notify protocol clients of request or sub request 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 specify the data organization that a protocol server maintains to participate in the protocol:

- **Partition-block:** The partition-block is a partition, which contains binary file data in an abstract model 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 the partition identifier.
- **File co-authoring tracker:** For each file, the protocol server keeps track of the client identifiers associated with each client that is co-authoring the file, the type of lock on the file, specifying an exclusive lock or shared lock, and the timeout associated with each client's shared lock on the co-authorable 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 operations of this protocol.

Method	Description
Cell sub request	Retrieves or uploads file's binary contents or file's metadata contents.
Coauth sub request	Gets a shared lock on a co-authorable 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.

Method	Description
SchemaLock sub request	Gets a shared lock on a co-authorable file that allows all clients with the same schema lock identifier to share the lock.
ExclusiveLock sub request	Gets an exclusive lock on the file which ensures only one client edits the file at an instant of time.
WhoAmI sub request	Retrieves the client **friendly name** and other client specific information for a client with a unique client identifier.
ServerTime sub request	Retrieves the server time.

3.1.4.1 Common Message Processing Rules and Events

The protocol server MUST follow the following common processing rules for all types of sub requests.

File URL: The **Url** attribute in the cell storage service **Request** element specifies the unique URL for the file for which the request is to be processed. The **Url** attribute is specified in section 2.2.3.2.

SubRequestToken: The **SubRequestToken** attribute in the cell storage service **SubRequest** element uniquely identifies a sub request for a file. The **SubRequestToken** attribute is specified in section 2.2.4.3.

Type: The **Type** attribute in the cell storage service **SubRequest** element specifies the type of cell storage service sub request. The protocol server uses the type attribute to identify the type of cell storage service sub request. The **Type** attribute is specified in section 2.2.4.2.

DependencyType: The **DependencyType** attribute in the cell storage service **SubRequest** element specifies the type of dependency that a sub request has on another sub request. **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 sub request is to be processed or not.

DependsOn: DependsOn attribute specifies the **SubRequestToken** of the sub request on which this sub request depends on. **DependsOn** attribute for a SubRequest element is specified in section 2.2.4.3.

The protocol server uses a combination of the **DependsOn** and the **DependencyType** attribute for a specific sub request to decide if a cell storage service sub request is executed or not. 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 in minimum, support one of the following sub requests:

- The co-authoring sub request.
- The schema lock sub request.

If the protocol server supports co-authoring sub request, it MUST support tracking co-authoring transition. Co-authoring transition allows for transition of numbers of users editing the file to increase from 1 to n users or decrease from n to 1 user, where n is the maximum number of users who are allowed for editing a single file at an instant of time. If the protocol server supports co-authoring sub request, it MUST return a co-authoring status as specified in section 2.3.1.7.

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

- A co-authoring sub request.
- A schema lock sub request.
- An exclusive lock sub request of type, "Convert to schema lock".
- An exclusive lock sub request of type, "Convert to schema lock with co-authoring transition tracked".

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

- An exclusive lock sub request.
- A co-authoring sub request of type, "Convert to exclusive lock".
- A schema lock sub request 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, then the protocol server MUST release that client's lock on the shared file. When the timeout for all clients holding a shared lock on the file expire, then the shared lock on the file is released by the protocol server<9>. 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 co-authorable files and non co-authorable files. The protocol server is also used by clients to synchronize file's metadata contents.

The protocol server returns an error code value set to "RequestNotSupported" for a cell storage service sub request if the following conditions are all true.

- The protocol client sent a co-authoring sub request.
- The protocol server supports shared locking with tracking of co-authoring transition.
- The co-authoring admin setting for the server is turned off.

The types of sub requests and protocol behavior are specified in the following sections.

3.1.4.2 Cell Sub Request

The operation is used to retrieve or upload binary or metadata contents of a file. The contents are uploaded or downloaded fully or partially. Download or upload of contents is accomplished by indicating the partition identifier whose contents are to be uploaded or downloaded. The cell sub request 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 **CellSubRequestType**, 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 sub request. 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 specified in [MS-FSSHTTPB] section 2.2.3.1.5. The cell sub request for upload of file contents is processed as specified in [MS-FSSHTTPB] section 3.1.4.5. The cell sub request for download of file contents is processed as specified in [MS-FSSHTTPB] section 3.1.4.2. The schema lock identifier MUST be specified in a cell sub request if the current client already shared the lock on the file and is in a co-authoring session. Schema lock identifier is defined in section 2.3.1.1. When the schema lock identifier is specified in the cell sub request the protocol server returns a success error code only if the file currently has a shared lock with the specified schema lock identifier.

The bypass lock identifier MUST be present when uploading binary or metadata contents of a file. It MUST NOT be present when retrieving binary or metadata contents of a file. Bypass lock identifier is defined in section 2.3.3.1. When the bypass lock identifier is specified in a cell sub request exactly one of the following is true.

- A schema lock identifier is specified in the cell sub request. In this case the bypass lock identifier MUST be the same as the schema lock identifier. They have the same semantics in this case.
- An exclusive lock identifier is specified in the cell sub request. In this case the bypass lock identifier MUST be the same as the exclusive lock identifier. They have the same semantics in this case. Exclusive lock identifier is defined in section 2.3.3.1.
- If neither exclusive lock identifier nor schema lock identifier are present in a cell sub request then the bypass lock identifier has the same semantics as the exclusive lock identifier. It MUST have the same value as the exclusive lock identifier if an exclusive lock identifier were to be present.

  If in a file content upload cell sub request the **ExpectNoFileExists** attribute is set to **true**, the **Etag** attribute MUST be an empty string. In this case, the protocol server MUST fail the cell sub request with a coherency error if, and only if, the file already exists on the server. **ExpectNoFileExists** attribute is defined in section 2.3.3.1. **Etag** attribute is defined in section 2.3.3.1. Coherency failure error codes are specified in [MS-FSSHTTPB]

  If the **Coalesce** attribute is set to **true** in a cell sub request, the protocol server ensures to persist all changes to the file with 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:

- **LockType** attribute is set to "ExclusiveLock" in the cell sub request, to request the protocol server to take an exclusive lock on the file.
- **ExclusiveLockID** attribute is set to a unique lock identifier.

The **Coalesce** attribute and **LockType** attribute are defined in section 2.3.3.1. **ExclusiveLockID** attribute is defined in section 2.3.1.1.

When the protocol server receives the **ExclusiveLockID** in a cell sub request, the server performs the following steps as an atomic operation:

- Get an exclusive lock on the file.
- Commit the file's binary contents, as specified in [MS-FSSHTTPB] section 3.1.4.5.

For updates to other partitions that do not contain binary file contents, the **Coalesce** attribute is set to false in the cell sub request.

- 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 sent back by the component responsible for implementing [MS-FSSHTTPB] are prepared as a response by the protocol server and sent 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 as defined in section 2.2.3.5 and the SubResponse element is as 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, and 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 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 "FileNotExistsOrCannotBeCreated" in the **ErrorCode** attribute sent back in the **SubResponse** element.
- 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 Co-auth Sub Request

The operation is used to request the protocol server for a shared lock on the file or for other shared lock related operations on a file. A protocol server that supports a co-authoring sub request MUST also support tracking co-authoring transition. Depending on the **CoauthRequestType** attribute value, the protocol server interprets the sub request as one of the following types of shared lock operation:

- Join co-authoring session.
- Exit co-authoring session.
- Refresh co-authoring session.
- Convert to Exclusive lock.
- Check lock availability.
- Mark transition to complete.
- Get co-authoring status.

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

The protocol client sends a co-authoring **SubRequest** message which is of **CoauthSubRequestType**, specified in section 2.3.1.6. The protocol server responds with a co-authoring **SubResponse** message which is of type, **CoauthSubResponseType**, 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 co-authoring sub request. 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 co-authoring request, the protocol server processes the sub request as specified in sections 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 and 3.1.4.3.7. The protocol server uses the **ClientID** attribute sent in a co-authoring sub request to do the following:

  - Uniquely identify each client

  - Keep track of each client and their timeout on the shared lock for the file.

  - Decide when to release the shared lock on the file.

    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 **CoauthSubResponseDataType** defines the type of the **SubResponseData** element inside the co-authoring **SubResponse** element. The **CoauthSubResponseDataType** is defined in section 2.3.1.7.

    If the co-authoring sub request is of type, "Join co-authoring session" or "Refresh co-authoring 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". Lock type is specified as **LockType** attribute in the co-authoring **SubResponseData** element. The **SubResponseData** element returned for a co-authoring sub request is of type, **CoauthSubResponseDataType**, and is defined in section 2.3.1.7. **LockType** attribute is specified in section 2.3.1.7.

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 "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 co-authoring sub request is one of the following types:

  - Join co-authoring session

  - Exit co-authoring session

  - Refresh co-authoring session

  - Convert to an exclusive lock

- Mark in transition complete

  A concurrency violation happens when a current client's request to save the file co-authoring tracker fails because another client's request to edit and save the file co-authoring 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 co-authoring tracker is defined in section 3.1.1.

- The protocol server returns an error code value set to "RequestNotSupported" if server does not support this request type.
- **ErrorCode** includes "Success" to indicate success in processing the co-authoring request.

The protocol behavior for each type of the co-authoring sub request is specified in the following sections.

3.1.4.3.1 Join Co-authoring Session

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

If the file is already having a shared lock on the server with the given schema lock identifier and the client already joined the co-authoring session, the protocol server does all the following:

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

If the co-authoring 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, then the protocol server returns an error code value set to "LockNotConvertedAsCoauthDisabled".

AllowFallbackToExclusive attribute is defined in section 2.3.1.5. The result of the lock type got 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 co-authoring status, the protocol server checks to see the number of clients editing the file at that instant of 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 co-author or third or greater co-author joining the co-authoring session, the protocol server MUST return a **CoauthStatus** set to "Coauthoring", which indicates that the current client is coauthoring when editing the doc. 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 co-authoring session, the server creates a new transition request using the **TransitionID** of 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 co-authoring session makes the **IsOnlyClient** web service request, the server will return "false".

Description of transition request and **IsOnlyClient** web service request is specified in [MS-SHDACCWS]. This sequence of sub requests is described in the figure in section 3.1.4.3.6.

If there is a current exclusive lock on the file or 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 co-authorable file is checked out on the server and is checked out by a client with a different user name than the current client, then 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 co-authorable clients allowed to join a co-authoring session to edit a co-authorable file is reached.
- The current client is not allowed to edit the file because the limit has been reached.

If any failure occurs such that the sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Errors returned by the protocol server directly are outside of the scope of this document. 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 Co-authoring Session

If the **CoauthRequestType** attribute is set to "ExitCoauthoring", the protocol server considers this as a co-authoring sub request of type, "Exit co-authoring session". The protocol server processes the request to exit the co-authoring session and checks to see the number of clients editing the doc at that instant of time. When the protocol server receives a co-authoring sub request of type "Exit co-authoring session" from the last and only client editing the doc, the protocol server does all of the following:

- Delete the client identifier entry associated with the client in the file co-authoring tracker. The file co-authoring tracker is defined in section 3.1.1.
- Delete the co-authoring session and the shared lock on the file, if the client that sent the sub request of type "Exit co-authoring session" is the last client in the file co-authoring tracker. If the current client is not already present in the co-authoring session, the protocol server does one of the following:
  - Returns error code "Success", if there are other clients present in the co-authoring session.
  - Returns error code "FileNotLockedOnServer", if no clients are present in the co-authoring session.

The protocol server SHOULD return error code <10>"FileAlreadyLockedOnServer" if there is a current exclusive lock on the file or there is a shared lock on the file with a different schema lock identifier.

If the co-authoring session is already deleted, the protocol server returns an error code value set to "Success". "Success" is defined in section 2.2.5.6.

If a failure occurs such that the sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Errors returned by the protocol server directly are outside of the scope of this document. **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 Co-authoring Session

If the **CoauthRequestType** attribute is set to "RefreshCoauthoring", the protocol server considers this as a co-authoring sub request of type, "Refresh co-authoring session". The protocol server refreshes the client's timeout of the shared lock on the co-authorable file and also checks the co-authoring 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 as the timeout value expired on the lock in the file co-authoring tracker, the protocol server does one of the following:

- It co-authoring feature is enabled on the protocol server, it considers this as co-authoring sub request of type, "Join co-authoring session" and gets a new shared lock on the file.
- If the co-authoring feature is disabled and the **AllowFallbackToExclusive** attribute is set to true, the protocol server gets an exclusive lock on the file.
- If the co-authoring feature is disabled and the **AllowFallbackToExclusive** attribute is set to false, then the protocol server returns an error code value set to "LockNotConvertedAsCouthDisabled".

The timeout attribute is defined in section 2.3.1.5 and the file co-authoring tracker is defined in section 3.1.1.

Once the protocol server has ensured that there is a shared lock with the same schema lock identifier as used by the current client and 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 co-authoring sub request. If the client is sending a request to refresh the shared lock with a timeout value lower than or equal to the current timeout on the shared lock, then the protocol server considers the co-authoring sub request of type, "Refresh co-authoring session" as 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, then the protocol server updates the file co-authoring tracker with the new timeout value for that specific client and specific file. The timeout attribute is defined in section 2.3.1.5 and the file co-authoring tracker is defined in section 3.1.1.

To get the co-authoring status, the protocol server checks to see the number of clients editing the doc at that instant of 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 or greater than second client trying to edit the doc, the protocol server MUST return a **CoauthStatus** set to "Coauthoring", which indicates that the current client is coauthoring when editing the doc. 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 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 co-authorable file is checked out on the server and is checked out by a client with a different user name than the current client, then the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer".

If a failure occurs such that the sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification.

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 as a co-authoring sub request of type, "Convert to Exclusive lock". The protocol server processes the request to convert a shared lock on the co-authorable file to an exclusive lock on the file.

The protocol server performs the following two operations:

- Conversion of the shared lock to an exclusive lock.
- Deletion of coauthoring session.

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.
- There is no co-authoring session for the file.
- The current client is not present in the co-authoring session.

If there is a current exclusive lock on the file or there is 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 doc. If the shared lock is successfully converted to an exclusive lock, the protocol server MUST use the unique **ExclusiveLockID** attribute sent by the client to identify the lock. **ExclusiveLockID** attribute is specified in section 2.3.1.5.

If there is more than one client currently editing the file and **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 all **true**:

- **ReleaseLockOnConversionToExclusiveFailure** attribute is set to **true** in the co-authoring sub request.
- Multiple clients are in the co-authoring session.

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

If any failure occurs such that the sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification. **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 as a co-authoring sub request 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 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 co-authorable file is checked out on the server and is checked out by a client with a different user name than the current client, then 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 as a co-authoring sub request 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 co-authoring sub request of type, "Mark transition to complete" is requested by a protocol client that first joined a co-authoring session or a protocol client that has a co-authoring 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 specified in [MS-SHDACCWS].

The following is a sequence diagram that specifies a sequence in which a co-authoring sub request of type, "Mark transition to complete" is sent by client1 (C1). C1 is the first client that joined the co-authoring session to edit the file. Client2 (C2) is the second client sharing the lock and joining the co-authoring session.

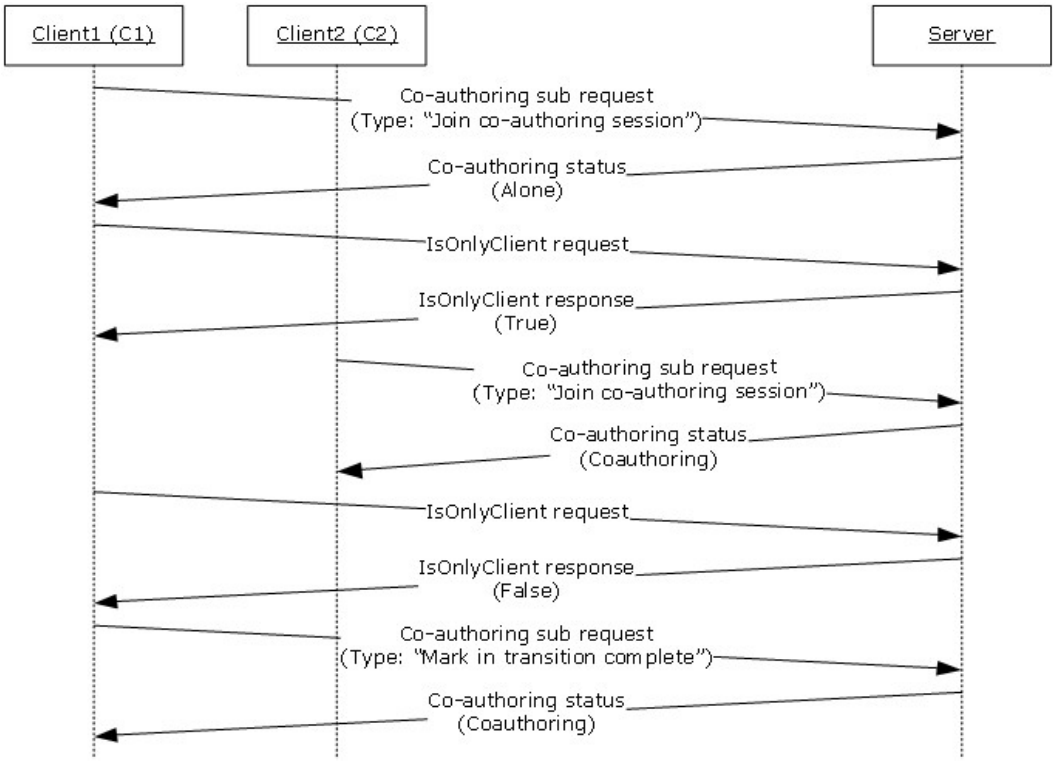

Figure 3: Sequence of co-authoring sub request types

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

The **CoauthStatus** attribute is not set by the server in the sub response returned for this sub request. The client MUST set its local co-authoring status to "Coauthoring", if the **ErrorCode** attribute in the sub response is set to "Success" to indicate the successful processing of this sub request.

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 co-authoring session for the file.
- The current client is not present in the co-authoring session.

If a failure occurs such that the sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification. **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 Co-authoring Status

If the **CoauthRequestType** attribute is set to "GetCoauthoringStatus", the protocol server considers this as a co-authoring sub request of type, "Get co-authoring status". The protocol server checks to get the co-authoring status of the co-authorable file. The protocol client keeps sending co-authoring sub request of type, "Get co-authoring status" while it is actively co-authoring to check if it is the only client editing the file. If the count of number of clients editing the file goes back to one, the protocol client stops sending the get co-authoring status request and sends the **IsOnlyClient** web service request as specified in [MS-SHDACCWS].

If the co-authoring session does not exist or the current client is not present in the co-authoring session, the protocol server returns an error code value set to "InvalidCoauthSession". If the current client is the only client editing the co-authorable file, the protocol server MUST set the **CoauthStatus** attribute value to "Alone", indicating no one else is editing the file. If the current client is the second or greater than second client trying to edit the doc, the protocol server MUST return a **CoauthStatus** set to "Coauthoring", which indicates that the current client is coauthoring when editing the doc.

If any failure occurs such that the sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification. Generic error code types are defined in section 2.2.5.6.

3.1.4.4 SchemaLock Sub Request

The operation is used to request the protocol server for a shared lock on a file or for different types of shared lock operations on a co-authorable file. The schema lock sub request is conceptually a sub set of the co-authoring sub request and differs from the co-authoring sub request by not keeping tracks of the clients currently in the co-authoring session that are sharing the lock on the file.

The name schema lock represents that all clients with the same schema lock identifier share the lock and clients with a different schema lock identifier are required to wait until the shared lock is released by all 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 operation:

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

The **SchemaLockRequestType** attribute is defined in section 2.3.2.4. The **SubRequestData** element for schema lock sub request 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 **SchemaLockSubRequestType**, specified in section 2.3.1.14. The protocol server responds with a schema lock **SubResponse** message which is of type, **SchemaLockSubResponseType**, 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 sub request. 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 sub request, the protocol sever process the request as specified in sections 3.1.4.4.1, 3.1.4.4.2, 3.1.4.4.3, 3.1.4.4.4, and 3.1.4.4.5. The protocol server uses the **ClientID** attribute sent in a schema lock sub request to uniquely identify each client, 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 sub request 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:

  - When the timeout expires for a client, and no refresh on the timeout was received, then the protocol server MUST release that client's lock on the shared file. When the timeout for all clients holding a shared lock on the file expire, then 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 sub request 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". Lock type is sent as **LockType** attribute in the schema lock **SubResponseData** element. The **SubResponseData** element returned for a schema lock sub request is of type, **SchemaLockSubResponseDataType**, and is defined in section 2.3.1.15. **LockType** attribute is specified in section 2.3.1.15. The **LockType** attribute values are specified in section 2.2.5.9.

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 "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 sub request 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 co-authoring tracker fails because another client's request to edit and save the file co-authoring 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 co-authoring tracker is defined in section 3.1.1.

- The protocol server returns an error code value set to "RequestNotSupported" if server does not support this request type.
- **ErrorCode** includes "Success" to indicate success in processing the schema lock request.

The protocol behavior for each type of the schema lock sub request is specified in the following sections.

3.1.4.4.1 Get Lock

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

If the file already has a shared lock on the server with the given schema lock identifier and the client already joined the co-authoring session, the protocol server does all the following:

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

If the co-authoring 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**, then the protocol server returns an error code value set to "LockNotConvertedAsCouthDisabled".

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 co-authorable clients allowed to join a co-authoring session to edit a co-authorable file is 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 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 co-authorable file is checked out on the server and is checked out by a client with a different user name than the current client, then the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer".

If a failure occurs such that the sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification. **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 as a schema lock sub request of type, "Release lock". The protocol server processes the request to exit the co-authoring session and checks to see the number of clients editing the doc at that instant of time. When the protocol server receives a schemaLock sub request of type "ReleaseLock" from the last and only client editing the doc, the protocol server does all of the following:

- Delete the client identifier entry associated with the client in the file co-authoring tracker. The file co-authoring tracker is defined in section 3.1.1.
- Delete the co-authoring session and the shared lock on the file, if the client that sent the sub request of type "Release lock" is the last client in the file co-authoring tracker.

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

- Returns error code "Success", if there are other clients present in the co-authoring session.
- Returns error code "FileNotLockedOnServer", if no clients are present in the co-authoring session.

The protocol server SHOULD<11> return error code "FileAlreadyLockedOnServer" if there is a current exclusive lock on the file or there is a shared lock on the file with a different schema lock identifier.

If the co-authoring session is already deleted, the protocol server returns an error code value set to "Success". "Success" is defined in section 2.2.5.6.

If any failure occurs such that the sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification. **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 as a schema lock sub request of type, "Refresh lock". The protocol server refreshes the client's timeout of the shared lock on the co-authorable file.

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

- It the co-authoring feature is enabled on the protocol server, it considers this a schema lock sub request of type, "Get lock" and gets a new shared lock on the file.
- If the co-authoring feature is disabled and the **AllowFallbackToExclusive** attribute is set to true, the protocol server gets an exclusive lock on the file.
- If the co-authoring feature is disabled and if the **AllowFallbackToExclusive** attribute is set to false, then the protocol server returns an error code value set to "LockNotConvertedAsCouthDisabled".

Once the protocol server has ensured that there is a shared lock with the same schema lock identifier as used by the current client and 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 sub request. If the client is sending a request to refresh the shared lock with a timeout value lower than the current timeout on the shared lock, then the protocol server considers the co-authoring sub request of type, "Refresh lock" as a no-operation. The timeout attribute is defined in section 2.3.1.13.

If there is a current exclusive lock on the file or 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 co-authorable file is checked out on the server and 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, the appropriate 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 as a schema lock sub request of type, "Convert to exclusive lock". The protocol server process the request to convert a shared lock on the co-authorable file to an exclusive lock on the file.

The protocol server performs the following operations:

- Conversion of the shared lock to an exclusive lock.
- Deletion of 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 co-authoring session for the file. The current client is not present in the co-authoring session.
- If there is a current exclusive lock on the file or there is 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 doc. If the shared lock is successfully converted to an exclusive lock, the protocol server MUST use the unique **ExclusiveLockID** attribute sent by the client to identify the lock. **ExclusiveLockID** attribute is specified in section 2.3.1.13.

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

- **ReleaseLockOnConversionToExclusiveFailure** attribute is set to true in the co-authoring sub request.

- Multiple clients are in the co-authoring session.

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

If any failure occurs such that the sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification. **LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8 and **GenericErrorCodeTypes** is 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 as a schema lock sub request 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 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 co-authorable file is checked out on the server and is checked out by a client with a different user name than the current client, then 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 Sub Request

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

- 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. **SubRequestData** element for an exclusive lock sub request 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 **ExclusiveLockSubRequestType**, specified in section 2.3.1.10. The protocol server responds with an exclusive lock **SubResponse** message which is of type, **ExclusiveLockSubResponseType** and is 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 sub request. The **SubRequestData** element is of type, **ExclusiveLockSubRequestDataType** and is defined in section 2.3.1.9.

- The protocol server receives the request and parses the logic and depending on the type of exclusive lock sub request, the protocol server processes the request as specified in sections 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, and 3.1.4.5.6.

  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 **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 "FileNotExistsOrCannotBeCreated" in the **ErrorCode** attribute sent back in the **SubResponse** element.

- If the protocol server gets an exclusive lock sub request of type, "Convert to schema lock with co-authoring 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".

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

  - Convert to schema lock with co-authoring transition tracked.

  - Convert to schema lock.

    A concurrency violation happens when a current client's request to save the file co-authoring tracker fails because another client's request to edit and save the file co-authoring 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 co-authoring tracker is defined in section 3.1.1.

- **ErrorCode** includes "Success" to indicate success in processing the exclusive lock request.

The protocol behavior for each type of the exclusive lock sub request is specified in the following sections.

3.1.4.5.1 Get Lock

If the **ExclusiveLockRequestType** attribute is set to "GetLock", the protocol server considers this as an exclusive lock sub request 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:

- File is already locked with an exclusive lock with a different exclusive lock identifier.
- 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 sub request 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 check out of files, then the protocol server returns an error code value set to "DocumentCheckoutRequired". "DocumentCheckoutRequired" error code value indicates to the protocol client that a checkout needs to be done before the file is locked. The checkout of the file is done by the client using the **CheckoutFile** web service call defined in [MS-LISTSWS].

If the protocol server encounters issue in locking the file because a checkout is done by another client, the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer". If the checkout of the file is 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 sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification. 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.

3.1.4.5.2 Release Lock

If the **ExclusiveLockRequestType** attribute is set to "ReleaseLock", the protocol server considers this as an exclusive lock sub request 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**:

- File is already locked with an exclusive lock with a different exclusive lock identifier.
- 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 sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification.
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 as an exclusive lock sub request 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 being checked out on the server and the file is saved to a document library that requires check out files, then the protocol server returns an error code value set to "DocumentCheckoutRequired". "DocumentCheckoutRequired" error code value indicates to the protocol client that a checkout needs to be done before the file is to be locked and the lock is 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 is already 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 sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification.
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 Co-authoring Transition Tracked

If the **ExclusiveLockRequestType** attribute is set to "ConvertToSchemaJoinCoauth", the protocol server considers this as an exclusive lock sub request of type, "Convert to schema lock with co-authoring transition tracked". When the protocol server receives this sub-request, it does all of the following:

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

Once the request to convert the exclusive lock to a shared lock is processed successfully, the protocol server gets the co-authoring status and returns the status to the client.

The protocol server uses the **ClientID** attribute sent in an exclusive lock sub request of type, "Convert to schema lock with co-authoring transition tracked", to uniquely identify each client, keep track of each client and their timeout on the shared lock for the file. The protocol server also uses the **ClientID** sent in the exclusive lock sub request to decide when to release the shared lock on the file. The protocol server uses the **SchemaLockID** attribute sent in an exclusive lock sub request of

type, "Convert to schema lock with co-authoring transition tracked", to ensure that once the exclusive lock on the file is converted to a shared lock, the protocol server MUST only allow other clients with the same schema lock identifier to share the lock on the file. **SchemaLockID** and **ClientID** attributes are defined in section 2.3.1.9.

- If the feature of transitioning a file that is currently being exclusively locked to a shared lock is not supported by the protocol server, then the protocol server returns an error code value set to "RequestNotSupported".
- If the co-authoring feature is disabled by the protocol server, then 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 being 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 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 is present on the file, existing on the server, then 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 check out files and the document is not being checked out on the server, then the protocol server returns an error code value set to "DocumentCheckoutRequired". "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 co-authoring status, the protocol server checks to see the number of clients editing the doc at that instant of 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 not the only client editing the document, the protocol server MUST return a **CoauthStatus** 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** is represented as the **CoauthStatus** attribute for the **SubResponseData** element. The **CoauthStatus** attribute sent as part of the **SubResponseData** element is defined in section 2.2.8.2.

If any failure occurs such that the sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification.
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 as an exclusive lock sub request 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 sub request is a subset of the "Convert to schema lock with co-authoring transition tracked" type of exclusive lock sub request. The "Convert to schema lock with co-authoring transition tracked" type of sub request converts the exclusive lock on the file to a shared lock and also keeps track of all client user information that are editing the file concurrently. The "Convert to schema lock" type of sub request converts the exclusive lock on the file to a shared lock and does not keep track of co-authoring transition. The "Convert to schema lock" type and "Convert to schema lock with co-authoring transition tracked" type of sub requests both add the client identifiers to the file co-authoring tracker.

The protocol server uses the **ClientID** attribute sent in an exclusive lock sub request of type, "Convert to schema lock", to uniquely identify each client, 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.

- If the feature of transitioning a file that is currently being exclusively locked to a shared lock is not supported by the protocol server, then the protocol server returns an error code value set to "RequestNotSupported".
- If the feature of co-authoring is completely not supported by the protocol server, then the protocol server returns an error value set to "LockNotConvertedAsCoauthDisabled".
- If the protocol server is unable to convert the exclusive lock on the file to a shared lock because the file is being 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 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 because no lock exists on the server, then 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 check out files and the document is not being checked out on the server, then the protocol server returns an error code value set to "DocumentCheckoutRequired". "DocumentCheckoutRequired" error code value indicates to the protocol client that a checkout needs to be done before the file is exclusively locked and 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 trying to convert 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 sub request cannot be processed successfully, the protocol server returns an error. The protocol server can return any error the implementer chooses as appropriate. Error codes specific to the protocol server are out of scope for this specification.
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 as an exclusive lock sub request 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 with a different exclusive lock identifier than the one specified by the current client or there is a shared lock on the file, then the protocol server returns an error code value set to "FileAlreadyLockedOnServer".
- If the file is checked out on the server and 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.

If the protocol server encounters unknown exceptions or failures in processing the exclusive lock sub request of type, "Check lock availability", the protocol server returns an error code value that is set to "LockRequestFail".

3.1.4.6 WhoAmI Sub Request

This operation is used to retrieve current client user information that helps in showing a client friendly name when a co-authorable file is edited by more than one client.

The protocol client sends a **WhoAmI** SubRequest message which is of **WhoAmISubRequestType**, specified in section 2.3.1.20. The protocol server responds with a **WhoAmI** SubResponse message of type, **WhoAmISubResponseType**, 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 for client specific user information. The client specific user information requested are prepared as a response and sent back to the protocol client.

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 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 client specific user information requested, as attributes in the **WhoAmI SubResponseData** element. The attributes sent by the protocol server are defined in section 2.3.3.6.

The protocol returns results based on the following conditions:

- If the processing of the **WhoAmI** sub request by the protocol server failed to get the client specific user information or hit an unknown exception, the protocol server returns an error code value set to "SubRequestFail".
- Otherwise, the protocol server set the error code value to "Success" to indicate success in processing the **WhoAmI** sub request.

3.1.4.7 ServerTime Sub Request

The operation is used to retrieve server time, expressed in tick count (number of 100-nanosecond intervals) that have elapsed since 12:00:00 midnight, January 1st, 0001. Server time SHOULD<12> be expressed in Coordinated Universal Time (UTC).

The protocol client sends a **ServerTime SubRequest** message which is of **ServerTimeSubRequestType**, specified in section 2.3.1.17. The protocol server responds with a

ServerTime SubResponse message which is of type, **ServerTimeSubResponseType**, 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 server time information requested is prepared as a response and sent back to the protocol client.

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 **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. **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** sub request by the server failed to get the server time or hit 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** sub request.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

4 Protocol Examples

This section provides common scenarios involving usage of the different types of sub request operations for synchronization of file's contents or file metadata contents.

The following example shows the request message schema sent by a protocol client.

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

The **Body** element of each SOAP request message contains a **RequestVersion** element and a **RequestCollection** element. Details of the **RequestVersion** element are specified in section 2.2.3.4. Each **RequestCollection** element contains one or more **Request** elements. Each **Request** element contains one or more **SubRequest** elements. Details of the **RequestCollection** element are specified in section 2.2.3.3 and details of the **Request** element are specified in section 2.2.3.2. Details of the **SubRequest** element are specified in section 2.2.3.8 and details of the **SubRequestElementGenericType** are specified in section 2.2.4.2.

The following example shows the response message schema sent by a protocol server.

```
<?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" />

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

The **Body** element of each SOAP response message contains a **ResponseVersion** element and a **ResponseCollection** element. Details of the **ResponseVersion** element are specified in section 2.2.3.7. Each **ResponseCollection** element contains one or more **Response** elements. Each **Response** element contains one or more **SubResponse** elements. Details of the **ResponseCollection** element are specified in section 2.2.3.6 and details of the **Response** element are specified in section 2.2.3.5. Details of the **SubResponse** element are specified in section 2.2.3.10 and details of the **SubResponseElementGenericType** are specified in section 2.2.4.5.

While key elements and attributes are described in detail in each of the following scenarios, some elements and attributes are not described completely, for the sake of brevity and readability of the example.

4.1 Successful File Open of a Co-authorable Document

A client wants to open a file on a protocol server. This file is a co-authorable document. The client successfully opens the co-authorable document by sending a series of sub requests to initiate a download of the file contents for shared editing.

Protocol server: Example

Source file to be opened: 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 sub request to the protocol server and is uniquely identified by the

SubRequestToken attribute. Details of the **SubRequestToken** attribute are specified in section 2.2.4.3. The type of the sub request 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 co-authoring sub request of type "Join co-authoring session" that requests a shared lock on the file and the co-authoring status. Details of the co-authoring sub request of type "Join co-authoring session" are specified in section 3.1.4.3.1.

The second **SubRequest** element is a schema lock sub request of type "Get lock" that also requests a shared lock on the file. Unlike the first **SubRequest** element, the co-authoring sub request, the schema lock sub request does not get the co-authoring status. This schema lock sub request is executed only if the first **SubRequest** element, the co-authoring sub request, is not supported by the protocol server. The dependency of the schema lock sub request on the first **SubRequest** element is defined by the **DependsOn** attribute and the **DependencyType** attribute of the schema lock sub request. In this case, the **DependsOn** value of 1 specifies the **SubRequestToken** of the first **SubRequest** element, on which the second element is dependent. The **DependencyType** value of "OnNotSupported" specifies that the second schema lock sub request gets called only if the first **SubRequest** element is not supported. Details of the **DependsOn** and **DependencyType** attributes are specified in section 2.2.4.3. Details of the schema lock sub request of type "Get lock" are specified in section 3.1.4.4.1.

The third, fourth, and fifth **SubRequest** elements are cell sub requests that request the download of file contents or file metadata contents. All three cell sub requests 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 sub request 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 from which the file contents or file metadata contents are downloaded. The binary data sent in the **SubRequestData** element of a cell sub request indicates if this is a file content upload or download request. The cell sub request for download of file contents is processed as specified 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 of the **PartitionID** attribute and **GetFileProps** attribute are specified in section 2.3.3.1. Details of the cell sub request are specified in section 3.1.4.2.

The sixth **SubRequest** element is a **ServerTime** sub request that gets server time information, as specified in section 3.1.4.7.

The seventh **SubRequest** element is a **WhoAmI** sub request that requests the current client user information. This client user information helps in showing a client friendly name when a co-authorable file is edited by more than one client. Details of the **WhoAmI** sub request 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/eVQjrABAAAAAAAAAVgIAAAAAAAAAAAAAAAAAAAAAAAAFVQ4CBgADBQD6AiQADLX1p
FfYJdlCjxOPdqBs0jsAhABBBwGLAQ==</SubResponseData>
        </SubResponse>
        29d
        <SubResponse SubRequestToken="4" ErrorCode="Success" HResult="0">
          <SubResponseData Etag="&amp;quot;{600CE272-068F-4BD7-A1FB-4AC10C54386C},1&amp;quot;"
CoalesceHResult="0" ContainsHotboxData="False" CreateTime="129092725940000000"
LastModifiedTime="129092725940000000" ModifiedBy="Jayne
Darcy">DAALAJ3PKfM5lAabFgMCAACsAgAMVgxdgv0sL6qXQpGLEnn1TeJGgEXxEmphk/RDliumkvzDIGwBAAAAAAAAAA
MFVQ4CBgADBQD6AiQADF2C/SwvqpdCkYsSefVN4kYAhAAmAiAA9jV6MmEHFESWhlHpAGZ6TaQAeCRNZ9PslLQ+v5Vxq4R
eeFt+AHN4JLKYLBNrS8FAlXGrhF54W34Aa1ETASYCIAATHwkQgsj7QJiGZTP5NMIdbAFwLQz5C0E3b9GZRKbDJyMu3KcR
CTIAAAC1EwEmAiAADul2OjKADE253fPGUClDPkwBICYMspgsE2tLwUCVcauEXnhbfmsApRMBQQcBiwE=</SubResponse
Data>
        </SubResponse>
        1bc
        <SubResponse SubRequestToken="3" ErrorCode="Success" HResult="0">
          <SubResponseData CoalesceHResult="0"
ContainsHotboxData="True">DAALAJ3PKfM5lAabFgMCAACsAgAMVgx6KmdPAwAaQpf2Pb9Nlj2NgCminTMNs9JDuty
BtjY1HeoBAAAAAAAAAAOIVAxZA/FHJtJ3R6iV1MCVAWthgNcfQoujYe9GnWDh5152y9MCAAAAAAAAAAUMVgxZA/FHJtJ3
R6iV1MCVAWthgNcfQoujYe9GnWDh5152y9MBAAAAAAAAAAVgIAAAAAAAAAAAAAAAAAAAAAAAAFVQ4CBgADBQD6AiQADHoqZ
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 sub request. 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 co-authoring sub request of type "Join co-authoring session". The **ErrorCode** attribute of "Success" indicates that the co-authoring sub request 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 co-authoring 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 co-authoring session and is editing the file. Details of 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 sub request of type "Get Lock". The **ErrorCode** attribute of "DependentOnlyOnNotSupportedRequestGetSupported", defined in section 2.2.5.2, indicates that the co-authoring sub request on which this schema lock sub request is dependent was supported by the protocol server, and so the schema lock sub request 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 sub requests. The **ErrorCode** attributes in each **SubResponse** element of "Success" indicates that all three cell sub requests were successfully processed. The third and fifth **SubResponse** elements contain the requested file contents or file metadata contents. The server uses **chunked transfer encoding** to send the response. The fourth **SubResponse** element contains the file properties requested as part of the corresponding cell sub request. 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 of the **Etag** attribute are specified in section 2.3.3.2.

The format of the binary data values of the **SubResponseData** elements is described by [MS-FSSHTTPB] section 2.2.3.1.5.

The **SubResponse** element with a **SubRequestToken** set to 6 is the response for the **ServerTime** sub request. The **ErrorCode** attribute of "Success" indicates that the **ServerTime** sub request 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** sub request. The **ErrorCode** attribute of "Success" indicates that the **WhoAmI** sub request 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 Co-authorable Document

A client wants to save a file that has been edited back to the protocol server. This file is a co-authorable document. The client successfully saves the co-authorable document by sending a series of sub requests to initiate an upload of the file contents.

Protocol server: Example

Source file to be saved: 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 co-authoring sub request of type "Refresh co-authoring session", which requests a refresh of the client's timeout of the shared lock and an update of the co-authoring status. Details of the co-authoring sub request of type "Refresh co-authoring session" are specified in section 3.1.4.3.3.

The second **SubRequest** element is a schema lock sub request of type "Refresh lock", which also requests a refresh of the client's timeout of the shared lock on the file. This schema lock sub request is executed only if the first **SubRequest** element, the co-authoring sub request, is not supported by the protocol server because the **DependencyType** attribute is set to "OnNotSupported". Details of the schema lock sub request of type "Refresh lock" are specified in section 3.1.4.4.3.

The third **SubRequest** element is a cell sub request that requests the upload of file contents or file metadata contents. Because the DependencyType attribute for this sub request is set to "OnSuccessOrNotSupported", it is executed if either the co-authoring sub request or the schema lock sub request were successfully executed or both these sub requests were 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 versioning of the file for co-authoring. This setting of the **CoauthVersioning** attribute helps prevent the protocol server from doing multiple version updates in a short period of time. Details of the **Coalesce** attribute and **CoauthVersioning** attribute are specified in section 2.3.3.1.

The **Include** element within the **SubRequestData** element of the cell sub request is used for encapsulating and sending large amounts of binary data. Details of the **Include** element are specified in section 2.2.3.1. The cell sub request for upload of file contents is processed as specified in [MS-FSSHTTPB] section 3.1.4.5. Details of the cell sub request 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="&amp;quot;{600CE272-068F-4BD7-A1FB-4AC10C54386C},2&amp;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 sub request from the protocol client, which was of type "Refresh co-authoring session". The **ErrorCode** attribute of "Success" indicates that the co-authoring sub request 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 is "SchemaLock", in the **LockType** attribute, and that the current co-authoring status is "Alone", in the **CoauthStatus** attribute.

The second **SubResponse** element is the response to the schema lock sub request of type "Refresh Lock".

The third **SubResponse** element is the response from the protocol server to the cell sub request. The **ErrorCode** attribute of "Success" indicates that the cell sub request for uploading of file contents or file metadata contents was successfully processed. The other attributes and elements of this **SubResponse** element are described in the previous example. The format of the value of the **SubResponseData** element is described in [MS-FSSHTTPB] section 2.2.3.1.5.

4.3 Successful File Open of a Non-Co-authorable Document

A client wants to open a file on a protocol server. This file is a non-co-authorable document. The client successfully opens the non-co-authorable document by sending two sub requests to initiate a download of the file contents for exclusive editing.

Protocol server: Example

Source file to be opened: 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 sub request 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 of the **ExclusiveLockID** attribute are specified in section 2.3.1.9. Details of the exclusive lock sub request of type "Get lock" are specified in section 3.1.4.5.1.

The second **SubRequest** element is a cell sub request that requests the download of file contents or file metadata contents and file properties. Details of the cell sub request 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="&amp;quot;{901072DF-38C4-4211-BA16-0A882E036AB1},1&amp;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 sub request of type "Get lock". The **ErrorCode** attribute of "Success" indicates that the co-authoring sub request 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 sub request. The **ErrorCode** of "Success" indicates that the cell sub request for 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 sub request is used for encapsulating and sending large amounts of binary data. Details of the **Include** element are specified in section 2.2.3.1.

4.4 Unsuccessful File Open of a Non-Co-authorable Document

A client wants to open a file on the protocol server. This file is a non-co-authorable document. The client sends two sub requests to initiate a download of the file contents for exclusive editing. The client fails to open the non-co-authorable document for exclusive editing, because the file already is exclusively locked by another client.

Protocol server: Example

Source file to be opened: 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 sub request of type "Get lock" that requests an exclusive lock on the file and provides an **ExclusiveLockID** of "{9BCE3023-0F1F-496B-A561-610144B54040}". Details of the exclusive lock sub request of type "Get lock" are specified in section 3.1.4.5.1.

The second **SubRequest** element is a cell sub request that requests the download of file contents or file metadata contents and file properties. Details of the cell sub request 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="&amp;quot;{901072DF-38C4-4211-BA16-0A882E036AB1},2&amp;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 sub request 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 sub

request that failed and gives more hints on the cause of the failure. Details of the **ErrorMessage** attribute and **HResult** attribute are specified in section 2.2.4.6.

Although the first sub request failed, the second sub request is executed by the server as the DependencyType attribute on the second sub request is set to "OnExecute". The second **SubResponse** element is the response from the protocol server for the cell sub request. The **ErrorCode** of "Success" indicates that the cell sub request for downloading of file contents and file properties were successfully processed. In this example, because the exclusive lock sub request of type "Get lock" failed but the cell sub request 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 Non-Co-authorable Document

A client wants to save a file that it has edited back to the protocol server. This file is a non-co-authorable document. The client successfully saves the non-co-authorable document by sending two sub requests to initiate an upload of the file contents.

Protocol server: Example

Source file to be saved: 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 sub request of type "Refresh lock" that requests a refresh of the client's timeout of the exclusive lock on the file. Details of the exclusive lock sub request of type "Refresh lock" are specified in section 3.1.4.5.3.

The second **SubRequest** element is a cell sub request that requests the upload of file contents or file metadata contents. This cell sub request is executed only if the first **SubRequest** element, the

exclusive lock sub request, succeeded or is not supported by the protocol server. The dependency of the cell sub request on the first **SubRequest** element is defined by the **DependsOn** attribute and the **DependencyType** attribute of the cell sub request. In this case, the **DependsOn** value of 1 specifies the **SubRequestToken** of the first **SubRequest** element, on which the second element is dependent. The **DependencyType** value of "OnSuccessOrNotSupported" specifies that the second cell sub request gets called only if the first **SubRequest** element succeeded or is not supported, but not if it failed for a different reason. Details of the cell sub request 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="&amp;quot;{901072DF-38C4-4211-BA16-0A882E036AB1},2&amp;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 sub request of type "Refresh lock". The **ErrorCode** attribute of "Success" indicates that the exclusive lock sub request 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 sub request. The **ErrorCode** attribute of "Success" indicates that the cell sub request for 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 Co-authorable Document

A client wants to open a co-authorable document for co-authoring from a protocol server. The client fails to open the document for co-authoring 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 sub request ("Coauth") fails because the maximum number of coauthors has been reached.

Protocol server: Example

Source file to be opened: 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 sub requests 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="&amp;quot;{18EA7896-87E5-4FF2-8B56-
D6932D37A920},2&amp;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">DAALAJ3PKfM5lAabFgMCAACsAgAMVgwiNmq4KENARZbfdZYMxE0zgNV311Vft
JZJo4z7N5ngOMcBAAAAAAAAAAOINACAAYc8L30CEECd7/lCoJY2CAEAAAAAAAAAABVUOAgYAAwUA+gIkAAwiNmq4KENARZ
bfdZYMxE0zAIQAQQcBiwE=</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" sub request. It shows the "Coauth" sub request failed with **ErrorCode** "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

There are no security considerations that are specific to this protocol.

This protocol does not introduce any additional security considerations beyond those that apply to its underlying protocols.

5.2 Index of Security Parameters

None.

6 Appendix A: Message Schemas

The request message schema and response message schema are provided in the following sections.

6.1 Request Message Schema

The following specifies the protocol request 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" />
 <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="0"/>
    </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: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>

  <!-- 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: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: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>

  <!--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="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>

  <!--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="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

The following specifies the protocol response 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" />

<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="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="0"/>
    </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"/>
  </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="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="CoauthStatusType">
    <xs:restriction base="xs:string">
      <!--Alone -->
      <xs:enumeration value="Alone"/>
      <!--Coauthoring-->
      <xs:enumeration value="Coauthoring"/>
    </xs:restriction>
  </xs:simpleType>
```

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

  <xs:simpleType name="UserLoginType">
    <xs:restriction base="xs:string">
      <xs:pattern value="^[a-zA-Z]([a-zA-Z0-9\-_])*\\[a-zA-Z]([a-zA-Z0-9])*"/>
    </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: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: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="CoalesceHResult" type="xs:integer" use="required" />
    <xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
    <xs:attribute name="ContainsHotboxData" type="xs:boolean" use="required" />
    <xs:attribute name="HaveOnlyDemotionChanges" type="xs:boolean" use="required" />
</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:attribute name="ExclusiveLockReturnReason" type="tns:ExclusiveLockReturnReasonTypes"
use="optional" />
  </xs:complexType>

  <xs:complexType name="WhoAmISubResponseDataType">
    <xs:attributeGroup ref="tns:WhoAmISubResponseDataOptionalAttributes"/>
    <xs:attribute name="UserLogin" type="tns:UserLoginType" use="required"/>
  </xs:complexType>

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

  <xs:complexType name="SubResponseDataGenericType" mixed="true">
     <xs:all>
      <xs:element ref="i:Include" minOccurs="0" maxOccurs="1" />
     </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="1"
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>

  <!--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: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: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® SharePoint® Foundation 2010
- Microsoft® SharePoint® Workspace 2010
- Microsoft® SharePoint® Server 2010

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 2.2.8.1: Word 2010 and PowerPoint 2010 use string "29358EC1-E813-4793-8E70-ED0344E7B73C" for the **SchemaLockID** attribute sent in the different sub requests of type **Coauthoring**, **ExclusiveLock** and **SchemaLock**.

<2> Section 2.2.8.2: Office 2010 servers may return a time value that is not the current time. The protocol client does not change behavior because the protocol client uses the differences between **ServerTime** values, and not differences between the **ServerTime** and the time at the client.

<3> 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 sub requests of type **Coauthoring**, **ExclusiveLock** and **SchemaLock**.

<4> Section 2.3.1.5: Word 2010 and PowerPoint 2010 use string "29358EC1-E813-4793-8E70-ED0344E7B73C" for the **SchemaLockID** attribute sent in the different sub requests of type **Coauthoring**, **ExclusiveLock**, and **SchemaLock**.

<5> Section 2.3.1.9: Word 2010 and PowerPoint 2010 use string "29358EC1-E813-4793-8E70-ED0344E7B73C" for the **SchemaLockID** attribute sent in the different sub requests of type **Coauthoring**, **ExclusiveLock**, and **SchemaLock**.

<6> 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 sub requests of type **Coauthoring**, **ExclusiveLock**, and **SchemaLock**.

<7> Section 2.3.1.18: Office 2010 servers may return a time value that is not the current time. The protocol client does not change behavior because the protocol client uses the differences between **ServerTime** values, and not differences between the **ServerTime** and the time at the client.

<8> Section 2.3.3.1: Microsoft Office 2010 servers will save all changes to the underlying store at the end of processing a sub request with the following four exceptions.

1. The sub request is a **Put Changes** sub request as defined in section 2.2.2.1.5 of [MS-FSSHTTPB]. And the **Partial** bit of the Put Changes request is 1. The **Partial** bit is defined in section 2.2.2.1.5 of [MS-FSSHTTPB]. In this case Microsoft Office 2010 servers will not save changes to the underlying store. Instead the changes will be stored in a write cache.

2. The Microsoft Office 2010 servers will write to an intermediate write cache while processing a sub request if the sub request is a **Put Changes** sub request as defined in section 2.2.2.1.5 of [MS-FSSHTTPB]. If writing to the write cache failed then Microsoft Office 2010 servers will not save changes to the underlying store.

3. If prior to the current attempt to save changes to the underlying store there have been 26 to 73 consecutive failed attempts immediately preceding and the most recent failed attempt is within 60 minutes of the current sub request then Microsoft Office 2010 servers will not save changes to the underlying store. Instead the changes will be 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 sub request then Microsoft Office 2010 servers will not save changes to the underlying store. Instead the changes will be stored in a write cache.

<9> Section 3.1.4.1: Office 2010 applications set the lock timeout interval to 3600 seconds and send a request to refresh the timeout at a regular interval of 2700 seconds.

<10> Section 3.1.4.3.2: Office 2010 release servers have a different behavior. Office 2010 release server returns 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 co-authoring session with one of more clients in it. It returns return error code "FileAlreadyLockedOnServer" if there is a shared lock with a different shared lock identifier and a co-authoring session with no clients in it.

<11> Section 3.1.4.4.2: Office 2010 release servers returns 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 co-authoring session with one of more clients in it. It returns return error code "FileAlreadyLockedOnServer" if there is a shared lock with a different shared lock identifier and a co-authoring session with no clients in it.

<12> Section 3.1.4.7: Office 2010 servers may return a time value that is not the current time. The protocol client does not change behavior because the protocol client uses the differences between **ServerTime** values, and not differences between the **ServerTime** and the time at the client.

8 Change Tracking

No table of changes is available. The document is either new or has had no changes since its last release.

9 Index

F

G

I

L

M

T

U

V

W