Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation ("this documentation") for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.

- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.

- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.

- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft Open Specifications Promise or the Microsoft Community Promise. If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.

- **License Programs.** To see all of the protocols in scope under a specific license program and the associated patents, visit the Patent Map.

- **Trademarks.** The names of companies and products contained in this documentation might be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights. For a list of Microsoft trademarks, visit www.microsoft.com/trademarks.

- **Fictitious Names.** The example companies, organizations, products, domain names, email addresses, logos, people, places, and events that are depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

**Support.** For questions and support, please contact dochelp@microsoft.com.
## Revision Summary

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision History</th>
<th>Revision Class</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>7/13/2009</td>
<td>0.1</td>
<td>Major</td>
<td>Initial Availability</td>
</tr>
<tr>
<td>8/28/2009</td>
<td>0.2</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>11/6/2009</td>
<td>0.3</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>2/19/2010</td>
<td>1.0</td>
<td>Major</td>
<td>Updated and revised the technical content</td>
</tr>
<tr>
<td>3/31/2010</td>
<td>1.01</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>4/30/2010</td>
<td>1.02</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>6/7/2010</td>
<td>1.03</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>6/29/2010</td>
<td>1.04</td>
<td>Editorial</td>
<td>Changed language and formatting in the technical content.</td>
</tr>
<tr>
<td>7/23/2010</td>
<td>1.05</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>9/27/2010</td>
<td>1.06</td>
<td>Editorial</td>
<td>Changed language and formatting in the technical content.</td>
</tr>
<tr>
<td>11/15/2010</td>
<td>1.06</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>12/17/2010</td>
<td>1.06</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>3/18/2011</td>
<td>1.06</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>6/10/2011</td>
<td>1.06</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>1/20/2012</td>
<td>2.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>4/11/2012</td>
<td>2.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>7/16/2012</td>
<td>2.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>9/12/2012</td>
<td>2.1</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>10/8/2012</td>
<td>2.2</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>2/11/2013</td>
<td>3.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>7/30/2013</td>
<td>3.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>11/18/2013</td>
<td>3.2</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>2/10/2014</td>
<td>3.3</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>4/30/2014</td>
<td>3.4</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>7/31/2014</td>
<td>3.5</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>10/30/2014</td>
<td>3.6</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>3/16/2015</td>
<td>4.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>Date</td>
<td>Revision History</td>
<td>Revision Class</td>
<td>Comments</td>
</tr>
<tr>
<td>----------</td>
<td>------------------</td>
<td>----------------</td>
<td>--------------------------------------------------------------------------</td>
</tr>
<tr>
<td>6/30/2015</td>
<td>5.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>2/26/2016</td>
<td>6.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>4/14/2016</td>
<td>7.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>7/15/2016</td>
<td>7.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>8/23/2016</td>
<td>7.0</td>
<td>None</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>9/7/2016</td>
<td>7.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>12/15/2016</td>
<td>7.2</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>3/28/2017</td>
<td>8.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>6/20/2017</td>
<td>9.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>9/19/2017</td>
<td>9.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>12/12/2017</td>
<td>9.2</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>7/24/2018</td>
<td>10.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>10/1/2018</td>
<td>11.0</td>
<td>Major</td>
<td>Significantly changed the technical content.</td>
</tr>
<tr>
<td>12/11/2018</td>
<td>11.1</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>3/19/2019</td>
<td>11.2</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>9/24/2019</td>
<td>11.3</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
</tbody>
</table>
# Table of Contents

1 **Introduction** .......................................................................................................................... 9  
1.1 Glossary ................................................................................................................................. 9  
1.2 References .............................................................................................................................. 10  
1.2.1 Normative References ....................................................................................................... 11  
1.2.2 Informative References ..................................................................................................... 12  
1.3 Overview .................................................................................................................................. 12  
1.4 Relationship to Other Protocols .............................................................................................. 14  
1.5 Prerequisites/Preconditions .................................................................................................... 14  
1.6 Applicability Statement .......................................................................................................... 14  
1.7 Versioning and Capability Negotiation .................................................................................... 14  
1.8 Vendor-Extensible Fields ........................................................................................................ 15  
1.9 Standards Assignments .......................................................................................................... 15  

2 **Messages** .................................................................................................................................. 16  
2.1 Transport ................................................................................................................................. 16  
2.2 Common Message Syntax ....................................................................................................... 16  
2.2.1 Namespaces ....................................................................................................................... 16  
2.2.2 Messages ........................................................................................................................... 16  
2.2.2.1 Request ......................................................................................................................... 17  
2.2.2.2 Response ....................................................................................................................... 17  
2.2.2.3 SOAP Fault ................................................................................................................ 18  
2.2.3 Elements ............................................................................................................................ 18  
2.2.3.1 Include ......................................................................................................................... 19  
2.2.3.2 Request ......................................................................................................................... 19  
2.2.3.3 RequestCollection ....................................................................................................... 23  
2.2.3.4 RequestVersion ........................................................................................................... 23  
2.2.3.5 Response ..................................................................................................................... 24  
2.2.3.6 ResponseCollection .................................................................................................... 25  
2.2.3.7 ResponseVersion ........................................................................................................ 25  
2.2.3.8 SubRequest .................................................................................................................. 26  
2.2.3.9 SubRequestData .......................................................................................................... 26  
2.2.3.10 SubResponse ............................................................................................................. 26  
2.2.3.11 SubResponseData ...................................................................................................... 27  
2.2.4 Complex Types .................................................................................................................. 27  
2.2.4.1 GenericPropertiesType ............................................................................................... 27  
2.2.4.2 PropertyType ............................................................................................................... 28  
2.2.4.3 SubRequestDataGenericType .................................................................................... 28  
2.2.4.4 SubRequestElementGenericType ............................................................................. 29  
2.2.4.5 SubRequestType .......................................................................................................... 31  
2.2.4.6 SubResponseDataGenericType .................................................................................. 32  
2.2.4.7 SubResponseElementGenericType .......................................................................... 33  
2.2.4.8 SubResponseType ...................................................................................................... 35  
2.2.4.9 VersionType ................................................................................................................ 36  
2.2.5 Simple Types ...................................................................................................................... 37  
2.2.5.1 CoauthStatusType ....................................................................................................... 38  
2.2.5.2 DependencyCheckRelatedErrorCodeTypes ............................................................. 39  
2.2.5.3 DependencyTypes ....................................................................................................... 39  
2.2.5.4 ErrorCodeTypes ......................................................................................................... 40  
2.2.5.5 ExclusiveLockReturnReasonTypes ............................................................................ 41  
2.2.5.6 GenericErrorCodeTypes ............................................................................................. 41  
2.2.5.7 GUID ........................................................................................................................... 43  
2.2.5.8 LockAndCoauthRelatedErrorCodeTypes ................................................................. 43  
2.2.5.9 LockTypes ................................................................................................................... 45  
2.2.5.10 MinorVersionNumberType ....................................................................................... 46  
2.2.5.11 SubRequestAttributeType ......................................................................................... 47
2.2.5.12 TRUEFALSE ................................................................. 49
2.2.5.13 VersionNumberType ................................................. 49
2.2.5.14 NewEditorsTableCategoryErrorCodeTypes ......................... 49
2.2.5.15 FileVersionNumberType ........................................... 50
2.2.5.16 VersioningRelatedErrorCodeTypes ................................ 50
2.2.6 Attributes ................................................................. 51
2.2.7 Groups ...................................................................... 51
2.2.8 Attribute Groups .......................................................... 51
2.2.8.1 SubRequestDataOptionalAttributes .................................. 51
2.2.8.2 SubResponseDataOptionalAttributes .................................. 54
2.2.9 Common Data Structures .................................................. 55
2.3 Subsidiary Message Syntax ............................................... 55
2.3.1 Complex Types .............................................................. 55
2.3.1.1 CellSubRequestDataType .............................................. 58
2.3.1.2 CellSubRequestType .................................................... 59
2.3.1.3 CellSubResponseDataType ........................................... 59
2.3.1.4 CellSubResponseType .................................................. 60
2.3.1.5 CoauthSubRequestDataType ......................................... 60
2.3.1.6 CoauthSubRequestType ................................................ 62
2.3.1.7 CoauthSubResponseDataType ....................................... 63
2.3.1.8 CoauthSubResponseType ............................................. 63
2.3.1.9 ExclusiveLockSubRequestDataType ................................ 64
2.3.1.10 ExclusiveLockSubRequestType ..................................... 65
2.3.1.11 ExclusiveLockSubResponseDataType .............................. 65
2.3.1.12 ExclusiveLockSubResponseType .................................. 66
2.3.1.13 SchemaLockSubRequestDataType .................................. 66
2.3.1.14 SchemaLockSubRequestType ....................................... 68
2.3.1.15 SchemaLockSubResponseDataType ................................. 68
2.3.1.16 SchemaLockSubResponseType ..................................... 69
2.3.1.17 ServerTimeSubRequestType ........................................ 69
2.3.1.18 ServerTimeSubResponseDataType ................................. 70
2.3.1.19 ServerTimeSubResponseType ..................................... 70
2.3.1.20 WhoAmISubRequestType ............................................. 70
2.3.1.21 WhoAmISubResponseDataType ..................................... 71
2.3.1.22 WhoAmISubResponseType .......................................... 71
2.3.1.23 EditorsTableSubRequestDataType ................................ 71
2.3.1.24 EditorsTableSubRequestType ...................................... 73
2.3.1.25 EditorsTableSubResponseType ..................................... 73
2.3.1.26 GetDocMetaInfoSubRequestType .................................. 74
2.3.1.27 GetDocMetaInfoSubResponseDataType ............................ 74
2.3.1.28 GetDocMetaInfoSubResponseType .................................. 74
2.3.1.29 GetDocMetaInfoSubResponseType ................................. 75
2.3.1.30 GetDocMetaInfoSubResponseType ................................. 75
2.3.1.31 GetVersionsSubRequestType ....................................... 75
2.3.1.32 GetVersionsSubResponseDataType ................................ 76
2.3.1.33 FileOperationSubRequestDataType ................................ 76
2.3.1.34 FileOperationSubRequestType ..................................... 76
2.3.1.35 FileOperationSubResponseType .................................... 77
2.3.1.36 VersioningSubRequestDataType ................................... 77
2.3.1.37 VersioningSubRequestType .......................................... 78
2.3.1.38 VersioningSubResponseDataType .................................. 78
2.3.1.39 VersioningSubResponseType ........................................ 79
2.3.1.40 VersioningUserTableType ........................................... 79
2.3.1.41 VersioningVersionListType ........................................ 79
2.3.1.42 UserDataType .......................................................... 80
2.3.1.43 FileVersionDataType ................................................ 80
2.3.1.44 FileVersionEventDataType ........................................... 81
2.3.1.45 AmIAIAloneSubRequestDataType ................................... 81
3 Protocol Details ......................................................................................................................... 104

3.1 Server Details ...................................................................................................................... 104
3.1.1 Abstract Data Model ........................................................................................................... 104
3.1.2 Timers ................................................................................................................................. 104
3.1.3 Initialization ......................................................................................................................... 104
3.1.4 Message Processing Events and Sequencing Rules .............................................................. 104
3.1.4.1 Common Message Processing Rules and Events .............................................................. 105
3.1.4.2 Cell Subrequest ................................................................................................................ 106
3.1.4.3 Coauth Subrequest .......................................................................................................... 108
  3.1.4.3.1 Join Coauthoring Session .......................................................................................... 110
  3.1.4.3.2 Exit Coauthoring Session .......................................................................................... 111
  3.1.4.3.3 Refresh Coauthoring Session ...................................................................................... 111
  3.1.4.3.4 Convert to Exclusive Lock .......................................................................................... 112
  3.1.4.3.5 Check Lock Availability ............................................................................................. 113
  3.1.4.3.6 Mark Transition to Complete ...................................................................................... 113
  3.1.4.3.7 Get Coauthoring Session ............................................................................................ 114
3.1.4.4 SchemaLock Subrequest ..................................................................................................... 115
  3.1.4.4.1 Get Lock ..................................................................................................................... 117
  3.1.4.4.2 Release Lock ............................................................................................................... 117
  3.1.4.4.3 Refresh Lock .............................................................................................................. 118
  3.1.4.4.4 Convert to Exclusive Lock .......................................................................................... 119

2.3.2 Simple Types .................................................................................................................... 86
  2.3.2.1 CoauthRequestTypes ...................................................................................................... 87
  2.3.2.2 CoauthRequestTypes ...................................................................................................... 88
  2.3.2.3 ExclusiveLockRequestTypes ........................................................................................ 89
  2.3.2.4 SchemaLockRequestTypes ............................................................................................ 89
  2.3.2.5 EditorsTableRequestTypes ............................................................................................ 90
  2.3.2.6 UserLoginType .............................................................................................................. 91
  2.3.2.7 UserNameType .............................................................................................................. 91
  2.3.2.8 FileOperationRequestTypes .......................................................................................... 91
  2.3.2.9 VersioningRequestTypes ............................................................................................... 92
  2.3.2.10 PropertiesRequestTypes ............................................................................................... 92

2.3.3 Attribute Groups ............................................................................................................... 93
  2.3.3.1 CellSubRequestDataOptionalAttributes .................................................................... 93
  2.3.3.2 CellSubResponseDataOptionalAttributes .................................................................. 95
  2.3.3.3 CoauthSubRequestDataOptionalAttributes ................................................................ 96
  2.3.3.4 ExclusiveLockSubRequestDataOptionalAttributes ..................................................... 98
  2.3.3.5 SchemaLockSubRequestDataOptionalAttributes ........................................................ 99
  2.3.3.6 WhoAmISubResponseDataOptionalAttributes .......................................................... 100
  2.3.3.7 EditorsTableSubRequestDataOptionalAttributes ....................................................... 100
  2.3.3.8 FileOperationSubRequestDataOptionalAttributes ..................................................... 101
  2.3.3.9 VersioningSubRequestDataOptionalAttributes .......................................................... 102
  2.3.3.10 PropertiesSubRequestDataOptionalAttributes .......................................................... 102
3.1.4.4.5 Check Lock Availability .................................................................119
3.1.4.5 ExclusiveLock Subrequest ........................................................................120
3.1.4.5.1 Get Lock ..............................................................................................121
3.1.4.5.2 Release Lock .......................................................................................121
3.1.4.5.3 Refresh Lock .......................................................................................122
3.1.4.5.4 Convert to Schema Lock with Coauthoring Transition Tracked .............122
3.1.4.5.5 Convert to Schema Lock .......................................................................124
3.1.4.5.6 Check Lock Availability .......................................................................125
3.1.4.6 WhoAmI Subrequest ................................................................................125
3.1.4.7 ServerTime Subrequest ...........................................................................126
3.1.4.8 EditorsTable Subrequest .........................................................................126
3.1.4.8.1 Join Editing Session ............................................................................127
3.1.4.8.2 Leave Editing Session .........................................................................128
3.1.4.8.3 Refresh Editing Session .......................................................................128
3.1.4.8.4 Update Editor Metadata ......................................................................128
3.1.4.8.5 Remove Editor Metadata .....................................................................128
3.1.4.9 GetDocMetaInfo Subrequest ..................................................................128
3.1.4.10 GetVersions Subrequest .......................................................................129
3.1.4.11 Versioning Subrequest .........................................................................130
3.1.4.11.1 Get Version List ................................................................................130
3.1.4.11.2 Restore Version ................................................................................131
3.1.4.12 FileOperation Subrequest .....................................................................131
3.1.4.13 AmIAlone Subrequest ..........................................................................132
3.1.4.14 LockStatus Subrequest .........................................................................132
3.1.4.15 Properties Subrequest .........................................................................133
3.1.4.15.1 Property Enumerate .........................................................................133
3.1.4.15.2 Property Get .....................................................................................134
3.1.5 Timer Events ...........................................................................................134
3.1.6 Other Local Events ..................................................................................134

4 Protocol Examples ..........................................................................................135
4.1 Successful File Open of a Coauthorable Document .....................................135
4.1.1 Request .....................................................................................................135
4.1.2 Response ..................................................................................................136
4.2 Successful File Save of a Coauthorable Document .................................138
4.2.1 Request .....................................................................................................138
4.2.2 Response ..................................................................................................139
4.3 Successful File Open of a Document that Is Not Coauthorable ....................140
4.3.1 Request .....................................................................................................140
4.3.2 Response ..................................................................................................141
4.4 Unsuccessful File Open of a Document that Is Not Coauthorable ..............142
4.4.1 Request .....................................................................................................142
4.4.2 Response ..................................................................................................142
4.5 Successful File Save of a Document that Is Not Coauthorable ....................143
4.5.1 Request .....................................................................................................143
4.5.2 Response ..................................................................................................144
4.6 Unsuccessful File Open of a Coauthorable Document .................................145
4.6.1 Request .....................................................................................................145
4.6.2 Response ..................................................................................................146

5 Security ..........................................................................................................147
5.1 Security Considerations for Implementers ......................................................147
5.2 Index of Security Parameters ........................................................................147

6 Appendix A: Full XML Schema ....................................................................148
6.1 Request Message Schema ............................................................................148
6.2 Response Message Schema .........................................................................157

7 Appendix B: Product Behavior .......................................................................168
8 Change Tracking..............................................................................................................173
9 Index................................................................................................................................174
1 Introduction

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

Sections 1.5, 1.8, 1.9, 2, and 3 of this specification are normative. All other sections and examples in this specification are informative.

1.1 Glossary

This document uses the following terms:

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

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

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

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

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

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

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

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

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

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

**Session Initiation Protocol (SIP) address**: A URL that does not include a "sip:" prefix and is used to establish multimedia communications sessions between two or more users over an IP network, as described in [RFC3261].
**SOAP**: A lightweight protocol for exchanging structured information in a decentralized, distributed environment. SOAP uses XML technologies to define an extensible messaging framework, which provides a message construct that can be exchanged over a variety of underlying protocols. The framework has been designed to be independent of any particular programming model and other implementation-specific semantics. SOAP 1.2 supersedes SOAP 1.1. See [SOAP1.2-1/2003].

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

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

**Subrequest**: A request within a SYNC_VOLUMES request. For details on requests, see section 3.1.4.

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

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

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

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

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

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

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

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

### 1.2 References

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

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


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


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


1.2.2 Informative References

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


1.3 Overview

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

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

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

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

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

In an upload or download request, the protocol allows for locking operations to be requested by the protocol client to the protocol server. The locking operations can be for an exclusive lock or a shared lock on a file. In the case of an exclusive lock, the protocol server ensures that only one client is allowed to edit the file and responds with success in locking the file for edit. For more details about the exclusive lock operation, see section 3.1.4.5. In the case of a shared lock, the protocol server allows multiple clients to edit the coauthorable file and responds with success in sharing the lock on the coauthorable file. Depending on the type of shared lock operation, the protocol server also keeps tracks of the clients editing the file and lets the protocol client know of the coauthoring status. For more details about the coauthoring status, see section 2.2.5.1. For more details about the shared lock operations, see section 3.1.4.3 and section 3.1.4.4.
In case of failure in an exclusive lock request or shared lock request, the protocol server responds with an error code value indicating the type of error. For more details about error code types, see section 2.2.5.4.

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

![Diagram of file upload and download to and from a server as well as lock requests to a server](image)

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

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

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

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

This protocol uses the **Simple Object Access Protocol (SOAP)** message protocol for formatting request and response messages, as described in [SOAP1.1], [SOAP1.2-1/2007] and [SOAP1.2-2/2007]. It transmits those messages by using **Hypertext Transfer Protocol (HTTP)**, as described in [RFC2616].

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

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

The protocol operates against a protocol server that is identified by a URL that is known by protocol clients. The protocol server endpoint is formed by appending "/_vti_bin/cellstorage.svc" to the URL of the protocol server. An example is http://www.contoso.com/_vti_bin/cellstorage.svc. The whole document URL is formed by appending "/_vti_bin/cellstorage.svc" to the URL of the file. An example is http://www.contoso.com/shared%20documents/test1.docx/_vti_bin/cellstorage.svc. The protocol server accepts the whole document URL in the first request of every scenario, and returns canonical URLs to use instead of the protocol server endpoint and the whole document URL, in **WebUrl** attribute specified in section 2.2.3.6 and **Url** attribute in section 2.2.3.5.

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

1.6 Applicability Statement

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

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

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

1.7 Versioning and Capability Negotiation

None.
1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.
2 Messages

2.1 Transport

For transport, this protocol requires the following:

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

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

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

2.2 Common Message Syntax

This section contains common definitions that are used by this protocol. The syntax of the definitions uses XML schema, as specified in [XMLSCHEMA1/2] and [XMLSCHEMA2/2], and WSDL, as specified in [WSDL].

2.2.1 Namespaces

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

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Namespace URI</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td><a href="http://schemas.xmlsoap.org/soap/envelope/">http://schemas.xmlsoap.org/soap/envelope/</a></td>
<td>[SOAP1.1]</td>
</tr>
<tr>
<td>xs</td>
<td><a href="http://www.w3.org/2001/XMLSchema">http://www.w3.org/2001/XMLSchema</a></td>
<td>[XMLSCHEMA1]</td>
</tr>
<tr>
<td>tns</td>
<td><a href="http://schemas.microsoft.com/sharepoint/soap/">http://schemas.microsoft.com/sharepoint/soap/</a></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td><a href="http://www.w3.org/2004/08/xop/include">http://www.w3.org/2004/08/xop/include</a></td>
<td>[XOP10]</td>
</tr>
</tbody>
</table>

2.2.2 Messages

<table>
<thead>
<tr>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>The detail element of the protocol request contains a RequestVersion element and a RequestCollection element.</td>
</tr>
<tr>
<td>Response</td>
<td>The detail element of the protocol response contains a ResponseVersion element and zero or one ResponseCollection elements.</td>
</tr>
</tbody>
</table>
### 2.2.2.1 Request

The protocol request schema is specified by the following:

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

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

### 2.2.2.2 Response

The protocol response schema is specified by the following:

```xml
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:tns="http://schemas.microsoft.com/sharepoint/soap/">
  attributeFormDefault="unqualified" elementFormDefault="qualified"
  targetNamespace="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:i="http://www.w3.org/2004/08/xop/include"/>
<xs: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:element>
      </xs:element>
      <xs:element>
      </xs:element>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
The **Body** element of each SOAP response message MUST contain a **ResponseVersion** element and zero or more **ResponseCollection** elements. Details about the **ResponseVersion** element are specified in section 2.2.3.7, and details about the **ResponseCollection** element are specified in section 2.2.3.6.

### 2.2.2.3 SOAP Fault

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

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

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

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

**ErrorString**: A string specifying the description of the error.

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

### 2.2.3 Elements

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

<table>
<thead>
<tr>
<th>Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Include</strong></td>
<td>The <strong>Include</strong> element is used for encapsulating and sending large amounts of binary data. The details about when an <strong>Include</strong> element is sent as part of a cell storage service request message and the details about when an <strong>Include</strong> element is sent as part of a cell storage service response message are specified in section 2.2.3.1.</td>
</tr>
<tr>
<td><strong>Request</strong></td>
<td>The <strong>Request</strong> element contains a cell storage service request that is specific to a <strong>URL</strong> for the file with a unique identifier that identifies the request by using a <strong>request token</strong>.</td>
</tr>
<tr>
<td><strong>RequestCollection</strong></td>
<td>The <strong>RequestCollection</strong> element contains a collection of <strong>Request</strong> elements.</td>
</tr>
<tr>
<td>Element</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>RequestVersion</td>
<td>The RequestVersion element specifies the version-specific information about the cell storage service request message.</td>
</tr>
<tr>
<td>Response</td>
<td>The Response element contains a cell storage service response that is specific to a URL for the file with a mapping response for the respective RequestToken.</td>
</tr>
<tr>
<td>ResponseVersion</td>
<td>The ResponseVersion element specifies the version-specific information about the cell storage service response message.</td>
</tr>
<tr>
<td>SubRequest</td>
<td>The SubRequest element specifies subrequests within a Request element.</td>
</tr>
<tr>
<td>SubRequestData</td>
<td>The SubRequestData element specifies the data required for processing the subrequest.</td>
</tr>
<tr>
<td>SubResponse</td>
<td>The SubResponse element specifies the corresponding response for the subrequest.</td>
</tr>
<tr>
<td>SubResponseData</td>
<td>The SubResponseData element contains any data requested as part of the subrequest.</td>
</tr>
</tbody>
</table>

### 2.2.3.1 Include

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

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

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

The SubRequestData element is specified in section 2.2.3.9. The SubResponseData element is specified in section 2.2.3.11. The Type attribute is specified in section 2.2.4.4. The Include element and the schema of the Include element are as specified in [XOP10] section 2.1. XML-binary Optimized Packaging (XOP) provides a means for more efficiently serializing an XML Information Set (Infoset) that has content types as specified in [XMLINFOSET].

### 2.2.3.2 Request

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

```xml
<xs:element name="Request">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="GenericProperties" type="tns:GenericPropertiesType" minOccurs="0" maxOccurs="unbounded" />
      <xs:element name="SubRequest" type="tns:SubRequestElementGenericType" minOccurs="1" maxOccurs="unbounded" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

---

[MS-FSSHTTP] - v20190924  
File Synchronization via SOAP over HTTP Protocol  
Copyright © 2019 Microsoft Corporation  
Release: September 24, 2019
GenericProperties: A GenericPropertiesType that specifies the generic properties of the request. The GenericPropertiesType is defined in section 2.2.4.1.

SubRequest: A SubRequestElementGenericType that specifies the type of subrequest for the URL for the file and input parameters needed by a protocol server for processing the subrequest. The SubRequest element is defined in section 2.2.3.8. The SubRequestElementGenericType is defined in section 2.2.4.4.

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

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

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
<th>K</th>
<th>L</th>
<th>M</th>
<th>Reserved</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Automatically generated request.</td>
</tr>
<tr>
<td>1</td>
<td>User action–generated request.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>User absent.</td>
</tr>
<tr>
<td>1</td>
<td>User present.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The protocol client has no local cache.</td>
</tr>
<tr>
<td>1</td>
<td>The protocol client has a local cache.</td>
</tr>
</tbody>
</table>
### D - Coauthoring Presence (2-bits): A 2-bit unsigned integer that specifies the protocol client’s awareness of other protocol clients’ coauthoring.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No other authors.</td>
</tr>
<tr>
<td>1</td>
<td>Other authors detected far from where the protocol client is working.</td>
</tr>
<tr>
<td>2</td>
<td>Other authors detected in the same section where the protocol client is working.</td>
</tr>
<tr>
<td>3</td>
<td>Other authors detected outside the same page where the protocol client is working.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Future user action in a new session.</td>
</tr>
<tr>
<td>1</td>
<td>Future user action in the current session.</td>
</tr>
<tr>
<td>2</td>
<td>Automatically after a short delay.</td>
</tr>
<tr>
<td>3</td>
<td>Immediately.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Recently.</td>
</tr>
<tr>
<td>1</td>
<td>Within one week.</td>
</tr>
<tr>
<td>2</td>
<td>Within one month.</td>
</tr>
<tr>
<td>3</td>
<td>More than one month.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Active.</td>
</tr>
<tr>
<td>1</td>
<td>Inactive.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The data is content and requires permanent reliable storage.</td>
</tr>
<tr>
<td>1</td>
<td>The data is recoverable. Less reliable stores can be used.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Unspecified or unknown.</td>
</tr>
<tr>
<td>1</td>
<td>Low-cost network.</td>
</tr>
<tr>
<td>2</td>
<td>Medium-cost network.</td>
</tr>
<tr>
<td>3</td>
<td>High-cost network.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No quota system.</td>
</tr>
<tr>
<td>Value</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------</td>
<td>---------</td>
</tr>
<tr>
<td>1</td>
<td>The quota system is active; the quota consumed is one half or less.</td>
</tr>
<tr>
<td>2</td>
<td>The quota system is active; the quota consumed is from one half through three quarters.</td>
</tr>
<tr>
<td>3</td>
<td>The quota system is active; the quota consumed is near or over the limit.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Powered by a continuous power source.</td>
</tr>
<tr>
<td>1</td>
<td>Powered by battery.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Protocol client is functioning normally.</td>
</tr>
<tr>
<td>1</td>
<td>Protocol client is not functioning normally.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>License type is unspecified or unknown.</td>
</tr>
<tr>
<td>1</td>
<td>License type is full.</td>
</tr>
<tr>
<td>2</td>
<td>License type is limited.</td>
</tr>
<tr>
<td>3</td>
<td>License type is free.</td>
</tr>
</tbody>
</table>

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

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

**UserAgent:** A guid that specifies a unique identifier of a user agent.\(\text{<3}\)\)

**UserAgentClient:** A string that specifies a user agent client.\(\text{<4}\)\)

**UserAgentPlatform:** A string that specifies a user agent platform.\(\text{<5}\)\)

**Build:** A string that specifies a user agent build number which represents as a four-part string with the following format: \(<\text{major version}>\.\text{<minor version>}\.\text{<build number>}\.\text{<revision>}\>\). For example, version 1.5.1254.0 indicates 1 as the major version, 5 as the minor version, 1254 as the build number, and 0 as the revision number.\(\text{<6}\)\)

Errors that occur during the parsing of the Request element cause the error code value to be set to "InvalidArgument". The protocol server MUST send the error code as an error code attribute in the Response element. The ErrorCode attribute is defined in section 2.2.3.5. Depending on the other types of errors, the error code for that type MUST be returned by the protocol server. Generic error code types are defined in section 2.2.5.6.

**ParentFolderResourceID:** If UseResourceID is true, the server creates a file in the given folder ResourceID, regardless of the request Url value.\(\text{<7}\)\) This attribute SHOULD only be used for the file content upload cell subrequest. If the folder with the specified ParentFolderResourceID does not exist, the server will fall back to use the Url value.
**ShouldReturnDisambiguatedFileName:** If an upload request fails with a coherency failure, this flag specifies whether the host will return a suggested/available file name that the client can try instead.<sup>8</sup>

**ResourceId:** A string that specifies the invariant ResourceID for a file that uniquely identifies the file whose response is being generated.<sup>9</sup> A ResourceID MUST NOT change over the lifetime of a file, even if the URL of the file changes. Attribute SHOULD NOT be present when the UseResourceID attribute is set as false.

**UseResourceID:** A TRUEFALSE value that specifies if the protocol server will perform ResourceID specific behavior for the file whose contents or metadata contents are requested for uploading to the server or downloading from the server.<sup>10</sup> When true, the protocol server SHOULD set the ResourceID attribute on the corresponding Response element to the ResourceID of the file. Also when true and the ResourceID attribute is set on the Request element, the protocol server SHOULD use the value of the ResourceID attribute to identify the file instead of the Url attribute. In the case where the protocol server is using the ResourceID attribute but the ResourceID attribute does not identify a valid file the protocol server SHOULD set an error code in the ErrorCode attribute of the corresponding Response attribute.

### 2.2.3.3 RequestCollection

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

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

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

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

Errors that occur during the parsing of the RequestCollection element MUST return a SOAP fault message. The SOAP fault message is defined in section 2.2.3.

### 2.2.3.4 RequestVersion

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

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

**VersionType** is specified in section 2.2.4.9.

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

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

```xml
<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="UrlIsEncoded" type="tns:TRUEFALSE" use="required"/>
    <xs:attribute name="RequestToken" type="xs:nonNegativeInteger" use="optional"/>
    <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:attribute name="SuggestedFileName" type="xs:string" use="optional"/>
    <xs:attribute name="ResourceId" type="xs:string" use="optional"/>
    <xs:attribute name="IntervalOverride" type="xs:nonNegativeInteger" use="optional"/>
  </xs:complexType>
</xs:element>
```

**SubResponse**: A SubResponseElementGenericType that specifies the response given by the protocol server for the corresponding subrequest as part of the Request element. The SubResponseElementGenericType is defined in section 2.2.4.7. The SubResponse element is defined in section 2.2.3.10. The SubRequest element is defined in section 2.2.3.8.

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

**UrlIsEncoded**: A TRUEFALSE value that specifies whether Url is encoded. Set to true if yes; Else false.

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

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

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

- The Url attribute of the corresponding Request element does not exist or is an empty string.<11> (The Url attribute and the Request element are defined in section 2.2.3.2.)
- The RequestToken attribute of the corresponding Request element is an empty string<12>. (The RequestToken attribute and the Request element are defined in section 2.2.3.2.)
- An exception occurred during the processing of a subrequest that was not entirely handled by the subrequest processing logic.

**ErrorMessage**: A string that specifies a description of the error message and also specifies information about what was expected by the server. This attribute MUST be present when the ErrorCode attribute is present and does not equal "Success". 
SuggestedFileName: The suggested filename that the host returns if the ShouldReturnDisambiguatedFileName flag is set on the Request\textsuperscript{13}.

ResourceId: A string that specifies the invariant ResourceId for a file, which uniquely identifies the file whose response is being generated.\textsuperscript{14} A ResourceId MUST NOT change over the lifetime of a file, even if the URL of the file changes. The ResourceId attribute SHOULD be present when the UseResourceId attribute is set to true in the corresponding Request element, and SHOULD NOT be present otherwise. If present, the string value MUST NOT be an empty string.

IntervalOverride An unsigned integer that if different than 0 specifies the value that the client MAY use as its request interval.

2.2.3.6 ResponseCollection

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

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

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

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

WebUrlIsEncoded: A TRUEFALSE value that specifies whether WebUrl is encoded. Set to true if yes; Else false.

2.2.3.7 ResponseVersion

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

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

VersionType: A complex type that is specified in section 2.2.4.9.

ErrorCode: A GenericErrorCodeTypes that specifies an error code value indicating the type of error that occurred during the processing of the mapping RequestVersion element. GenericErrorCodeTypes is defined in section 2.2.5.6. The RequestVersion element is defined in section 2.2.3.4. This attribute MUST be present only if any one of the following is true:
- The `Version` attribute of the `RequestVersion` element of the request message has a value that is less than 2. The `Version` attribute is defined in section 2.2.4.9. The `RequestVersion` element is defined in section 2.2.3.7.

- The protocol server identified by the `WebUrl` attribute of the `ResponseCollection` element does not exist or is not available. The `WebUrl` attribute and the `ResponseCollection` element are defined in section 2.2.3.6.

- The user does not have permission to issue a cell storage service request to the file identified by the `Url` attribute of the `Request` element. The `Url` attribute and the `Request` element are defined in section 2.2.3.2.

- This protocol is not enabled on the protocol server. <15>

**ErrorMessage**: A string that specifies a description of the error message and also specifies information about what was expected by the server.

### 2.2.3.8 SubRequest

The `SubRequest` element defines the subrequest for a specific URL for the file. `SubRequestElementGenericType` is defined in section 2.2.4.4.

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

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

### 2.2.3.9 SubRequestData

The `SubRequestData` element further qualifies the `SubRequest` element. `SubRequestData` specifies any data or input parameters that are used in processing the `SubRequest` element. `SubRequestDataGenericType` is defined in section 2.2.4.3.

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

The conditions under which a `SubRequestData` element MUST be sent as part of the `SubRequest` element are specified in section 2.2.4.4. The conditions under which a `SubRequestData` element MUST NOT be sent as part of a `SubRequest` element are specified in section 2.2.4.4.

### 2.2.3.10 SubResponse

Within a `Response` element for each `SubRequest` element that is in a `Request` element of a cell storage service request message, there MUST be a corresponding `SubResponse` element. The `SubResponse` element specifies the success or failure of processing the corresponding `SubRequest` element. In the case of success, the `SubResponse` element specifies the information requested as part of the `SubRequest` element. In the case of failure, the error code attribute in a `SubResponse` element specifies the error code result returned during processing of the `SubRequest` element. The error code attribute is defined in section 2.2.4.8.

`SubResponseElementGenericType` is defined in section 2.2.4.7. Each `SubResponse` element MUST have zero or one `SubResponseData` elements. The `SubResponseData` element is defined in section 2.2.3.11.
2.2.3.11 **SubResponseData**

A **SubResponseData** element is in a **SubResponse** element of a cell storage service response. The **SubResponseData** element specifies the responses for specific requests in the corresponding **SubRequestData** element. **SubResponseDataGenericType** is defined in section 2.2.4.6.

The conditions under which a **SubResponseData** element MUST be sent as part of the **SubResponse** element are specified in section 2.2.4.7.

### 2.2.4 Complex Types

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

<table>
<thead>
<tr>
<th>Complex type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenericPropertiesType</td>
<td>A generic type that contains information about the generic properties for the synchronization request.</td>
</tr>
<tr>
<td>PropertyType</td>
<td>A generic type that contains the property information for the synchronization request.</td>
</tr>
<tr>
<td>SubRequestDataGenericType</td>
<td>A generic type definition of data that is part of a cell storage service subrequest.</td>
</tr>
<tr>
<td>SubRequestElementGenericType</td>
<td>A generic type definition of a cell storage service subrequest.</td>
</tr>
<tr>
<td>SubRequestType</td>
<td>The base type definition of a cell storage service subrequest. <strong>SubRequestType</strong> is used to extend <strong>SubRequestElementGenericType</strong>.</td>
</tr>
<tr>
<td>SubResponseDataGenericType</td>
<td>A generic type definition of data that is part of a cell storage service subresponse.</td>
</tr>
<tr>
<td>SubResponseElementGenericType</td>
<td>A generic type definition of a cell storage service subresponse.</td>
</tr>
<tr>
<td>SubResponseType</td>
<td>The base type definition of a cell storage service subresponse. <strong>SubResponseType</strong> is used to extend <strong>SubResponseElementGenericType</strong>.</td>
</tr>
<tr>
<td>VersionType</td>
<td>The version of the message.</td>
</tr>
</tbody>
</table>

#### 2.2.4.1 **GenericPropertiesType**

The **GenericPropertiesType** complex type contains information about the generic properties for the synchronization request.
Property: A PropertyType that specifies the property information. PropertyType is defined in section 2.2.4.2.

2.2.4.2 PropertyType
The PropertyType complex type contains the property information for the synchronization request.

PropertyType
The PropertyType complex type contains the property information for the synchronization request.

<xs:complexType name="PropertyType">
  <xs:attribute name="Id" type="xs:string" use="required"/>
  <xs:attribute name="Value" type="xs:string" use="required"/>
</xs:complexType>

Id: The property id.
Value: The property value.

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

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

<xs:complexType name="SubRequestDataGenericType" mixed="true">
  <xs:all>
    <xs:element ref="i:Include" minOccurs="0" maxOccurs="1" />
    <xs:element name="PropertyIds" minOccurs="0" maxOccurs="1" type="tns:PropertyIdsType"/>
    <xs:attributeGroup ref="tns:SubRequestDataOptionalAttributes" />
  </xs:all>
</xs:complexType>

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

PropertyIds: An element of type PropertyIdsType (section 2.3.1.56) that specifies the set of properties. The PropertyIds element MUST only be included in the request if the Properties attribute value is set to "PropertyGet".

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

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

<table>
<thead>
<tr>
<th>Complex type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellSubRequestDataType</td>
<td>The type definition for cell subrequest data.</td>
</tr>
<tr>
<td>CoauthSubRequestDataType</td>
<td>The type definition for coauthoring subrequest data.</td>
</tr>
<tr>
<td>ExclusiveLockSubRequestDataType</td>
<td>The type definition for exclusive lock subrequest data.</td>
</tr>
</tbody>
</table>
The referenced Include element MUST be sent as part of the SubRequestData element in a cell storage service request message only if the Type attribute specified in the corresponding SubRequest element is set to a value of "Cell" and this cell subrequest is for the upload of the file's binary contents or metadata contents. The Type attribute is specified in section 2.2.4.4.

**2.2.4.4 SubRequestElementGenericType**

The SubRequestElementGenericType complex type contains information about a subrequest in a cell storage service request message. SubRequestElementGenericType provides a generic subrequest type definition. The SubRequestType definition from which SubRequestElementGenericType is extended is defined in section 2.2.4.5.

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

**SubRequestData:** A SubRequestDataGenericType that specifies the data or input parameters needed in processing the subrequest specified as part of the SubRequest element in a cell storage service request message. The SubRequestDataGenericType is defined in section 2.2.4.8. The SubRequest element is defined in section 2.2.3.8.

**Type:** A SubRequestAttributeType that specifies the type of subrequest to be processed. The subrequest is specified as part of the SubRequest element in a cell storage service request message. SubRequestAttributeType is defined in section 2.2.5.11. Depending on the type of subrequest, SubRequestElementGenericType MUST take one of the forms described in the following table.
<table>
<thead>
<tr>
<th>Complex type</th>
<th>Description</th>
</tr>
</thead>
</table>
| CoauthSubRequestType                 | The type definition for a coauthoring subrequest when the Type attribute is set to "Coauth".
| ExclusiveLockSubRequestType          | The type definition for an exclusive lock subrequest when the Type attribute is set to "ExclusiveLock".
| SchemaLockSubRequestType             | The type definition for a schema lock subrequest when the Type attribute is set to "SchemaLock".
| ServerTimeSubRequestType             | The type definition for a server time subrequest when the Type attribute is set to "ServerTime".
| WhoAmISubRequestType                 | The type definition for a Who Am I subrequest when the Type attribute is set to "WhoAmI".
| EditorsTableSubRequestType           | The type definition for an Editors Table subrequest.                         |
| GetDocMetaInfoSubRequestType         | The type definition for a Get Doc Meta Info subrequest when the Type attribute is set to "GetDocMetaInfo". |
| GetVersionsSubRequestType            | The type definition for a Get Versions subrequest when the Type attribute is set to "GetVersions". |
| FileOperationSubRequestType          | The type definition for a file operation subrequest when the Type attribute is set to "FileOperation". |
| VersioningSubRequestType             | The type definition for a versioning subrequest when the Type attribute is set to "Versioning". |
| AmIAI aloneSubRequestType            | The type definition for an Am I Alone subrequest when the Type attribute is set to "AmIAI alone". |
| LockStatusSubRequestType             | The type definition for a lock status subrequest when the Type attribute is set to "LockStatus". |
| PropertiesSubRequestType             | The type definition for a properties subrequest when the Type attribute is set to "Properties". |

CellSubRequestType is specified in section 2.3.1.2. CoauthSubRequestType is specified in section 2.3.1.6. ExclusiveLockSubRequestType is specified in section 2.3.1.10. SchemaLockSubRequestType is specified in section 2.3.1.14. ServerTimeSubRequestType is specified in section 2.3.1.17. WhoAmISubRequestType is specified in section 2.3.1.20. EditorsTableSubRequestType is specified in section 2.3.1.23. GetDocMetaInfoSubRequestType is specified in section 2.3.1.26. GetVersionsSubRequestType is specified in section 2.3.1.29. FileOperationSubRequestType is specified in section 2.3.1.32. VersioningSubRequestType is specified in section 2.3.1.34. AmIAI aloneSubRequestType is specified in section 2.3.1.37. LockStatusSubRequestType is specified in section 2.3.1.39. PropertiesSubRequestType is specified in section 2.3.1.53.

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

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

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

- The `Type` attribute that is specified in the `SubRequest` element is set to a value of "WhoAmI".
- The `Type` attribute that is specified in the `SubRequest` element is set to a value of "ServerTime".
- The `Type` attribute that is specified in the `SubRequest` element is set to a value of "GetDocMetaInfo".
- The `Type` attribute that is specified in the `SubRequest` element is set to a value of "GetVersions".
- The `Type` attribute that is specified in the `SubRequest` element is set to a value of "LockStatus".
- The `Type` attribute that is specified in the `SubRequest` element is set to a value of "Properties" and the `Properties` attribute value is set to "PropertyEnumerate".

### 2.2.4.5 SubRequestType

The `SubRequestType` complex type contains information about a basic cell storage service `subrequest`. The `SubRequestType` is used as the base complex type to extend the `SubRequestElementGenericType`. The `SubRequestElementGenericType` takes one of the following forms: `CellSubRequestType`, `CoauthSubRequestType`, `SchemaLockSubRequestType`, `ExclusiveLockSubRequestType`, `WhoAmISubRequestType`, `ServerTimeSubRequestType`, `GetDocMetaInfoSubRequestType`, or `FileOperationSubRequestType`. `CellSubRequestType` is specified in section 2.3.1.2, `CoauthSubRequestType` is specified in section 2.3.1.6, `ExclusiveLockSubRequestType` is specified in section 2.3.1.10, `SchemaLockSubRequestType` is specified in section 2.3.1.14, `ServerTimeSubRequestType` is specified in section 2.3.1.17, `WhoAmISubRequestType` is specified in section 2.3.1.20, `EditorsTableSubRequestType` is specified in section 2.3.1.24, `GetDocMetaInfoSubRequestType` is specified in section 2.3.1.26, `FileOperationSubRequestType` is specified in section 2.3.1.34, `VersioningSubRequestType` is specified in section 2.3.1.37, `AmIAloneSubRequestType` is specified in section 2.3.1.46, `LockStatusSubRequestType` is specified in section 2.3.1.49, `PropertiesSubRequestType` is specified in section 2.3.1.53.

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

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

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

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

2.2.4.6 SubResponseDataGenericType

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

```xml
<xs:complexType name="SubResponseDataGenericType" mixed="true">
  <xs:all>
    <xs:element ref="i:Include" minOccurs="0" maxOccurs="1" type="tns:GetDocMetaInfoPropertySetType"/>
    <xs:element name="DocProps" minOccurs="0" maxOccurs="1" type="tns:GetDocMetaInfoPropertySetType"/>
    <xs:element name="FolderProps" minOccurs="0" maxOccurs="1" type="tns:GetDocMetaInfoPropertySetType"/>
    <xs:element name="UserTable" minOccurs="0" maxOccurs="1" type="tns:VersioningUserTableType"/>
    <xs:element name="Versions" minOccurs="0" maxOccurs="1" type="tns:VersioningVersionListType"/>
    <xs:element name="PropertyIds" minOccurs="0" maxOccurs="1" type="tns:PropertyIdsType"/>
    <xs:element name="PropertyValues" minOccurs="0" maxOccurs="1" type="tns:PropertyValuesType"/>
  </xs:all>
  <xs:attributeGroup ref="tns:SubResponseDataOptionalAttributes"/>
</xs:complexType>
```

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

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

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

UserTable: An element of type VersioningUserTableType (section 2.3.1.40) that specifies data for the users represented in the version list. The UserTable element MUST be included in the response if the SubResponseType of the parent VersioningSubResponseType is of type "GetVersionList."

Versions: An element of type VersioningVersionListType (section 2.3.1.41) that specifies the list of versions of this file that exist on the server. The Versions element MUST be included in the response if the SubResponseType of the parent VersioningSubResponseType is of type "GetVersionList."

PropertyIds: An element of type PropertyIdsType (section 2.3.1.56) that specifies the set of properties.
**PropertyValues**: An element of type `PropertyValuesType` (section 2.3.58) that specifies the property values.

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

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

<table>
<thead>
<tr>
<th>Complex type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellSubResponseDataType</td>
<td>The type definition for cell subresponse data.</td>
</tr>
<tr>
<td>CoauthSubResponseDataType</td>
<td>The type definition for coauthoring subresponse data.</td>
</tr>
<tr>
<td>ExclusiveLockSubResponseDataType</td>
<td>The type definition for exclusive lock subresponse data.</td>
</tr>
<tr>
<td>SchemaLockSubResponseDataType</td>
<td>The type definition for schema lock subresponse data.</td>
</tr>
<tr>
<td>ServerTimeSubResponseDataType</td>
<td>The type definition for server time subresponse data.</td>
</tr>
<tr>
<td>WhoAmISubResponseDataType</td>
<td>The type definition for Who Am I subresponse data.</td>
</tr>
<tr>
<td>GetDocMetaInfoSubResponseDataType</td>
<td>The type definition for Get Doc Meta Info subresponse data.</td>
</tr>
<tr>
<td>VersioningSubResponseDataType</td>
<td>The type definition for versioning subresponse data.</td>
</tr>
<tr>
<td>AmIAloneSubResponseDataType</td>
<td>The type definition for Am I Alone subresponse data.</td>
</tr>
<tr>
<td>LockStatusSubResponseDataType</td>
<td>The type definition for Lock Status subresponse data.</td>
</tr>
<tr>
<td>PropertiesSubResponseDataType</td>
<td>The type definition for Properties subresponse data.</td>
</tr>
</tbody>
</table>

`CellSubResponseDataType` is specified in section 2.3.1.3. `CoauthSubResponseDataType` is specified in section 2.3.1.7. `ExclusiveLockSubResponseDataType` is specified in section 2.3.1.11. `SchemaLockSubResponseDataType` is specified in section 2.3.1.15. `ServerTimeSubResponseDataType` is specified in section 2.3.1.18. `WhoAmISubResponseDataType` is specified in section 2.3.1.21. `GetDocMetaInfoSubResponseDataType` is specified in section 2.3.1.27. `VersioningSubResponseDataType` is specified in section 2.3.1.38. `AmIAloneSubResponseDataType` is specified in section 2.3.1.47. `LockStatusSubResponseDataType` is specified in section 2.3.1.50. `PropertiesSubResponseDataType` is specified in section 2.3.1.54.

The referenced `Include` element MUST be sent as part of the `SubResponseData` element in a cell storage service response message only if the `Type` attribute specified in the corresponding `SubRequest` element is set to a value of "Cell" and this cell subrequest is for the download of the file’s binary contents or metadata contents. The `Type` attribute is specified in section 2.2.4.4.

### 2.2.4.7 SubResponseElementGenericType

The `SubResponseElementGenericType` complex type contains information about the success or failure in processing the cell storage service subrequest. In the case of success, it contains information requested as part of the subrequest. In the case of failure, the `ErrorCode` attribute that is part of a `SubResponse` element specifies the error code result for the subrequest. The `ErrorCode`
attribute is specified in section 2.2.4.8. **SubResponseElementGenericType** provides a generic subresponse type definition. The **SubResponseType** definition from which **SubResponseElementGenericType** is extended is defined in section 2.2.4.8.

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

**SubResponseData**: A **SubResponseDataGenericType** that specifies the response from the protocol server providing the data or file metadata information requested as part of the subrequest. **SubResponseDataGenericType** is defined in section 2.2.4.6.

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

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

<table>
<thead>
<tr>
<th>Complex type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellSubResponseType</td>
<td>The type definition for a cell subresponse.</td>
</tr>
<tr>
<td>CoauthSubResponseType</td>
<td>The type definition for a coauthoring subresponse.</td>
</tr>
<tr>
<td>ExclusiveLockSubResponseType</td>
<td>The type definition for an ExclusiveLock subresponse.</td>
</tr>
<tr>
<td>SchemaLockSubResponseType</td>
<td>The type definition for a SchemaLock subresponse.</td>
</tr>
<tr>
<td>ServerTimeSubResponseType</td>
<td>The type definition for a server time subresponse.</td>
</tr>
<tr>
<td>WhoAmISubResponseType</td>
<td>The type definition for a Who Am I subresponse.</td>
</tr>
<tr>
<td>EditorsTableSubResponseType</td>
<td>The type definition for an Editors Table subresponse.</td>
</tr>
<tr>
<td>GetDocMetaInfoSubResponseType</td>
<td>The type definition for a Get Doc Meta Info subresponse.</td>
</tr>
<tr>
<td>GetVersionsSubResponseType</td>
<td>The type definition for a Get Versions subresponse.</td>
</tr>
<tr>
<td>FileOperationSubResponseType</td>
<td>The type definition for a File Operation subresponse.</td>
</tr>
<tr>
<td>VersioningSubResponseType</td>
<td>The type definition for a Versioning subresponse.</td>
</tr>
<tr>
<td>AmIAloneSubResponseType</td>
<td>The type definition for an Am I Alone subresponse.</td>
</tr>
<tr>
<td>LockStatusSubResponseType</td>
<td>The type definition for a Lock Status subresponse.</td>
</tr>
<tr>
<td>PropertiesSubResponseType</td>
<td>The type definition for a Properties subresponse.</td>
</tr>
</tbody>
</table>
CellSubResponseType is specified in section 2.3.1.4. CoauthSubResponseType is specified in section 2.3.1.8. ExclusiveLockSubResponseType is specified in section 2.3.1.12. SchemaLockSubResponseType is specified in section 2.3.1.16. ServerTimeSubResponseType is specified in section 2.3.1.19. WhoAmISubResponseType is specified in section 2.3.1.22. EditorsTableSubResponseType is specified in section 2.3.1.25. GetDocMetaInfoSubResponseType is specified in section 2.3.1.30. GetVersionsSubResponseType is specified in section 2.3.1.32. FileOperationSubResponseType is specified in section 2.3.1.35. VersioningSubResponseType is specified in section 2.3.1.39. AmIAIaloneSubResponseType is specified in section 2.3.1.48. LockStatusSubResponseType is specified in section 2.3.1.51. PropertiesSubResponseType is specified in section 2.3.1.55.

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

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

The Type attribute is specified in section 2.2.4.4. The protocol server sets the value of the ErrorCode attribute to "Success" only if the protocol server succeeds in processing the cell storage service subrequest. The ErrorCode attribute is specified in section 2.2.4.8.

### 2.2.4.8 SubResponseType

The SubResponseType complex type contains information about a basic cell storage service subresponse. The SubResponseType is used as the base complex type to extend SubResponseElementGenericType. The SubResponseElementGenericType takes one of the following forms: CellSubResponseType, CoauthSubResponseType, SchemaLockSubResponseType, ExclusiveLockSubResponseType, ServerTimeSubResponseType, WhoAmISubResponseType, EditorsTableSubResponseType, GetDocMetaInfoSubResponseType, GetVersionsSubResponseType, VersioningSubResponseType, FileOperationSubResponseType, AmIAIaloneSubResponseType, LockStatusSubResponseType, or PropertiesSubResponseType. CellSubResponseType is specified in section 2.3.1.4. CoauthSubResponseType is specified in section 2.3.1.8. ExclusiveLockSubResponseType is specified in section 2.3.1.12. SchemaLockSubResponseType is specified in section 2.3.1.16. ServerTimeSubResponseType is specified in section 2.3.1.19. WhoAmISubResponseType is specified in section 2.3.1.22. EditorsTableSubResponseType is specified in section 2.3.1.25. GetDocMetaInfoSubResponseType is specified in section 2.3.1.30. GetVersionsSubResponseType is specified in section 2.3.1.32. FileOperationSubResponseType is specified in section 2.3.1.35. VersioningSubResponseType is specified in section 2.3.1.39. AmIAIaloneSubResponseType is specified in section 2.3.1.48. LockStatusSubResponseType is specified in section 2.3.1.51. PropertiesSubResponseType is specified in section 2.3.1.55.
is specified in section 2.3.1.16. **ServerTimeSubResponseType** is specified in section 2.3.1.19. **WhoAmISubResponseType** is specified in section 2.3.1.22. **EditorsTableSubResponseType** is specified in section 2.3.1.25. **GetDocMetaInfoSubResponseType** is specified in section 2.3.1.30. **GetVersionsSubResponseType** is specified in section 2.3.1.32. **FileOperationSubResponseType** is specified in section 2.3.1.35. **VersioningSubResponseType** is specified in section 2.3.1.39. **AmIAloneSubResponseType** is specified in section 2.3.1.48. **LockStatusSubResponseType** is specified in section 2.3.1.51. **PropertiesSubResponseType** is specified in section 2.3.1.55.

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

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

**ServerCorrelationId**: A guid that specifies a unique identifier that equals to the CorrelationId as specified in section 2.2.3.3.<16>

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

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

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

### 2.2.4.9 VersionType

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

```xml
<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 2.2.5.13.

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

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

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

<table>
<thead>
<tr>
<th>Simple type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>VersionNumberType</strong></td>
<td>The type of the <code>Version</code> attribute of the <code>RequestVersion</code> element and <code>ResponseVersion</code> element. The <code>RequestVersion</code> element and <code>ResponseVersion</code> element are used in cell storage service request and cell storage service response messages, respectively.</td>
</tr>
<tr>
<td><strong>NewEditorsTableCategoryErrorCodeTypes</strong></td>
<td>A subset of error codes returned as part of a cell storage service response message. <code>NewEditorsTableCategoryErrorCodeTypes</code> is an enumeration of error codes specific to an editors table subrequest.</td>
</tr>
<tr>
<td><strong>FileVersionNumberType</strong></td>
<td>The type that specifies a unique version of the file used in a versioning subrequest.</td>
</tr>
<tr>
<td><strong>VersioningRelatedErrorCodeTypes</strong></td>
<td>A subset of error codes returned as part of a cell storage service response message. <code>VersioningRelatedErrorCodeTypes</code> is an enumeration of error codes specific to a versioning subrequest.</td>
</tr>
</tbody>
</table>

### 2.2.5.1 CoauthStatusType

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

```xml
<xs:simpleType name="CoauthStatusType">
  <xs:restriction base="xs:string">
    <xs:enumeration value="None"/>
    <xs:enumeration value="Alone"/>
    <xs:enumeration value="Coauthoring"/>
  </xs:restriction>
</xs:simpleType>
```

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;None&quot;</td>
<td>None.</td>
</tr>
<tr>
<td>&quot;Alone&quot;</td>
<td>A string value of &quot;Alone&quot;, indicating a coauthoring status of alone. The alone status specifies that there is only one user in the coauthoring session who is editing the file.</td>
</tr>
<tr>
<td>&quot;Coauthoring&quot;</td>
<td>A string value of &quot;Coauthoring&quot;, indicating a coauthoring status of coauthoring. The coauthoring status specifies that the targeted URL for the file has more than one user in the coauthoring session and</td>
</tr>
</tbody>
</table>
### 2.2.5.2 DependencyCheckRelatedErrorCodeTypes

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

```
<x: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>
</x:simpleType>
```

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;DependentRequestNotExecuted&quot;</td>
<td>Indicates an error when the <strong>subrequest</strong> on which this specific subrequest is dependent has not been executed and the <strong>DependencyType</strong> attribute in this subrequest is set to &quot;OnExecute&quot;.</td>
</tr>
<tr>
<td>&quot;DependentOnlyOnSuccessRequestFailed&quot;</td>
<td>Indicates an error when the subrequest on which this specific subrequest is dependent has failed and the <strong>DependencyType</strong> attribute in this subrequest is set to &quot;OnSuccess&quot; or &quot;OnSuccessOrNotSupported&quot;.</td>
</tr>
<tr>
<td>&quot;DependentOnlyOnFailRequestSucceeded&quot;</td>
<td>Indicates an error when the subrequest on which this specific subrequest is dependent has succeeded and the <strong>DependencyType</strong> attribute in this subrequest is set to &quot;OnFail&quot;.</td>
</tr>
<tr>
<td>&quot;DependentOnlyOnNotSupportedRequestGetSupported&quot;</td>
<td>Indicates an error when the subrequest on which this specific subrequest is dependent is supported and the <strong>DependencyType</strong> attribute in this subrequest is set to &quot;OnNotSupported&quot; or &quot;OnSuccessOrOnNotSupported&quot;.</td>
</tr>
<tr>
<td>&quot;InvalidRequestDependencyType&quot;</td>
<td>Indicates an error when a subrequest dependency type that is not valid is specified.</td>
</tr>
</tbody>
</table>

**DependencyTypes** is defined in section [2.2.5.3](#dependencytypes).

### 2.2.5.3 DependencyTypes

The **DependencyTypes** simple type is used to represent the type of dependency that a cell storage service **subrequest** has on another cell storage service subrequest. The other cell storage service subrequest is identified by the **DependsOn** attribute of the **SubRequest** element. **DependsOn** is defined in section [2.2.4.5](#dependson). Depending on the dependency type, a cell storage service subrequest is either processed or not.
The value of **DependencyTypes** MUST be one of the values in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;OnExecute&quot;</td>
<td>Indicates that the subrequest MUST be processed only on the execution of the other subrequest. If RequestB has an OnExecute dependency on RequestA, the dependency succeeds if RequestA succeeds OR (RequestA failed NOT because an OnSuccess dependency failed AND RequestA failed NOT because an OnFailure dependency failed AND RequestA failed NOT because an OnExecute dependency failed)</td>
</tr>
<tr>
<td>&quot;OnSuccess&quot;</td>
<td>Indicates that the subrequest MUST be processed only on the successful execution of the other subrequest.</td>
</tr>
<tr>
<td>&quot;OnFail&quot;</td>
<td>Indicates that the subrequest MUST be processed only on the failed execution of the other subrequest.</td>
</tr>
<tr>
<td>&quot;OnNotSupported&quot;</td>
<td>Indicates that the subrequest MUST be processed only if the other subrequest is not supported.</td>
</tr>
<tr>
<td>&quot;OnSuccessOrNotSupported&quot;</td>
<td>Indicates that the subrequest MUST be processed only when one of the following conditions is true:</td>
</tr>
<tr>
<td></td>
<td>- On the successful execution of the other subrequest.</td>
</tr>
<tr>
<td></td>
<td>- If the other subrequest is not supported.</td>
</tr>
</tbody>
</table>

### 2.2.5.4 ErrorCodeTypes

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

```xml
<xs:simpleType name="ErrorCodeTypes">
</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. NewEditorsTableCategoryErrorCodeTypes is defined in section 2.2.5.14. VersioningRelatedErrorCodeTypes is defined in section 2.2.5.16.

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.

```xml
<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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;CoauthoringDisabled&quot;</td>
<td>The string value &quot;CoauthoringDisabled&quot;, indicating that an exclusive lock is granted on a file because coauthoring is disabled.</td>
</tr>
<tr>
<td>&quot;CheckedOutByCurrentUser&quot;</td>
<td>The string value &quot;CheckedOutByCurrentUser&quot;, 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.</td>
</tr>
<tr>
<td>&quot;CurrentUserHasExclusiveLock&quot;</td>
<td>The string value &quot;CurrentUserHasExclusiveLock&quot;, 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&lt;17&gt;.</td>
</tr>
</tbody>
</table>

2.2.5.6 GenericErrorCodeTypes

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

```xml
<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="PathNotFound"/>
      <xs:enumeration value="ResourceIdDoesNotExist"/>
      <xs:enumeration value="ResourceIdDoesNotMatch"/>
      <xs:enumeration value="InvalidSubRequest"/>
      <xs:enumeration value="SubRequestFail"/>
      <xs:enumeration value="BlockedFileType"/>
   </xs:restriction>
</xs:simpleType>
```
The value of **GenericErrorCodeTypes** MUST be one of the values in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Success&quot;</td>
<td>Indicates that the cell storage service subrequest succeeded for the given URL for the file.</td>
</tr>
<tr>
<td>&quot;IncompatibleVersion&quot;</td>
<td>Indicates an error when any an incompatible version number is specified as part of the RequestVersion element of the cell storage service.</td>
</tr>
<tr>
<td>&quot;InvalidUrl&quot;</td>
<td>Indicates that the associated protocol server site URL is empty.</td>
</tr>
<tr>
<td>&quot;FileDoesntExistOrCannotBeCreated&quot;</td>
<td>Indicates an error when either the targeted URL for the file specified as part of the Request element does not exist or file creation failed on the protocol server.</td>
</tr>
<tr>
<td>&quot;FileUnauthorizedAccess&quot;</td>
<td>Indicates an error when the targeted URL for the file specified as part of the Request element does not have correct authorization.</td>
</tr>
<tr>
<td>&quot;PathNotFound&quot;</td>
<td>Indicates an error when the file path is not found.</td>
</tr>
<tr>
<td>&quot;ResourceIdDoesNotExist&quot;</td>
<td>Indicates an error when neither of the ResourceID and Url attributes identify valid files.</td>
</tr>
<tr>
<td>&quot;ResourceIdDoesNotMatch&quot;</td>
<td>Indicates an error when the ResourceID does not identify a valid file but Url attribute identify a valid file.</td>
</tr>
<tr>
<td>&quot;InvalidSubRequest&quot;</td>
<td>Indicates an error when one or more SubRequest elements for a targeted URL for the file were unable to be parsed.</td>
</tr>
<tr>
<td>&quot;SubRequestFail&quot;</td>
<td>Indicates an unknown error when processing any SubRequest element for a targeted URL for the file.</td>
</tr>
<tr>
<td>&quot;BlockedFileType&quot;</td>
<td>Indicates an error when the targeted URL to the file's file type is blocked on the protocol server.</td>
</tr>
<tr>
<td>&quot;DocumentCheckoutRequired&quot;</td>
<td>Indicates an error when the targeted URL for the file is not yet checked out by the current client before sending a lock request on the file. The client sends a CheckoutFile web service request, as specified in [MS-LISTSWS], before the protocol client gets a lock on the file. If the document is not checked out by the current client, the protocol server MUST return an error code value set to &quot;DocumentCheckoutRequired&quot; in the cell storage service response message.</td>
</tr>
<tr>
<td>&quot;InvalidArgument&quot;</td>
<td>Indicates an error when any of the cell storage service subrequests for the targeted URL for the file contains input parameters that are not valid.</td>
</tr>
<tr>
<td>Value</td>
<td>Meaning</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&quot;RequestNotSupported&quot;</td>
<td>Indicates an error when the targeted cell storage service subrequest is a valid subrequest, but the server does not support that subrequest.</td>
</tr>
<tr>
<td>&quot;InvalidWebUrl&quot;</td>
<td>Indicates an error when the associated protocol server site URL is not found.</td>
</tr>
<tr>
<td>&quot;WebServiceTurnedOff&quot;</td>
<td>Indicates an error when the web service is turned off during the processing of the cell storage service request.</td>
</tr>
<tr>
<td>&quot;ColdStoreConcurrencyViolation&quot;</td>
<td>Indicates an error when the file that is correctly stored on the server is modified by another user before the current user finished writing to the underlying store. The &quot;ColdStoreConcurrencyViolation&quot; error code value MUST only be sent as part of processing one of the following types of cell storage service request messages:</td>
</tr>
<tr>
<td></td>
<td>- Cell</td>
</tr>
<tr>
<td></td>
<td>- Coauth</td>
</tr>
<tr>
<td></td>
<td>- SchemaLock</td>
</tr>
<tr>
<td></td>
<td>- ExclusiveLock</td>
</tr>
<tr>
<td></td>
<td>The protocol client can retry the request if this error occurs.</td>
</tr>
<tr>
<td>&quot;HighLevelExceptionThrown&quot;</td>
<td>Indicates any undefined error that occurs during the processing of the cell storage service request. This could be caused by a temporarily unavailable resource such as a network or machine.</td>
</tr>
<tr>
<td>&quot;Unknown&quot;</td>
<td>Indicates any undefined error that occurs during the processing of the cell storage service request.</td>
</tr>
</tbody>
</table>

### 2.2.5.7 GUID

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

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

### 2.2.5.8 LockAndCoauthRelatedErrorCodeTypes

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

```xml
<xs:simpleType name="LockAndCoauthRelatedErrorCodeTypes">
  <xs:restriction base="xs:string">
  </xs:restriction>
</xs:simpleType>
```
The value of **LockAndCoauthRelatedErrorCodeTypes** MUST be one of the values in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;LockRequestFail&quot;</td>
<td>Indicates an undefined error that occurs during the processing of lock operations requested as part of a cell storage service subrequest.</td>
</tr>
<tr>
<td>&quot;FileAlreadyLockedOnServer&quot;</td>
<td>Indicates an error when there is an already existing exclusive lock on the targeted URL for the file or a schema lock on the file with a different schema lock identifier. When the &quot;FileAlreadyLockedOnServer&quot; error code is returned as the error code value in the SubResponse element, the protocol server returns the identity of the users who are currently holding the lock on the file in the ErrorMessage attribute. The ErrorMessage and ErrorCode attributes are defined in section 2.2.4.8.</td>
</tr>
<tr>
<td>&quot;FileNotLockedOnServer&quot;</td>
<td>Indicates an error when no exclusive lock exists on a file and a release of the lock or a conversion of the lock is requested as part of a cell storage service request.</td>
</tr>
<tr>
<td>&quot;FileNotLockedOnServerAsCoauthDisabled&quot;</td>
<td>Indicates an error when no shared lock exists on a file because coauthoring of the file is disabled on the server.</td>
</tr>
<tr>
<td>&quot;LockNotConvertedAsCoauthDisabled&quot;</td>
<td>Indicates an error when a protocol server fails to process a lock conversion request sent as part of a cell storage service request because coauthoring of the file is disabled on the server.</td>
</tr>
<tr>
<td>&quot;FileAlreadyCheckedOutOnServer&quot;</td>
<td>Indicates an error when the file is checked out by another client, which is preventing the file from being locked by the current client. When the &quot;FileAlreadyCheckedOutOnServer&quot; error code is returned as the error code value in the SubResponse element, the protocol server returns the identity of the user who has currently checked out the file in the error message attribute. The ErrorMessage and ErrorCode attributes are defined in section 2.2.4.8.</td>
</tr>
<tr>
<td>&quot;ConvertToSchemaFailedFileCheckedOutByCurrentUser&quot;</td>
<td>Indicates an error when converting to a shared lock fails because the file is checked out by the current client.</td>
</tr>
<tr>
<td>Value</td>
<td>Meaning</td>
</tr>
<tr>
<td>-----------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>&quot;CoauthRefBlobConcurrencyViolation&quot;</td>
<td>Indicates an error when a save of the File coauthoring tracker, which is maintained by the protocol server, fails after some other client edited the File coauthoring tracker before the save is done by the current client. The File coauthoring tracker is defined in section 3.1.1.</td>
</tr>
<tr>
<td>&quot;MultipleClientsInCoauthSession&quot;</td>
<td>Indicates an error when all of the following conditions are true:</td>
</tr>
<tr>
<td></td>
<td>￭ A coauthoring subrequest of type &quot;Convert to exclusive lock&quot; or schema lock subrequest of type &quot;Convert to exclusive lock&quot; is requested on a file.</td>
</tr>
<tr>
<td></td>
<td>￭ There is more than one client in the current coauthoring session for that file.</td>
</tr>
<tr>
<td></td>
<td>￭ The ReleaseLockOnConversionToExclusiveFailure attribute specified as part of the subrequest is set to false.</td>
</tr>
<tr>
<td>&quot;InvalidCoauthSession&quot;</td>
<td>Indicates an error when one of the following conditions is true when a coauthoring subrequest or schema lock subrequest is sent:</td>
</tr>
<tr>
<td></td>
<td>￭ No coauthoring session exists for the file.</td>
</tr>
<tr>
<td></td>
<td>￭ The current client does not exist in the coauthoring session for the file.</td>
</tr>
<tr>
<td></td>
<td>￭ The current client exists in the coauthoring session, but protocol server is unable to remove it from the coauthoring session for the file.</td>
</tr>
<tr>
<td></td>
<td>A coauthoring session indicates a shared lock on the coauthorable file that is shared by one or more clients.</td>
</tr>
<tr>
<td>&quot;NumberOfCoauthorsReachedMax&quot;</td>
<td>Indicates an error when the number of users that coauthor a file has reached the threshold limit. The threshold limit specifies the maximum number of users allowed to coauthor a file at any instant in time. The threshold limit MUST be set to an integer value with a minimum allowed value of 2 and maximum allowed value of 99.</td>
</tr>
<tr>
<td>ExitCoauthSessionAsConvertToExclusiveFailed</td>
<td>Indicates an error when a coauthoring subrequest or schema lock subrequest of type &quot;Convert to exclusive lock&quot; is sent by the client with the ReleaseLockOnConversionToExclusiveFailure attribute set to true, and there is more than one client editing the file.</td>
</tr>
</tbody>
</table>

### 2.2.5.9 LockTypes

The LockTypes simple type is used to represent the type of file lock. LockTypes specifies the type of lock requested or granted in a cell storage service request or cell storage service response message.
The value of **LockTypes** MUST be one of the values in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;None&quot;, &quot;0&quot;</td>
<td>The string value &quot;None&quot; or integer value &quot;0&quot;, indicating no type of file lock on the file. This value is only for server internal use, it will not appear in the response.</td>
</tr>
<tr>
<td>&quot;SchemaLock&quot;, &quot;1&quot;</td>
<td>The string value &quot;SchemaLock&quot; or integer value &quot;1&quot;, indicating a shared lock on the file. In a cell storage service request message, a shared lock indicates a request for sharing the lock on the file, which allows for coauthoring the file. In a cell storage service response message, a shared lock indicates that the current client is granted a shared lock on the file, which allows for coauthoring the file along with other clients.</td>
</tr>
<tr>
<td>&quot;ExclusiveLock&quot;, &quot;2&quot;</td>
<td>The string value &quot;ExclusiveLock&quot; or integer value &quot;2&quot;, indicating an exclusive lock on the file. In a cell storage service request message, an exclusive lock indicates a request for exclusive access to the file. In a cell storage service response message, an exclusive lock indicates that an exclusive lock is granted to the current client for that specific file. In a cell storage service response message, an exclusive lock also indicates that all other clients requesting an exclusive lock on that file MUST be allowed to open this file only in read-only mode.</td>
</tr>
</tbody>
</table>

**2.2.5.10 MinorVersionNumberType**

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

```xml
<xs:simpleType name="MinorVersionNumberType">
  <xs:restriction base="xs:unsignedShort">
    <xs:minInclusive value="0"/>
    <xs:maxInclusive value="3"/>
  </xs:restriction>
</xs:simpleType>
```
The value of **MinorVersionNumberType** MUST be the value that is listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| 0     | In requests, indicates that the protocol client will manage the editors table through **PutChanges** requests as specified in [MS-FSSHTTP].  
In responses, indicates that the protocol server is not capable of managing the editors table and expects the protocol client to do so through **PutChanges** requests. |
| 2     | In requests, indicates that the protocol client is capable of managing the editors table.  
In responses, indicates that the protocol server is capable of managing the editors table. |
| 3     | In requests, indicates that the protocol client is capable of performing ResourceID specific behavior.  
In responses, indicates that the protocol server is capable of performing ResourceID specific behavior. |

### 2.2.5.11 SubRequestMethodType

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

- Cell
- Coauthoring
- Schema lock
- Exclusive lock
- Who Am I
- Server time
- Editors Table
- Get Doc Meta Info
- Get Versions
- File Operation
- Versioning
- Am I Alone
- Lock Status
- Properties
The value of `SubRequestAttributeType` MUST be one of the values in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Cell&quot;</td>
<td>The string value &quot;Cell&quot;, indicating the cell storage service subrequest is to be processed as a cell subrequest operation.</td>
</tr>
<tr>
<td>&quot;Coauth&quot;</td>
<td>The string value &quot;Coauth&quot;, indicating the cell storage service subrequest is to be processed as a coauthoring subrequest operation.</td>
</tr>
<tr>
<td>&quot;SchemaLock&quot;</td>
<td>The string value &quot;SchemaLock&quot;, indicating the cell storage service subrequest is to be processed as a schema lock subrequest operation.</td>
</tr>
<tr>
<td>&quot;WhoAmI&quot;</td>
<td>The string value &quot;WhoAmI&quot;, indicating the cell storage service subrequest is to be processed as a Who Am I subrequest operation.</td>
</tr>
<tr>
<td>&quot;ServerTime&quot;</td>
<td>The string value &quot;ServerTime&quot;, indicating the cell storage service subrequest is to be processed as a server time subrequest operation.</td>
</tr>
<tr>
<td>&quot;ExclusiveLock&quot;</td>
<td>The string value &quot;ExclusiveLock&quot;, indicating the cell storage service subrequest is to be processed as an exclusive lock subrequest operation.</td>
</tr>
<tr>
<td>&quot;EditorsTable&quot;</td>
<td>The string value &quot;EditorsTable&quot;, indicating the cell storage service subrequest is to be processed as an editors table subrequest operation.</td>
</tr>
<tr>
<td>&quot;GetDocMetaInfo&quot;</td>
<td>The string value &quot;GetDocMetaInfo&quot;, indicating the cell storage service subrequest is to be processed as a Get Doc Meta Info subrequest operation.</td>
</tr>
<tr>
<td>&quot;GetVersions&quot;</td>
<td>The string value &quot;GetVersions&quot;, indicating the cell storage service subrequest is to be processed as a Get Versions subrequest operation.</td>
</tr>
<tr>
<td>&quot;FileOperation&quot;</td>
<td>The string value &quot;FileOperation&quot;, indicating the cell storage service subrequest is to be processed as a File Operation subrequest operation.</td>
</tr>
<tr>
<td>&quot;Versioning&quot;</td>
<td>The string value &quot;Versioning&quot;, indicating the cell storage service subrequest is to be processed as a Versioning subrequest operation.</td>
</tr>
<tr>
<td>Value</td>
<td>Meaning</td>
</tr>
<tr>
<td>------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>storage service subrequest is to be processed as a versioning subrequest operation.</td>
<td></td>
</tr>
<tr>
<td>&quot;AmIAlone&quot;</td>
<td>The string value &quot;AmIAlone&quot;, indicating the cell storage service subrequest is to be processed as an Am I Alone subrequest operation.</td>
</tr>
<tr>
<td>&quot;LockStatus&quot;</td>
<td>The string value &quot;LockStatus&quot;, indicating the cell storage service subrequest is to be processed as a Lock Status subrequest operation.</td>
</tr>
<tr>
<td>&quot;Properties&quot;</td>
<td>The string value &quot;Properties&quot;, indicating the cell storage service subrequest is to be processed as a Properties subrequest operation.</td>
</tr>
</tbody>
</table>

### 2.2.5.12 TRUEFALSE

This type is used to specify a Boolean value.

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

### 2.2.5.13 VersionNumberType

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>A version number of 2.</td>
</tr>
</tbody>
</table>

### 2.2.5.14 NewEditorsTableCategoryErrorCodeTypes

The `NewEditorsTableCategoryErrorCodeTypes` simple type is used to represent error codes for the processing of an EditorsTable subrequest.

```xml
<xs:simpleType name="NewEditorsTableCategoryErrorCodeTypes">
  <xs:restriction base="xs:string">
    <xs:enumeration value="EditorMetadataQuotaReached"/>
    <xs:enumeration value="EditorMetadataStringExceedsLengthLimit"/>
    <xs:enumeration value="EditorClientIdNotFound"/>
  </xs:restriction>
</xs:simpleType>
```
The value of **NewEditorsTableCategoryErrorCodeTypes** MUST be one of the values in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;EditorMetadataQuotaReached&quot;</td>
<td>Indicates an error when the protocol client has already exceeded its quota for number of key/value pairs.</td>
</tr>
<tr>
<td>&quot;EditorMetadataStringExceedsLengthLimit&quot;</td>
<td>Indicates an error when the key and value exceeds the server’s length limit.</td>
</tr>
<tr>
<td>&quot;EditorClientIdNotFound&quot;</td>
<td>Indicates an error when the specify client does not currently exist in the editors table.</td>
</tr>
</tbody>
</table>

### 2.2.5.15 FileVersionNumberType

The **FileVersionNumberType** simple type is used to represent the unique version number on the server for a version of a file.

```xml
<xs:simpleType name="FileVersionNumberType">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]+[.][0-9]+" />
  </xs:restriction>
</xs:simpleType>
```

The value of a **FileVersionNumberType** is typically in the form `major.minor` (for example, 1.0).

### 2.2.5.16 VersioningRelatedErrorCodeTypes

The **VersioningRelatedErrorCodeTypes** simple type is used to represent error codes that occur during the processing of a versioning **subrequest**.

```xml
<xs:simpleType name="VersioningRelatedErrorCodeTypes">
  <xs:restriction base="xs:string">
    <xs:enumeration value="VersionNotFound"/>
  </xs:restriction>
</xs:simpleType>
```

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;VersionNotFound&quot;</td>
<td>Indicates that the version number specified by the protocol client doesn’t match a version of the file on the protocol server.</td>
</tr>
</tbody>
</table>
2.2.6 Attributes
This specification does not define any common XML schema attribute definitions.

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

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

<table>
<thead>
<tr>
<th>Attribute Groups</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SubRequestDataOptionalAttributes</td>
<td>Contains XML schema attributes used in all SubRequestData elements. It is a union of subrequest data attributes used for all types of subrequests.</td>
</tr>
<tr>
<td>SubResponseDataOptionalAttributes</td>
<td>Contains XML schema attributes used in all SubResponseData elements. It is a union of subresponse data attributes used for all types of subresponses.</td>
</tr>
</tbody>
</table>

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

```xml
<xs:attributeGroup name="SubRequestDataOptionalAttributes">
  <xs:attributeGroup ref="tns:CellSubRequestDataOptionalAttributes"/>
  <xs:attributeGroup ref="tns:CoauthSubRequestDataOptionalAttributes"/>
  <xs:attributeGroup ref="tns:SchemaLockSubRequestDataOptionalAttributes"/>
  <xs:attributeGroup ref="tns:ExclusiveLockSubRequestDataOptionalAttributes"/>
  <xs:attributeGroup ref="tns:EditorsTableSubRequestDataOptionalAttributes"/>
  <xs:attributeGroup ref="tns:FileOperationSubRequestDataOptionalAttributes"/>
  <xs:attributeGroup ref="tns:VersioningSubRequestDataOptionalAttributes"/>
  <xs:attributeGroup ref="tns:PropertiesSubRequestDataOptionalAttributes"/>
  <xs:attribute name="ClientID" type="xs:string" use="optional"/>
  <xs:attribute name="AllowFallbackToExclusive" type="xs:boolean" use="optional"/>
  <xs:attribute name="ReleaseLockOnConversionToExclusiveFailure" type="xs:boolean" use="optional"/>
  <xs:attribute name="SchemaLockID" type="xs:string" use="optional"/>
  <xs:attribute name="Timeout" type="xs:integer" use="optional"/>
  <xs:attribute name="ExclusiveLockID" type="xs:string" use="optional"/>
  <xs:attribute name="BinaryDataSize" type="xs:long" use="optional"/>
  <xs:attribute name="AsEditor" type="xs:boolean" use="optional"/>
  <xs:attribute name="Key" type="xs:string" use="optional"/>
  <xs:attribute name="Value" type="xs:binary" use="optional"/>
  <xs:attribute name="NewFileName" type="xs:string" use="optional"/>
  <xs:attribute name="Version" type="tns:FileVersionNumberType" use="optional"/>
  <xs:attribute name="TransitionID" type="tns:guid" use="optional"/>
</xs:attributeGroup>
```
CellSubRequestDataOptionalAttributes: An attribute group that specifies attributes that MUST be used only for SubRequestData elements whose parent SubRequest element’s Type attribute is set to "Cell". This attribute group is defined in section 2.3.3.1.

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

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

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

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

FileOperationSubRequestDataOptionalAttributes: An attribute group that specifies attributes that MUST be used only for SubRequestData elements whose parent SubRequest element’s Type attribute is set to "FileOperation". This attribute group is defined in section 2.3.3.8.

VersioningSubRequestDataOptionalAttributes: An attribute group that specifies attributes that MUST be used only for SubRequestData elements whose parent SubRequest element’s Type attribute is set to "Versioning". This attribute group is defined in section 2.3.3.9.

PropertiesSubRequestDataOptionalAttributes: An attribute group that specifies attributes that MUST be used only for SubRequestData elements whose parent SubRequest element’s Type attribute is set to "Properties". This attribute group is defined in section 2.3.3.10.

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

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

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

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

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

When all the preceding conditions are true:

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

**SchemaLockID**: A string that is globally unique and known among all protocol clients that share the same protocol version. This schema lock identifier is used by the protocol server to block other clients that have different schema lock identifiers. After a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST allow only other protocol clients that specify the same schema lock identifier to share the file lock. The protocol server ensures that at any instant of time, only clients having the same schema lock identifier can lock the file. After all the protocol clients have released their lock for that file, the protocol server MUST allow a protocol client with a different schema lock identifier to get a shared lock for that file. The string "29358EC1-E813-4793-8E70-ED034E7B73C" has been reserved for use for this attribute.<21>

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

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

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

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

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

- Join editing session
- Refresh editing session

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

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

- Update Editor Metadata
- Remove Editor Metadata

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

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

The types of editors table subrequests are defined in section 2.3.3.7.
NewFileName: This string MUST only contain a valid file name, with no relative path. The NewFileName attribute MUST be sent only when the file operation subrequest has a FileOperationSubRequestType attribute set to "Rename".

Version: A FileVersionNumberType that serves to uniquely identify a version of a file on the server. FileVersionNumberType is defined in section 2.2.5.15. Version MUST be specified when the versioning subrequest has a VersioningSubRequestType attribute set to "RestoreVersion". The types of versioning subrequest are defined in section 2.3.3.9.

TransitionID: A guid that specifies the unique file identifier for that file in the protocol server.

2.2.8.2 SubResponseDataOptionalAttributes

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

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

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

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

ServerTime: A positive integer that specifies the server time, which is expressed as a tick count. A single tick represents 100 nanoseconds, or one ten-millionth of a second. ServerTime specifies the number of 100-nanosecond intervals that have elapsed since 00:00:00 on January 1, 0001, which SHOULD be Coordinated Universal Time (UTC). The ServerTime attribute MUST be specified in a server time subresponse that is generated in response to a server time subrequest.

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

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

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

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

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

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

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

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

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

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

**AmIAlone:** A Boolean value that specifies whether the user is alone in the coauthoring session.

**LockedID:** A **guid** that specifies the id of the lock.

**LockedBy:** A string that specifies the user that has the file locked, if any.

### 2.2.9 Common Data Structures

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

### 2.3 Subsidiary Message Syntax

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

### 2.3.1 Complex Types

The following table summarizes the set of other **XML schema** complex type definitions defined by this specification.
<table>
<thead>
<tr>
<th>Complex type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellSubRequestDataType</td>
<td>The type definition for cell subrequest data.</td>
</tr>
<tr>
<td>CellSubRequestType</td>
<td>The type definition for a cell subrequest when the <strong>Type</strong> attribute is set to &quot;Cell&quot;.</td>
</tr>
<tr>
<td>CellSubResponseDataType</td>
<td>The type definition for cell subresponse data.</td>
</tr>
<tr>
<td>CellSubResponseType</td>
<td>The type definition for a cell subresponse.</td>
</tr>
<tr>
<td>CoauthSubRequestDataType</td>
<td>The type definition for coauthoring subrequest data.</td>
</tr>
<tr>
<td>CoauthSubRequestType</td>
<td>The type definition for a coauthoring subrequest when the <strong>Type</strong> attribute is set to &quot;Coauth&quot;.</td>
</tr>
<tr>
<td>CoauthSubResponseDataType</td>
<td>The type definition for coauthoring subresponse data.</td>
</tr>
<tr>
<td>CoauthSubResponseType</td>
<td>The type definition for a coauthoring subresponse.</td>
</tr>
<tr>
<td>ExclusiveLockSubRequestDataType</td>
<td>The type definition for exclusive lock subrequest data.</td>
</tr>
<tr>
<td>ExclusiveLockSubRequestType</td>
<td>The type definition for an exclusive lock subrequest when the <strong>Type</strong> attribute is set to &quot;ExclusiveLock&quot;.</td>
</tr>
<tr>
<td>ExclusiveLockSubResponseDataType</td>
<td>The type definition for exclusive lock subresponse data.</td>
</tr>
<tr>
<td>ExclusiveLockSubResponseType</td>
<td>The type definition for an ExclusiveLock subresponse.</td>
</tr>
<tr>
<td>SchemaLockSubRequestDataType</td>
<td>The type definition for schema lock subrequest data.</td>
</tr>
<tr>
<td>SchemaLockSubRequestType</td>
<td>The type definition for a schema lock subrequest when the <strong>Type</strong> attribute is set to &quot;SchemaLock&quot;.</td>
</tr>
<tr>
<td>SchemaLockSubResponseDataType</td>
<td>The type definition for schema lock subresponse data.</td>
</tr>
<tr>
<td>SchemaLockSubResponseType</td>
<td>The type definition for a schema lock subresponse.</td>
</tr>
<tr>
<td>ServerTimeSubRequestType</td>
<td>The type definition for a server time subrequest when the <strong>Type</strong> attribute is set to &quot;ServerTime&quot;.</td>
</tr>
<tr>
<td>ServerTimeSubResponseDataType</td>
<td>The type definition for server time subresponse data.</td>
</tr>
<tr>
<td>ServerTimeSubResponseType</td>
<td>The type definition for a server time subresponse.</td>
</tr>
<tr>
<td>WhoAmISubRequestType</td>
<td>The type definition for a Who Am I subrequest when the <strong>Type</strong> attribute is set to &quot;WhoAmI&quot;.</td>
</tr>
<tr>
<td>WhoAmISubResponseDataType</td>
<td>The type definition for Who Am I subresponse data.</td>
</tr>
<tr>
<td>WhoAmISubResponseType</td>
<td>The type definition for a Who Am I subresponse.</td>
</tr>
<tr>
<td>EditorsTableSubRequestDataType</td>
<td>The type definition for editors table subrequest data.</td>
</tr>
<tr>
<td>EditorsTableSubRequestType</td>
<td>The type definition for an editors table subrequest when the <strong>Type</strong> attribute is set to &quot;EditorsTable&quot;.</td>
</tr>
<tr>
<td>EditorsTableSubResponseDataType</td>
<td>The type definition for an editors table subresponse.</td>
</tr>
<tr>
<td>GetDocMetaInfoSubRequestType</td>
<td>The type definition for a Get Doc Meta Info subrequest when the <strong>Type</strong> attribute is set to &quot;GetDocMetaInfo&quot;</td>
</tr>
<tr>
<td>GetDocMetaInfoSubResponseDataType</td>
<td>The type definition for Get Doc Meta Info subresponse data.</td>
</tr>
<tr>
<td>Complex type</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>GetDocMetaInfoPropertySetType</td>
<td>Describes the set of metainfo related to the file.</td>
</tr>
<tr>
<td>GetDocMetaInfoPropertyType</td>
<td>Contains a metainfo key/value pair that is related either to the file against which the request is made or its parent directory as part of the corresponding GetDocMetaInfo subrequest.</td>
</tr>
<tr>
<td>GetDocMetaInfoSubResponseType</td>
<td>The type definition for a Get Doc Meta Info subresponse.</td>
</tr>
<tr>
<td>GetVersionsSubRequestType</td>
<td>The type definition for a Get Versions subrequest when the attribute is set to &quot;GetVersions&quot;</td>
</tr>
<tr>
<td>GetVersionsSubResponseType</td>
<td>The type definition for a Get Versions subresponse.</td>
</tr>
<tr>
<td>FileOperationSubRequestDataType</td>
<td>The type definition for File Operation subrequest data.</td>
</tr>
<tr>
<td>FileOperationSubRequestType</td>
<td>The type definition for a File Operation subrequest when the Type attribute is set to &quot;FileOperation&quot;.</td>
</tr>
<tr>
<td>FileOperationSubResponseType</td>
<td>The type definition for File Operation subresponse data.</td>
</tr>
<tr>
<td>VersioningSubRequestDataType</td>
<td>The type definition for Versioning subrequest data.</td>
</tr>
<tr>
<td>VersioningSubRequestType</td>
<td>The type definition for a Versioning subrequest when the Type attribute is set to &quot;Versioning&quot;.</td>
</tr>
<tr>
<td>VersioningSubResponseDataType</td>
<td>The type definition for Versioning subresponse data.</td>
</tr>
<tr>
<td>VersioningSubResponseType</td>
<td>The type definition for a Versioning subresponse.</td>
</tr>
<tr>
<td>VersioningUserTableType</td>
<td>Specifies the information about users that are represented in the versions of the file.</td>
</tr>
<tr>
<td>VersioningVersionListType</td>
<td>Specifies a list describing the versions of the file.</td>
</tr>
<tr>
<td>UserDataType</td>
<td>Describes the details of a user.</td>
</tr>
<tr>
<td>FileVersionDataType</td>
<td>Describes the details about a single version of the file.</td>
</tr>
<tr>
<td>FileVersionEventDataType</td>
<td>Describes the details about a file event that happened to a specific version of a file.</td>
</tr>
<tr>
<td>AmIAloneSubRequestDataType</td>
<td>The type definition for Am I Alone subrequest data.</td>
</tr>
<tr>
<td>AmIAloneSubRequestType</td>
<td>The type definition for an Am I Alone subrequest when the Type attribute is set to &quot;AmIAlone&quot;.</td>
</tr>
<tr>
<td>AmIAloneSubResponseDataType</td>
<td>The type definition for Am I Alone subresponse data.</td>
</tr>
<tr>
<td>AmIAloneSubResponseType</td>
<td>The type definition for an Am I Alone subresponse.</td>
</tr>
<tr>
<td>LockStatusSubRequestType</td>
<td>The type definition for Lock Status subrequest when the Type attribute is set to “LockStatus”</td>
</tr>
<tr>
<td>LockStatusSubResponseDataType</td>
<td>The type definition for Lock Status subresponse data.</td>
</tr>
<tr>
<td>PropertiesSubRequestDataType</td>
<td>The type definition for a Properties subrequest data.</td>
</tr>
<tr>
<td>PropertiesSubResponseType</td>
<td>The type definition for Properties subrequest when the</td>
</tr>
</tbody>
</table>
### Complex type

<table>
<thead>
<tr>
<th>Complex type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PropertiesSubResponseDataType</td>
<td>The type definition for Properties subresponse data.</td>
</tr>
<tr>
<td>PropertiesSubResponseType</td>
<td>The type definition for a Properties subresponse.</td>
</tr>
<tr>
<td>PropertyIdsType</td>
<td>The type definition for property Ids.</td>
</tr>
<tr>
<td>PropertyIdType</td>
<td>The type definition for a property Id.</td>
</tr>
<tr>
<td>PropertyValuesType</td>
<td>The type definition for property values.</td>
</tr>
<tr>
<td>PropertyValueType</td>
<td>The type definition for a property value.</td>
</tr>
</tbody>
</table>

#### 2.3.1.1 CellSubRequestDataType

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

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

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

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

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

- The Coalesce attribute is set to true, and the protocol client holds a shared lock on the file. The Coalesce attribute is defined in section 2.3.3.1.
- The cell subrequest is for uploading binary file contents in a partition.
The string "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use for this attribute.<24>

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

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

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

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

### 2.3.1.2 CellSubRequestType

The **CellSubRequestType** complex type contains information about a cell subrequest. The **SubRequestType** definition from which **CellSubRequestType** is extended is defined in section 2.2.4.5.

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

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

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

### 2.3.1.3 CellSubResponseDataType

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

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

[MS-FSSHTTP] - v20190924
File Synchronization via SOAP over HTTP Protocol
Copyright © 2019 Microsoft Corporation
Release: September 24, 2019
Include: A complex type, as specified in [XOP10] section 2.1, that is used for encapsulating and sending large amounts of binary data. The referenced Include element is specified in section 2.2.3.1. As part of processing the cell subrequest, the referenced Include element MUST be sent as part of the SubResponseData element in a cell storage service response message only if the cell subrequest is for the download of a file's binary or metadata contents and only when these contents are non-empty.

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

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

2.3.1.4 CellSubResponseType

The CellSubResponseType complex type contains information about the success or failure in processing the cell subrequest. In the case of success, it contains information requested as part of the cell subrequest. SubResponseType definition from which CellSubResponseType is extended is defined in section 2.2.4.8.

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

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

2.3.1.5 CoauthSubRequestDataType

The CoauthSubRequestDataType complex type contains information about data or input parameters used in processing a coauthoring subrequest. The SchemaLockID attribute and the CoauthRequestType attribute specified in the CoauthSubRequestDataOptionalAttributes attribute group MUST both be specified for a coauthoring subrequest. The SchemaLockID attribute and the CoauthRequestType attribute is specified as part of the SubRequestData element associated with a coauthoring SubRequest element. CoauthSubRequestDataOptionalAttributes is defined in section 2.3.3.3. If the CoauthRequestType attribute is not provided, a
"HighLevelExceptionThrown" error code SHOULD<25> be returned as part of the ResponseVersion element. If other attributes are not provided, an "InvalidArgument" error code MUST be returned as part of the SubResponseData element associated with the coauthoring subresponse.

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

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

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

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

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

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

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

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

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

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

SchemaLockID: A string that is globally unique and known among all protocol clients that share the same protocol version. The schema lock identifier is used by the protocol server to block other clients with different schema identifiers. After a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST allow only other protocol clients that specify the same
schema lock identifier to share the file lock. The protocol server ensures that at any instant of time, only clients having the same schema lock identifier can lock the file. After all the protocol clients have released their lock for that file, the protocol server MUST allow a protocol client with a different schema lock identifier to get a shared lock for that file. The `SchemaLockID` attribute MUST be sent on all types of coauthoring subrequests. The string "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use for this attribute.<26>

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

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

The types of coauthoring subrequests are defined in section 2.3.3.3.

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

### 2.3.1.6 CoauthSubRequestType

The **CoauthSubRequestType** complex type contains information about a coauthoring subrequest. The `SubRequestType` definition from which `CoauthSubRequestType` is extended is defined in section 2.2.4.5.

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

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

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

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

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

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

- Join coauthoring session
- Refresh coauthoring session

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

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

- Join coauthoring session
- Refresh coauthoring session
- Get coauthoring status.

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

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

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

The types of coauthoring subrequests are defined in section 2.3.3.3.

2.3.1.8 CoauthSubResponseType

The CoauthSubResponseType complex type contains information about the success or failure in processing the coauthoring subrequest. In the case of success, it contains coauthoring information requested as part of the coauthoring subrequest. In the case of failure, the ErrorCode attribute that is part of a SubResponse element specifies the error code result for this subrequest. ErrorCode is specified in section 2.2.4.8. The SubResponseType definition from which CoauthSubResponseType is extended is defined in section 2.2.4.8.
SubResponseData: A CoauthSubResponseDataType that specifies coauthoring-related information provided by the protocol server that was requested as part of the coauthoring subrequest. CoauthSubResponseDataType is defined in section 2.3.1.7. As part of processing the coauthoring subrequest, the SubResponseData element MUST be sent as part of the SubResponse element in a cell storage service response message only if the following condition is true:

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

The protocol server sets the value of the ErrorCode attribute to "Success" only if the protocol server succeeds in processing the coauthoring subrequest. ErrorCode is specified in section 2.2.4.8.

2.3.1.9 ExclusiveLockSubRequestDataType

The ExclusiveLockSubRequestDataType complex type contains information about data or input parameters used in processing an exclusive lock subrequest. The ExclusiveLockID attribute and the ExclusiveLockRequestType attribute specified in the ExclusiveLockSubRequestDataOptionalAttributes attribute group MUST both be specified for an exclusive lock subrequest. The ExclusiveLockID attribute and the ExclusiveLockRequestType attribute are specified as part of the SubRequestData element associated with an exclusive lock SubRequest element. ExclusiveLockSubRequestDataOptionalAttributes is defined in section 2.3.3.4. If the ExclusiveLockRequestType attribute is not provided, a "HighLevelExceptionThrown" error code SHOULD<28> be returned as part of the ResponseVersion element. If other attributes are not provided, an "InvalidArgument" error code MUST be returned as part of the SubResponseData element associated with the exclusive lock subresponse.

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

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

SchemaLockID: A string that is globally unique and known among all the protocol clients that share the same protocol version. The schema lock identifier is used by the protocol server to block other clients that have different schema identifiers. After a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST allow only other protocol clients that specify the same schema lock identifier to share the file lock. The protocol server ensures that at any instant in time, only clients having the same schema lock identifier can lock the document. After all the
protocol clients have released their lock for that file, the protocol server MUST allow a protocol client with a different schema lock identifier to get a shared lock for that file. The **SchemaLockID** attribute MUST be sent when the exclusive lock subrequest has an **ExclusiveLockSubRequestType** attribute set to "ConvertToSchemaLock" or "ConvertToSchemaJoinCoauth". **ExclusiveLockSubRequestType** specifies the different types of exclusive lock subrequests and is defined in section 2.3.3.4. The string "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use for this attribute.<sup>29</sup>

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

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

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

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

### 2.3.1.10 ExclusiveLockSubRequestType

The **ExclusiveLockSubRequestType** complex type contains information about an exclusive lock subrequest. The **SubRequestType** definition from which **ExclusiveLockSubRequestType** is extended is defined in section 2.2.4.5.

```xml
<xs:complexType name="ExclusiveLockSubRequestType">
  <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:complexType>
```

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

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

### 2.3.1.11 ExclusiveLockSubResponseDataType

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

```xml
<xs:complexType name="ExclusiveLockSubResponseDataType">
  <xs:attribute name="CoauthStatus" type="tns:CoauthStatusType" use="optional"/>
</xs:complexType>
```
CoauthStatus: A CoauthStatusType that specifies the coauthoring status in an exclusive lock subresponse. CoauthStatusType is defined in section 2.2.5.1. CoauthStatus MUST be specified only in an exclusive lock subresponse that is generated in response to an exclusive lock subrequest of type "Convert to schema lock with coauthoring transition tracked" if the ErrorCode attribute that is part of the SubResponse element is set to a value of "Success". The types of exclusive lock subrequest are defined in section 2.3.3.4.

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

2.3.1.12 ExclusiveLockSubResponseType

The ExclusiveLockSubResponseType complex type contains information about the success or failure in processing the exclusive lock subrequest. In the case of success, it contains information requested as part of the exclusive lock subrequest. In the case of failure, the ErrorCode attribute that is part of a SubResponse element specifies the error code result for this subrequest. ErrorCode is specified in section 2.2.4.8. The SubResponseType definition from which ExclusiveLockSubResponseType is extended is defined in section 2.2.4.8.

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

2.3.1.13 SchemaLockSubRequestDataType

The SchemaLockSubRequestDataType complex type contains information about data or input parameters used in processing a schema lock subrequest. The SchemaLockID attribute and the SchemaLockRequestType attribute specified in the SchemaLockSubRequestDataOptionalAttributes attribute group MUST both be specified for a schema lock subrequest. The SchemaLockID attribute and the SchemaLockRequestType attribute are specified as part of the SubRequestData element associated with a schema lock SubRequest element. SchemaLockSubRequestDataOptionalAttributes is defined in section 2.3.3.5. If the SchemaLockRequestType attribute is not provided, a "HighLevelExceptionThrown" error code SHOULD be returned as part of the ResponseVersion element. If other attributes are not provided, an "InvalidArgument" error code MUST be returned as part of the SubResponseData element associated with the schema lock subresponse.
SchemaLockSubRequestDataOptionalAttributes: An attribute group that specifies the set of attributes that are provided only for a SubRequestData element whose parent SubRequest element’s Type attribute is set to "SchemaLock". SchemaLockSubRequestDataOptionalAttributes is defined in section 2.3.3.5.

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

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

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

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

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

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

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

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

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

SchemaLockID: A string that is globally unique and known among all protocol clients that share the same protocol version. The schema lock identifier is used by the protocol server to block other clients that have different schema identifiers. After a protocol client is able to get a shared lock for a file with a specific schema lock identifier, the server MUST allow only other protocol clients that specify the same schema lock identifier to share the file lock. The protocol server ensures that at any instant in time, only clients having the same schema lock identifier can lock the file. After all the protocol clients have released their lock for that file, the protocol server MUST allow a protocol client with a different schema lock identifier to get a shared lock for that file. The SchemaLockID attribute MUST be sent on all types of schema lock subrequests. The string "29358EC1-E813-4793-8E70-ED0344E7B73C" has been reserved for use for this attribute. <31>
Timeout: An integer that specifies the time, in seconds, after which the shared lock for that particular file will expire for that specific protocol client. The Timeout attribute MUST be set to a value ranging from 3,600 to 120,000. When the Timeout is set to a value ranging from 60 to 3600, the server also returns success but sets the Timeout to an implementation-specific default value. When more than one client is editing the file, the protocol server MUST maintain a separate timeout value for each client. The client's timeout on a shared lock for a file is refreshed by sending a schema lock subrequest of type "Refresh lock". The Timeout attribute MUST be specified in all of the following types of schema lock subrequests:

- Get lock
- Refresh lock
- Convert to exclusive lock

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

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

2.3.1.14 SchemaLockSubRequestType

The SchemaLockSubRequestType complex type contains information about a schema lock subrequest. The SubRequestType definition from which SchemaLockSubRequestType is extended is defined in section 2.2.4.5.

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

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

2.3.1.15 SchemaLockSubResponseData

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

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

2.3.1.16 SchemaLockSubResponseType

The SchemaLockSubResponseType complex type contains information about the success or failure in processing the schema lock subrequest. In the case of success, it contains information requested as part of the schema lock subrequest. In the case of failure, the ErrorCode attribute that is part of a SubResponse element specifies the error code result for this subrequest. ErrorCode is specified in section 2.2.4.8. The SubResponseType definition from which SchemaLockSubResponseType is extended is defined in section 2.2.4.8.

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

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

2.3.1.17 ServerTimeSubRequestType

The ServerTimeSubRequestType complex type contains information about a server time subrequest. The SubRequestType definition from which ServerTimeSubRequestType is extended is defined in section 2.2.4.5. The SubRequestData element is not contained in a SubRequest element of type ServerTimeSubRequestType.

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

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

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

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

ServerTime: A positive integer that specifies the server time, which is expressed as a tick count. A single tick represents 100 nanoseconds, or one ten-millionth of a second. ServerTime specifies the number of 100-nanosecond intervals that have elapsed since 00:00:00 on January 1, 0001, which SHOULD be Coordinated Universal Time (UTC). If the request for server time information from the server is successful, the ServerTime attribute MUST be specified in a server time subresponse that is generated in response to a server time subrequest.

2.3.1.19 ServerTimeSubResponseType

The ServerTimeSubResponseType complex type contains information about the success or failure in processing the server time subrequest. In the case of success, it contains information requested as part of a server time subrequest. In the case of failure, the ErrorCode attribute that is part of a SubResponse element specifies the error code result for this subrequest. ErrorCode is specified in section 2.2.4.8. The SubResponseType definition from which ServerTimeSubResponseType is extended is defined in section 2.2.4.8.

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

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

2.3.1.20 WhoAmISubRequestType

The WhoAmISubRequestType complex type contains information about Who Am I subrequest. The SubRequestType definition from which WhoAmISubRequestType is extended is defined in section 2.2.4.5. The SubRequestData element is not contained in a SubRequest element of type, WhoAmISubRequestType.

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

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

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

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

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

2.3.1.22 WhoAmISubResponseType

The **WhoAmISubResponseType** complex type contains information about the success or failure in processing a **WhoAmI** subrequest. In the case of success, it contains information requested as part of the **WhoAmI** subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this subrequest. **ErrorCode** is specified in section 2.2.4.8. The **SubResponseType** definition from which **WhoAmISubResponseType** is extended is defined in section 2.2.4.8.

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

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

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

The protocol server sets the value of the **ErrorCode** attribute to "Success" only if the protocol server succeeds in processing the **WhoAmI** subrequest. **ErrorCode** is specified in section 2.2.4.8.

2.3.1.23 EditorsTableSubRequestDataType

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

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

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

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

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

- Join editing session
- Refresh editing session

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

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

- Join editing session
- Refresh editing session

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

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

- Update Editor Metadata
- Remove Editor Metadata

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

**Value**: Ignore.

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

The types of EditorsTable subrequests are defined in section 2.3.3.7.

### 2.3.1.24 EditorsTableSubRequestType

The EditorsTableSubRequestType complex type contains information about an editors table subrequest. The SubRequestType definition from which WhoAmISubRequestType is extended is defined in section 2.2.4.5.

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

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

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

### 2.3.1.25 EditorsTableSubResponseType

The EditorsTableSubResponseType complex type contains information about the success or failure in processing an EditorsTable (section 3.1.4.8) subrequest. In the case of failure, the ErrorCode attribute that is part of a SubResponse element specifies the error code result for this subrequest. The ErrorCode attribute is specified in section 2.2.4.8. The SubResponseType definition from which EditorsTableSubResponseType is extended is defined in section 2.2.4.8.

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

**SubResponseData:** It MUST be an empty element without any attributes. As part of processing the EditorsTable subrequest, the SubResponseData element MUST be sent as part of the SubResponse element in a cell storage service response message only if the following condition is true:
- The **ErrorCode** attribute that is part of the **SubResponse** element is set to a value of "Success".

The protocol server sets the value of the **ErrorCode** attribute to "Success" only if the protocol server succeeds in processing the editors table subrequest. **ErrorCode** is specified in 2.2.4.8.

### 2.3.1.26 GetDocMetaInfoSubRequestType

The **GetDocMetaInfoSubRequestType** complex type contains information about a **GetDocMetaInfo subrequest**. The **SubRequestType** definition from which **GetDocMetaInfoSubRequestType** is extended is defined in section 2.2.4.5. The **SubRequestData** element is not contained in a **SubRequest** element of type **GetDocMetaInfoSubRequestType**.

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

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

### 2.3.1.27 GetDocMetaInfoSubResponseDataType

The **GetDocMetaInfoSubResponseDataType** complex type contains no additional data beyond the **SubResponseDataGenericType** that it extends. **SubResponseDataGenericType** is defined in section 2.2.4.6.

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

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

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

### 2.3.1.28 GetDocMetaInfoPropertySetType

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

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

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

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

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

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

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

2.3.1.30 GetDocMetaInfoSubResponseType

The `GetDocMetaInfoSubResponseType` complex type contains information about the success or failure in processing `GetDocMetaInfo` subrequest. In the case of success, its child elements contain information requested as part of the `GetDocMetaInfo` subrequest. In the case of failure, the `ErrorCode` attribute that is part of a `SubResponse` element specifies the error code result for this subrequest. The `ErrorCode` attribute is specified in section 2.2.4.8. `SubResponseType` is defined in section 2.2.4.8.

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

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

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

The protocol server sets the value of the `ErrorCode` attribute to "Success" only if the protocol server succeeds in processing the `GetDocMetaInfo` subrequest. The `ErrorCode` attribute is specified in section 2.2.4.8.

2.3.1.31 GetVersionsSubRequestType

The `GetVersionsSubRequestType` complex type contains information about the `GetVersions` subrequest. The `SubRequestType` definition from which `GetVersionsSubRequestType` is extended is defined in section 2.2.4.5.

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

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

The GetVersionsSubResponseType complex type contains information about the success or failure in processing the GetVersions subrequest. In the case of success, it contains information requested as part of the GetVersions subrequest. In the case of failure, the ErrorCode attribute that is part of a SubResponse element specifies the error code result for this subrequest. ErrorCode is specified in section 2.2.4.8. The SubResponseType definition from which GetVersionsSubResponseType is extended is defined in section 2.2.4.8.

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

2.3.1.33 FileOperationSubRequestDataType

The FileOperationSubRequestDataType complex type contains information about data or input parameters used in processing a file operation subrequest. The FileOperation attribute specified in the FileOperationSubRequestDataOptionalAttributes attribute group (section 2.3.3.8) MUST be specified for a file operation subrequest. The FileOperation attribute is specified as part of the SubRequestData element associated with a file operation SubRequest element. If the FileOperation attribute is not provided, a "HighLevelExceptionThrown" error code SHOULD<33> be returned as part of the ResponseVersion element. If other attributes are not provided, an "InvalidArgument" error code MUST be returned as part of the SubResponseData element associated with the file operation subresponse.

FileOperationSubRequestDataOptionalAttributes: An attribute group that specifies the set of attributes that are provided only for a SubRequestData element whose parent SubRequest element's Type attribute is set to "FileOperation". FileOperationSubRequestDataOptionalAttributes is defined in section 2.3.3.8.

NewFileName: A string that specifies a new name for the file on the server. This string MUST only contain a valid filename, with no relative path. The NewFileName attribute MUST be sent only when the file operation subrequest has a FileOperationSubRequestType attribute set to "Rename".

ExclusiveLockID: A string that serves as a unique identifier for the exclusive lock on the file at the time the file operation request is executed. This parameter is used to validate that the file operation
can be performed even though the file is under exclusive lock. The `ExclusiveLockID` attribute can be optionally sent only when the file operation subrequest has a `FileOperationSubRequestType` attribute set to "Rename".

### 2.3.1.34 FileOperationSubRequestType

The `FileOperationSubRequestType` complex type contains information about a file operation subrequest. The `SubRequestType` definition, from which `FileOperationSubRequestType` is extended, is defined in section 2.2.4.5.

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

**SubRequestData**: A `FileOperationSubRequestDataType` that specifies the data or input parameters needed for processing the file operation subrequest. `FileOperationSubRequestDataType` is defined in section 2.3.1.33.

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

### 2.3.1.35 FileOperationSubResponseType

The `FileOperationSubResponseType` complex type contains information about the success or failure in processing the file operation subrequest. In the case of success, it contains information requested as part of a file operation subrequest. In the case of failure, the `ErrorCode` attribute that is part of a `SubResponse` element specifies the error code result for this subrequest. `ErrorCode` is specified in section 2.2.4.8. The `SubResponseType` definition from which `FileOperationSubResponseType` is extended is defined in section 2.2.4.8.

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

The `SubResponseData` element is empty in a `SubResponse` element of type `FileOperationSubRequestType`.

### 2.3.1.36 VersioningSubRequestDataType

The `VersioningSubRequestDataType` complex type contains information about data or input parameters used in processing a versioning subrequest.
VersioningSubRequestDataOptionalAttributes: An attribute group that specifies the set of attributes that are provided only for a SubRequestData element whose parent SubRequest element’s Type attribute is set to ”Versioning”. VersioningSubRequestDataOptionalAttributes is defined in section 2.3.3.9.

Version: A FileVersionNumberType that serves to uniquely identify a version of a file on the server. FileVersionNumberType is defined in section 2.2.5.15. Version MUST be specified when the versioning subrequest has a VersioningSubRequestType attribute set to ”RestoreVersion”. The types of versioning subrequest are defined in section 2.3.3.9.

2.3.1.37 VersioningSubRequestType

The VersioningSubRequestType complex type contains information about a versioning subrequest. The SubRequestType definition from which VersioningSubRequestType is extended is defined in section 2.2.4.5.

SubRequestData: A VersioningSubRequestDataType that specifies the data or input parameters needed for processing the versioning subrequest. VersioningSubRequestDataType is defined in section 2.3.1.36.

Type: A SubRequestAttributeType that specifies the type of the subrequest. The Type attribute MUST be set to ”Versioning” for a versioning subrequest. SubRequestAttributeType is defined in section 2.2.5.11.

2.3.1.38 VersioningSubResponseDataType

The VersioningSubResponseDataType complex type contains no additional data beyond the SubResponseDataGenericType that it extends. SubResponseDataGenericType is defined in section 2.2.4.6.
**UserTable:** An element of type *VersioningUserTableType* (section 2.3.1.40) that specifies data for the users represented in the version list. The **UserTable** element MUST be included in the response if the SubResponseType of the parent *VersioningSubResponseType* is of type "GetVersionList."

**Versions:** An element of type *VersioningVersionListType* (section 2.3.1.41) that specifies the list of versions of this file that exist on the server. The **Versions** element MUST be included in the response if the SubResponseType of the parent *VersioningSubResponseType* is of type "GetVersionList."

### 2.3.1.39 VersioningSubResponseType

The **VersioningSubResponseType** complex type contains information about the success or failure in processing the versioning subrequest. In the case of success, it contains information requested as part of a versioning subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this subrequest. **ErrorCode** is specified in section 2.2.4.8. The **SubResponse** definition from which **VersioningSubResponseType** is extended is defined in section 2.2.4.8.

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

**SubResponseData:** A **VersioningSubResponseDataType** that specifies versioning related information provided by the protocol server that was requested as part of the versioning subrequest. **VersioningSubResponseDataType** is defined in section 2.3.1.38.

### 2.3.1.40 VersioningUserTableType

The **VersioningUserTableType** complex type contains information about users that are represented in the versions of the file described by the **VersioningVersionListType**.

```xml
<xs:complexType name="VersioningUserTableType">
  <xs:sequence>
    <xs:element name="User" maxOccurs="unbounded" minOccurs="1" type="tns:UserDataType" />
  </xs:sequence>
</xs:complexType>
```

**User:** An element of type **UserDataType** (section 2.3.1.42) which describes a single user.

### 2.3.1.41 VersioningVersionListType

The **VersioningVersionListType** complex type contains a list describing the versions of the file.

```xml
<xs:complexType name="VersioningVersionListType">
  <xs:sequence>
    <xs:element name="Version" maxOccurs="unbounded" minOccurs="1" type="tns:FileVersionDataType" />
  </xs:sequence>
</xs:complexType>
```
**Version**: An element of type `FileVersionDataType` (section 2.3.1.43) which describes a single version of the file on the server.

### 2.3.1.42 UserDataType

The **UserDataType** complex type describes the details of a user represented in the `VersioningVersionListType`.

```xml
<xs:complexType name="UserDataType">
    <xs:attribute name="UserId" type="xs:integer" use="required" />
    <xs:attribute name="UserLogin" type="xs:UserLoginType" use="required" />
    <xs:attribute name="UserName" type="xs:UserNameType" use="optional" />
    <xs:attribute name="UserEmailAddress" type="xs:string" use="optional" />
</xs:complexType>
```

**UserId**: An integer that uniquely specifies the user in this user table. The `UserId` is only used to identify users in attributes of the `FileVersionDataType` and `FileVersionEventDataType` and only needs to be consistent within the `VersioningSubRequestDataType` structure.

**UserLogin**: A `UserLoginType` that specifies the user login alias of the protocol client. `UserLoginType` is defined in section 2.3.2.6.

**UserName**: A `UserNameType` that specifies the user name for the protocol client. `UserNameType` is defined in section 2.3.2.7.

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

### 2.3.1.43 FileVersionDataType

The **FileVersionDataType** complex type describes the details about a single version of the file.

```xml
<xs:complexType name="FileVersionDataType">
    <xs:sequence>
        <xs:element name="Events" minOccurs="0" maxOccurs="1">
            <xs:complexType>
                <xs:sequence>
                    <xs:element name="Event" minOccurs="1" maxOccurs="unbounded" type="tns:FileVersionEventDataType" />
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:attribute name="IsCurrent" type="tns:TRUEFALSE" use="optional" />
        <xs:attribute name="Number" type="tns:FileVersionNumberType" use="required" />
        <xs:attribute name="LastModifiedTime" type="xs:positiveInteger" use="optional" />
        <xs:attribute name="UserId" type="xs:integer" use="optional" />
    </xs:sequence>
</xs:complexType>
```

**IsCurrent**: A TRUEFALSE value that specifies if this version is the most recent version of the file. This attribute is not present if this version is not the most recent version of the file.

**Number**: A `FileVersionNumberType` (section 2.2.5.15) that specifies the unique version number of the version of the file.

**LastModifiedTime**: A positive integer that specifies the last modified time of the version of the file, which is expressed as a tick count. A single tick represents 100 nanoseconds, or one ten-millionth of a second. `LastModifiedTime` specifies the number of 100-nanosecond intervals that have elapsed since 00:00:00 on January 1, 1601, which MUST be Coordinated Universal Time (UTC).
**UserId**: An integer that specifies the user that last modified the version of the file. The number MUST match the **UserId** attribute of a **UserDataType** (section 2.3.1.42) described in the **VersioningUserTableType** in the current **VersioningSubResponseDataType**.

**Events**: An array of events of type **FileVersionEventDataType** that represents an event that happened to the version of the file.

### 2.3.1.44 FileVersionEventDataType

The **FileVersionEventDataType** complex type describes the details about a file event that happened to a specific version of a file.

```xml
<xs:complexType name="FileVersionEventDataType">
  <xs:attribute name="Id" type="xs:integer" use="required" />
  <xs:attribute name="Type" type="xs:integer" use="required" />
  <xs:attribute name="CreateTime" type="xs:positiveInteger" use="optional" />  
  <xs:attribute name="UserId" type="xs:integer" use="optional" />
</xs:complexType>
```

**Id**: An integer that uniquely identifies an event among all events to all versions of the file.

**Type**: An integer that identifies the type of event that occurred to the file. The value MUST be one of the values in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A user shared the file with another user.</td>
</tr>
<tr>
<td>2</td>
<td>A user renamed the file.</td>
</tr>
<tr>
<td>3</td>
<td>A user restored the file content to its state at a previous version.</td>
</tr>
</tbody>
</table>

**CreateTime**: A positive integer that specifies the time when this event occurred, which is expressed as a tick count. A single tick represents 100 nanoseconds, or one ten-millionth of a second. **CreateTime** specifies the number of 100-nanosecond intervals that have elapsed since 00:00:00 on January 1, 1601, which MUST be **Coordinated Universal Time (UTC)**.

**UserId**: An integer which specifies which user performed this event. The **UserId** MUST match the **UserId** attribute of a **UserDataType** (section 2.3.1.42) described in the **VersioningUserTableType** in the current **VersioningSubResponseDataType**.

### 2.3.1.45 AmIAloneSubRequestDataType

The **AmIAloneSubRequestDataType** complex type contains information about data or input parameters used in processing an AmIAlone subrequest.

```xml
<xs:complexType name="AmIAloneSubRequestDataType">
  <xs:attribute name="TransitionID" type="tns:guid" use="optional" />
</xs:complexType>
```

**TransitionID**: A **guid** that specifies the unique file identifier for that file in the protocol server.
2.3.1.46 AmIAIoneSubRequestType

The **AmIAIoneSubRequestType** complex type contains information about an AmIAIone subrequest. The **SubRequestType** definition from which **AmIAIoneSubRequestType** is extended is defined in section 2.2.4.5.

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

**SubRequestData**: An **AmIAIoneSubRequestDataType** that specifies the data or input parameters needed for processing the AmIAIone subrequest. **AmIAIoneSubRequestDataType** is defined in section 2.3.1.45.

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

2.3.1.47 AmIAIoneSubResponseDataType

The **AmIAIoneSubResponseDataType** complex type contains information requested as part of the corresponding AmIAIone subrequest.

```xml
<xs:complexType name="AmIAIoneSubResponseDataType">
  <xs:attribute name="AmIAIone" type="xs:boolean" use="optional" />
</xs:complexType>
```

**AmIAIone**: A Boolean value that specifies whether the user is alone in the coauthoring session.

2.3.1.48 AmIAIoneSubResponseType

The **AmIAIoneSubResponseDataType** complex type contains information about the success or failure in processing the AmIAIone subrequest. In the case of success, it contains information requested as part of an AmIAIone subrequest. In the case of failure, the **ErrorCode** attribute that is part of a **SubResponse** element specifies the error code result for this subrequest. **ErrorCode** is specified in section 2.2.4.8. The **SubResponseType** definition from which **AmIAIoneSubResponseType** is extended is defined in section 2.2.4.8.

```xml
<xs:complexType name="AmIAIoneSubResponseDataType">
  <xs:complexContent>
    <xs:extension base="tns:SubResponseType">
      <xs:sequence minOccurs="0" maxOccurs="1">
        <xs:element name="SubResponseData" type="tns:AmIAIoneSubResponseDataType" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
SubResponseData: An AmIAloneSubResponseDataType that specifies the information about whether the user is alone that was requested as part of the AmIAlone subrequest. AmIAloneSubResponseDataType is defined in section 2.3.1.47.

2.3.1.49 LockStatusSubRequestType

The LockStatusSubRequestType complex type contains information about a LockStatus subrequest. The SubRequestType definition from which LockStatusSubRequestType is extended is defined in section 2.2.4.5. The SubRequestData element is not contained in a SubRequest element of type, LockStatusSubRequestType.

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

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

2.3.1.50 LockStatusSubResponseDataType

The LockStatusSubResponseDataType complex type contains information requested as part of the corresponding LockStatus subrequest.

```xml
<xs:complexType name="LockStatusSubResponseDataType">
  <xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
  <xs:attribute name="LockID" type="tns:guid" use="optional" />
  <xs:attribute name="LockedBy" type="xs:string" use="optional" />
</xs:complexType>
```

LockType: A LockTypes that specifies the type of lock granted in a coauthoring subresponse or a schema lock subresponse. LockTypes is defined in section 2.2.5.9.

LockedID: A guid that specifies the id of the lock.

LockedBy: A string that specifies the user that has the file locked, if any.

2.3.1.51 LockStatusSubResponseType

The LockStatusSubResponseType complex type contains information about the success or failure in processing the LockStatus subrequest. In the case of success, it contains information requested as part of a LockStatus subrequest. In the case of failure, the ErrorCode attribute that is part of a SubResponse element specifies the error code result for this subrequest. ErrorCode is specified in section 2.2.4.8. The SubResponseType definition from which LockStatusSubResponseType is extended is defined in section 2.2.4.8.

```xml
<xs:complexType name="LockStatusSubResponseType">
  <xs:complexContent>
    <xs:extension base="tns:SubResponseType">
      <xs:sequence minOccurs="0" maxOccurs="1">
        <xs:element name="SubResponseData" type="tns:LockStatusSubResponseDataType" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
```
SubResponseData: A **LockStatusSubResponseDataType** that specifies the information about the lock status of a file that was requested as part of the LockStatus subrequest. **LockStatusSubResponseDataType** is defined in section 2.3.1.50.

### 2.3.1.52 PropertiesSubRequestDataType

The **PropertiesSubRequestDataType** complex type contains information about data or input parameters used in processing a Properties subrequest. The Properties attribute specified in the **PropertiesSubRequestDataOptionalAttributes** attribute group (section 2.3.3.10) MUST be specified for a Properties subrequest.

```xml
<xs:complexType name="PropertiesSubRequestDataType">
  <xs:sequence>
    <xs:element name="PropertyIds" minOccurs="0" maxOccurs="1" type="tns:PropertyIdsType"/>
  </xs:sequence>
  <xs:attributeGroup ref="tns:PropertiesSubRequestDataOptionalAttributes"/>
</xs:complexType>
```

**PropertyIds:** An element of type **PropertyIdsType** (section 2.3.1.56) that specifies the set of properties. This element MUST only be included in the request if the Properties attribute value is set to "PropertyGet".

**PropertiesSubRequestDataOptionalAttributes:** An attribute group that specifies the set of attributes that are provided only for a SubRequestData element whose parent SubRequest element's Type attribute is set to "Properties". **PropertiesSubRequestDataOptionalAttributes** is defined in section 2.3.3.10.

### 2.3.1.53 PropertiesSubRequestType

The **PropertiesSubRequestType** complex type contains information about a Properties subrequest. The **SubRequestType** definition from which **PropertiesSubRequestType** is extended is defined in section 2.2.4.5.

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

SubRequestData: A **PropertiesSubRequestDataType** that specifies the data or input parameters needed for processing the Properties subrequest. **PropertiesSubRequestDataType** is defined in section 2.3.1.52.

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

The PropertiesSubResponseDataType complex type contains information requested as part of the corresponding Properties subrequest.

```xml
<xs:complexType name="PropertiesSubResponseDataType">
  <xs:sequence>
    <xs:element name="PropertyIds" minOccurs="0" maxOccurs="1" type="tns:PropertyIdsType"/>
    <xs:element name="PropertyValues" minOccurs="0" maxOccurs="1" type="tns:PropertyValuesType"/>
  </xs:sequence>
</xs:complexType>
```

PropertyIds: An element of type PropertyIdsType (section 2.3.1.56) that specifies the set of properties. This element MUST only be included in the response if the Properties attribute value is set to "PropertyEnumerate".

PropertyValues: An element of type PropertyValuesType (section 2.3.1.58) that specifies the property values. This element MUST only be included in the response if the Properties attribute value is set to "PropertyGet".

2.3.1.55 PropertiesSubResponseType

The PropertiesSubResponseType complex type contains information about the success or failure in processing the Properties subrequest. In the case of success, it contains information requested as part of a Properties subrequest. In the case of failure, the ErrorCode attribute that is part of a SubResponse element specifies the error code result for this subrequest. ErrorCode is specified in section 2.2.4.8. The SubResponseType definition from which PropertiesSubResponseType is extended is defined in section 2.2.4.8.

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

SubResponseData: A PropertiesSubResponseDataType that specifies the information about the properties for the resource that was requested as part of the Properties subrequest. PropertiesSubResponseDataType is defined in section 2.3.1.54.

2.3.1.56 PropertyIdsType

The PropertyIdsType complex type represents the type of the property Ids.

```xml
<xs:complexType name="PropertyIdsType">
  <xs:sequence>
    <xs:element name="PropertyId" minOccurs="0" maxOccurs="unbounded" type="tns:PropertyIdType" />
  </xs:sequence>
</xs:complexType>
```

PropertyId: A PropertyIdType (section 2.3.1.57) that specifies a property Id.
2.3.1.57 PropertyIdType
The PropertyIdType complex type represents the type of a single property Id.

<xs:complexType name="PropertyIdType">
  <xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>

id: A string that identifies a property.

2.3.1.58 PropertyValuesType
The PropertyValuesType complex type represents the type of the property values.

<xs:complexType name="PropertyValuesType">
  <xs:sequence>
    <xs:element name="PropertyValue" minOccurs="0" maxOccurs="unbounded" type="tns:PropertyValueType" />
  </xs:sequence>
</xs:complexType>

PropertyValue: A PropertyValueType (section 2.3.1.59) that specifies a property.

2.3.1.59 PropertyValueType
The PropertyValueType complex type represents a single property value.

<xs:complexType name="PropertyValueType">
  <xs:attribute name="id" type="xs:string" use="required" />
  <xs:attribute name="value" type="xs:string" use="required" />
</xs:complexType>

</xs:complexType>id: A string that identifies a property.
value: A string that identifies the value of the property.

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

<table>
<thead>
<tr>
<th>Simple type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellRequestErrorCodeTypes</td>
<td>A subset of error codes returned for a cell subrequest as part of a cell storage service response message. CellRequestErrorCodeTypes is an enumeration of error codes specific to a cell subrequest.</td>
</tr>
<tr>
<td>CoauthRequestTypes</td>
<td>The type of the CoauthRequestType attribute, which is part of a coauthoring subrequest. CoauthRequestTypes is an enumeration of all the coauthoring request types.</td>
</tr>
<tr>
<td>ExclusiveLockRequestTypes</td>
<td>The type of the ExclusiveLockRequestType attribute, which is part of an exclusive lock subrequest. ExclusiveLockRequestTypes is an</td>
</tr>
<tr>
<td>Simple type</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SchemaLockRequestTypes</td>
<td>The type of the <code>SchemaLockRequestType</code> attribute, which is part of a schema lock subrequest. <code>SchemaLockRequestTypes</code> is an enumeration of all the schema lock request types.</td>
</tr>
<tr>
<td>UserLoginType</td>
<td>A user login value.</td>
</tr>
<tr>
<td>UserNameType</td>
<td>A user name value.</td>
</tr>
<tr>
<td>FileOperationRequestTypes</td>
<td>The type of the <code>FileOperation</code> attribute, which is part of a file operation subrequest. <code>FileOperationRequestTypes</code> is an enumeration of all the file operation request types.</td>
</tr>
<tr>
<td>VersioningRequestTypes</td>
<td>The type of the <code>VersioningRequestType</code> attribute, which is part of a versioning subrequest. <code>VersioningRequestTypes</code> is an enumeration of all the versioning request types.</td>
</tr>
<tr>
<td>PropertiesRequestTypes</td>
<td>The type of the <code>Properties</code> attribute, which is part of a Properties subrequest. <code>PropertiesRequestTypes</code> is an enumeration of all the Properties request types.</td>
</tr>
</tbody>
</table>

### 2.3.2.1 CellRequestErrorCodeTypes

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;CellRequestFail&quot;</td>
<td>Indicates an error when processing a cell subrequest for the given URL for the file.</td>
</tr>
<tr>
<td>&quot;IRMDocLibrarysOnlySupportWebDAV&quot;</td>
<td>Indicates an error when the requested file is an Information Rights Management (IRM) protected document that is supported only through Web Distributed Authoring and Versioning Protocol (WebDAV).</td>
</tr>
</tbody>
</table>
2.3.2.2 CoauthRequestTypes

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

```xml
<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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;JoinCoauthoring&quot;</td>
<td>The string value &quot;JoinCoauthoring&quot;, specifying a coauthoring subrequest of type Join Coauthoring Session.</td>
</tr>
<tr>
<td>&quot;ExitCoauthoring&quot;</td>
<td>The string value &quot;ExitCoauthoring&quot;, specifying a coauthoring subrequest of type Exit Coauthoring Session.</td>
</tr>
<tr>
<td>&quot;RefreshCoauthoring&quot;</td>
<td>The string value &quot;RefreshCoauthoring&quot;, specifying a coauthoring subrequest of type Refresh Coauthoring Session.</td>
</tr>
<tr>
<td>&quot;ConvertToExclusive&quot;</td>
<td>The string value &quot;ConvertToExclusive&quot;, specifying a coauthoring subrequest of type Convert To Exclusive Lock.</td>
</tr>
<tr>
<td>&quot;CheckLockAvailability&quot;</td>
<td>The string value &quot;CheckLockAvailability&quot;, specifying a coauthoring subrequest of type Check Lock Availability.</td>
</tr>
<tr>
<td>&quot;MarkTransitionComplete&quot;</td>
<td>The string value &quot;MarkTransitionComplete &quot;, specifying a coauthoring subrequest of type Mark Transition To Complete.</td>
</tr>
<tr>
<td>&quot;GetCoauthoringStatus&quot;</td>
<td>The string value &quot;GetCoauthoringStatus&quot;, specifying a coauthoring subrequest of type Get Coauthoring Status.</td>
</tr>
</tbody>
</table>
2.3.2.3 ExclusiveLockRequestTypes

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

```xml
<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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;GetLock&quot;</td>
<td>The string value &quot;GetLock&quot;, indicating an exclusive lock subrequest of type Get Lock.</td>
</tr>
<tr>
<td>&quot;RefreshLock&quot;</td>
<td>The string value &quot;RefreshLock&quot;, indicating an exclusive lock subrequest of type Refresh Lock.</td>
</tr>
<tr>
<td>&quot;CheckLockAvailability&quot;</td>
<td>The string value &quot;CheckLockAvailability&quot;, indicating an exclusive lock subrequest of type Check Lock Availability.</td>
</tr>
</tbody>
</table>

2.3.2.4 SchemaLockRequestTypes

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

```xml
<xs:simpleType name="SchemaLockRequestTypes">
  <xs:restriction base="xs:string">
    <!-- GetLock -->
    <xs:enumeration value="GetLock"/>
  </xs:restriction>
</xs:simpleType>
```
The value of **SchemaLockRequestTypes** MUST be one of the values in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;GetLock&quot;</td>
<td>The string value &quot;GetLock&quot;, indicating a schema lock subrequest of type Get Lock.</td>
</tr>
<tr>
<td>&quot;RefreshLock&quot;</td>
<td>The string value &quot;RefreshLock&quot;, indicating a schema lock subrequest of type Refresh Lock.</td>
</tr>
<tr>
<td>&quot;ConvertToExclusive&quot;</td>
<td>The string value &quot;ConvertToExclusive&quot;, indicating a schema lock subrequest of type Convert To Exclusive Lock.</td>
</tr>
<tr>
<td>&quot;CheckLockAvailability&quot;</td>
<td>The string value &quot;CheckLockAvailability&quot;, indicating a schema lock subrequest of type Check Lock Availability.</td>
</tr>
</tbody>
</table>

### 2.3.2.5 EditorsTableRequestTypes

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| "JoinEditingSession"   | The string value "JoinEditingSession", indicating an
<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;LeaveEditingSession&quot;</td>
<td>The string value &quot;LeaveEditingSession&quot;, indicating an editors table subrequest of type Leave Editing Session.</td>
</tr>
<tr>
<td>&quot;RefreshEditingSession&quot;</td>
<td>The string value &quot;RefreshEditingSession&quot;, indicating an editors table subrequest of type Refresh Editing Session.</td>
</tr>
<tr>
<td>&quot;UpdateEditorMetadata&quot;</td>
<td>The string value &quot;UpdateEditorMetadata&quot;, indicating an editors table subrequest of type Update Editor Metadata.</td>
</tr>
<tr>
<td>&quot;RemoveEditorMetadata&quot;</td>
<td>The string value &quot;RemoveEditorMetadata&quot;, indicating an editors table subrequest of type Remove Editor Metadata.</td>
</tr>
</tbody>
</table>

2.3.2.6 UserLoginType

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

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

UserLoginType is the type definition for the UserLogin attribute. For example, "contoso\user01".

2.3.2.7 UserNameType

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

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

UserNameType is the type definition of the UserName attribute.

2.3.2.8 FileOperationRequestTypes

The FileOperationRequestTypes simple type is used to represent the type of file operation subrequest. FileOperationRequestTypes is the type definition of the FileOperation attribute, which is part of a file operation subrequest operation.

```xml
<xs:simpleType name="FileOperationRequestTypes">
  
```

<xs:restriction base="xs:string">
  <xs:enumeration value="Rename"/>
</xs:restriction>
</xs:simpleType>

The value of **FileOperationRequestTypes** MUST be the value in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Rename&quot;</td>
<td>The string value &quot;Rename&quot;, indicating a file operation subrequest of type Rename.</td>
</tr>
</tbody>
</table>

### 2.3.2.9 VersioningRequestTypes

The **VersioningRequestTypes** simple type is used to represent the type of versioning subrequest. **VersioningRequestTypes** is the type definition of the **VersioningRequestType** attribute, which is part of a versioning subrequest operation.

```xml
<xs:simpleType name="VersioningRequestTypes">
  <xs:restriction base="xs:string">
    <xs:enumeration value="GetVersionList"/>
    <xs:enumeration value="RestoreVersion"/>
  </xs:restriction>
</xs:simpleType>
```

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;RestoreVersion&quot;</td>
<td>The string value &quot;RestoreVersion&quot;, indicating a versioning subrequest of type RestoreVersion.</td>
</tr>
</tbody>
</table>

### 2.3.2.10 PropertiesRequestTypes

The **PropertiesRequestTypes** simple type is used to represent the type of Properties subrequest. **PropertiesRequestTypes** is the type definition of the **Properties** attribute, which is part of a Properties subrequest operation.

```xml
<xs:simpleType name="PropertiesRequestTypes">
  <xs:restriction base="xs:string">
    <xs:enumeration value="PropertyEnumerate"/>
    <xs:enumeration value="PropertyGet"/>
  </xs:restriction>
</xs:simpleType>
```
The value of PropertiesRequestTypes MUST be one of the values in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;PropertyEnumerate&quot;</td>
<td>The string value &quot;PropertyEnumerate&quot;, indicating a Properties subrequest of type PropertyEnumerate.</td>
</tr>
<tr>
<td>&quot;PropertyGet&quot;</td>
<td>The string value &quot;PropertyGet&quot;, indicating a Properties subrequest of type PropertyGet.</td>
</tr>
</tbody>
</table>

### 2.3.3 Attribute Groups

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

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CellSubRequestDataOptionalAttributes</td>
<td>Contains XML schema attributes that are used in cell subrequests.</td>
</tr>
<tr>
<td>CellSubResponseDataOptionalAttributes</td>
<td>Contains XML schema attributes that are used in cell subresponses.</td>
</tr>
<tr>
<td>CoauthSubRequestDataOptionalAttributes</td>
<td>Contains XML schema attributes that are used in coauthoring subrequests.</td>
</tr>
<tr>
<td>ExclusiveLockSubRequestDataOptionalAttributes</td>
<td>Contains XML schema attributes that are used in exclusive lock subrequests.</td>
</tr>
<tr>
<td>SchemaLockSubRequestDataOptionalAttributes</td>
<td>Contains XML schema attributes that are used in schema lock subrequests.</td>
</tr>
<tr>
<td>WhoAmISubResponseDataOptionalAttributes</td>
<td>Contains XML schema attributes that are used in Who Am I subresponses.</td>
</tr>
<tr>
<td>EditorsTableSubRequestDataOptionalAttributes</td>
<td>Contains XML schema attributes that are used in Editors Table subrequests.</td>
</tr>
<tr>
<td>FileOperationSubRequestDataOptionalAttributes</td>
<td>Contains XML schema attributes that are used in File Operation subrequests.</td>
</tr>
<tr>
<td>VersioningSubRequestDataOptionalAttributes</td>
<td>Contains XML schema attributes that are used in Versioning subrequests.</td>
</tr>
<tr>
<td>PropertiesSubRequestDataOptionalAttributes</td>
<td>Contains XML schema attributes that are used in Properties subrequests.</td>
</tr>
</tbody>
</table>

#### 2.3.3.1 CellSubRequestDataOptionalAttributes

The CellSubRequestDataOptionalAttributes attribute group contains attributes that MUST be used only for SubRequestData elements associated with the parent SubRequest element where the Type attribute has a value of CellSubRequestType. The attributes in CellSubRequestDataOptionalAttributes are used as input parameters for processing the data associated with a cell subrequest. The definition of the CellSubRequestDataOptionalAttributes attribute group is as follows:
Coalesce: A Boolean value that specifies whether the protocol server SHOULD fully save all changes to the underlying store without storing the changes in any intermediate write cache after processing the subrequest. If the Coalesce attribute is set to true in a cell subrequest, the protocol server persists all changes to the file after processing the subrequest. The Coalesce attribute is set to false in the cell subrequest for updates to other partitions that do not contain binary file contents.

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

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

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

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

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

PartitionID: A guid that specifies a unique identifier for the Partition-block in which the file contents or file metadata contents has been requested to be updated. The guid type is defined in section 2.2.5.7. The Partition-block is defined in section 3.1.1. A protocol client using a value of 2.2 or greater for the VersionNumberType, as specified in section 2.2.4.9, MUST specify the Target Partition Id attribute in a cell subrequest, as specified in [MS-FSSHTTPB] section 2.2.2.1.1, and MUST NOT include this attribute if communicating with a protocol server that has a version number of 2.2 or greater. The protocol client MUST specify the PartitionID attribute if communicating with a protocol server that has a versions number of 2.0. A protocol server that has a version number of 2.2 MUST accept a PartitionID attribute from clients using protocol numbers of 2.0 and 2.2. The value of PartitionID for the file contents is "00000000-0000-0000-0000-000000000000".

The value of PartitionID MUST be one of the values in the following table.
### ExpectNoFileExists:
A Boolean value that specifies whether the protocol server can expect that no file contents can be found when an empty Etag is sent by a client during an upload of file content.

### BypassLockID:
A unique string value. If a client has got an exclusive lock, this value MUST be the same as the value of ExclusiveLockID, as specified in section 2.3.1.1. If a client has got a shared lock, this value MUST be the same as the value of SchemaLockID, as specified in section 2.3.1.1. The value of LockID, as specified in [MS-FSSHTTPB] section 2.2.2.1.4.2, is the same as this value if the value of VersionNumberType, as specified in section 2.2.4.9, is 2.2 or greater. A protocol client that has a version number of 2.2 MUST specify LockID by using the cell subrequest parameter and MUST NOT include this attribute if communicating with a protocol server that has a version number of 2.2 or greater. The protocol client MUST specify the BypassLockID attribute if communicating with a protocol server that has a version number of 2.0. A protocol server that has a version number of 2.2 MUST accept both LockID and BypassLockID.

### LastModifiedTime:
An integer that specifies the last modified time, which is expressed as a tick count. A single tick represents 100 nanoseconds, or one ten-millionth of a second. LastModifiedTime specifies the number of 100-nanosecond intervals that have elapsed since 00:00:00 on January 1, 1601, which MUST be Coordinated Universal Time (UTC). The protocol server MUST save the file with this value as the LastModifiedTime instead of the current time.

### 2.3.3.2 CellSubResponseDataOptionalAttributes

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

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

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

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

### CreateTime:
An integer that specifies the time, which is expressed as a tick count, at which the file was created. A single tick represents 100 nanoseconds, or one ten-millionth of a second. CreateTime specifies the number of 100-nanosecond intervals that have elapsed since 00:00:00 on January 1, 1601, which MUST be Coordinated Universal Time (UTC). The protocol server MUST return and
specify the **CreateTime** attribute in the cell **SubResponseData** element only when the **GetFileProps** attribute is set to **true** in the cell subrequest.

**LastModifiedTime**: An integer that specifies the last modified time, which is expressed as a tick count. A single tick represents 100 nanoseconds, or one ten-millionth of a second. **LastModifiedTime** specifies the number of 100-nanosecond intervals that have elapsed since 00:00:00 on January 1, 1601, which MUST be Coordinated Universal Time (UTC). The protocol server MUST return and specify the **LastModifiedTime** attribute in the cell **SubResponseData** element only when the **GetFileProps** attribute is set to **true** in the cell subrequest.

**ModifiedBy**: A **UserNameType** that specifies the user name for the protocol client that last modified the file. **UserNameType** is defined in section 2.3.2.7.

**CoalesceErrorMessage**: A string that specifies a description of the error that occurs when the protocol server fully saves all changes with the underlying file provider. **CoalesceErrorMessage** also specifies information about what was expected by the protocol server. **CoalesceErrorMessage** MUST be sent only when the **CoalesceHResult** attribute is set to an integer value which is not equal to 0.<37>

**CoalesceHResult**: An integer that MUST be 0 except when the protocol server attempts to fully save all the changes in the underlying store. It specifies the HRESULT when the protocol server attempts to fully save all the changes in the underlying store. **CoalesceHResult** MUST be set to a value ranging from -2,147,483,648 through 2,147,483,647. A **CoalesceHResult** value of 0 indicates success. If **CoalesceHResult** is not equal to 0, it indicates an exception or failure condition that occurred.<38>

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

**HaveOnlyDemotionChanges**: A TRUEFALSE value that specifies whether the returned binary content has only protocol server property demotion changes compared to the protocol client’s last synced status. Property demotion changes are repeatable, automated changes made by a protocol server following a protocol client’s **Put Changes** request as specified in [MS-FSSHTTPB] section 2.2.2.1.4. If the value is set to **False**, the returned binary content contains changes other than property demotion changes.

### 2.3.3.3 CoauthSubRequestDataOptionalAttributes

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

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

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

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

In the following table, “Yes” signifies that the attribute MUST be specified as part of the **SubRequestData** element associated with the coauthoring subrequest.

<table>
<thead>
<tr>
<th>Value of CoauthRequest Type</th>
<th>Timeout</th>
<th>AllowFallbackToExclusive</th>
<th>ExclusiveLockID</th>
<th>Release lock on conversion failure</th>
<th>ClientID</th>
<th>SchemaLockID</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;JoinCoauthoring&quot;</td>
<td>Yes</td>
<td></td>
<td></td>
<td>If <strong>AllowFallbackToExclusive</strong> is set to true, the attribute for the exclusive lock identifier MUST be specified.</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(Join Coauthoring)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;ExitCoauthoring&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(Exit Coauthoring)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;RefreshCoauthoring&quot;</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(Refresh Coauthoring)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;ConvertToExclusive&quot;</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(Convert To Exclusive Lock Coauthoring)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;CheckLockAvailability&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>(Check Lock Availability)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;MarkTransitionComplete&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>(Mark Coauthoring Transition Complete)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;GetCoauthoring Status&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
2.3.3.4 ExclusiveLockSubRequestDataOptionalAttributes

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

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

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

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

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

In the following table, “Yes” signifies that the attribute MUST be specified as part of the **SubRequestData** element associated with the exclusive lock subrequest.

<table>
<thead>
<tr>
<th>Value of Exclusive LockRequestType</th>
<th>Timeout</th>
<th>SchemaLockID</th>
<th>ClientID</th>
<th>ExclusiveLockID</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;GetLock&quot; (Get Lock)</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;ReleaseLock&quot; (Release Lock)</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;RefreshLock&quot; (Refresh Lock)</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;ConvertToSchemaJoinCoauth&quot; (Convert To Schema Lock With Coauthoring Transition Tracked)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>&quot;ConvertToSchema&quot; (Convert To Schema Lock)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Value of Exclusive LockRequestType</td>
<td>Timeout</td>
<td>SchemaLockID</td>
<td>ClientID</td>
<td>ExclusiveLockID</td>
</tr>
<tr>
<td>----------------------------------------</td>
<td>---------</td>
<td>--------------</td>
<td>----------</td>
<td>----------------</td>
</tr>
<tr>
<td>&quot;CheckLockAvailability&quot; (Check Lock Availability)</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 2.3.3.5 SchemaLockSubRequestDataOptionalAttributes

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

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

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

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

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

In the following table, “Yes” signifies that the attribute MUST be specified as part of the `SubRequestData` element associated with the schema lock subrequest.

<table>
<thead>
<tr>
<th>Value of SchemaLockRequestType</th>
<th>Time out</th>
<th>AllowFallbackToExclusive</th>
<th>ExclusiveLockID</th>
<th>Release lock on conversion failure</th>
<th>ClientID</th>
<th>SchemaLockID</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;GetLock&quot; (Get Lock)</td>
<td>Yes</td>
<td></td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

| "ReleaseLock" (Release Lock)        |          |                          |                 |                                   | Yes      | Yes          |

| "RefreshLock" (Refresh Lock)       | Yes      |                          |                 |                                   | Yes      | Yes          |

| "ConvertToExclusive" (Convert To Exclusive Lock) | Yes | Yes | Yes | Yes | Yes | Yes |

---

[MS-FSSHTTP] - v20190924
File Synchronization via SOAP over HTTP Protocol
Copyright © 2019 Microsoft Corporation
Release: September 24, 2019
2.3.3.6 WhoAmISubResponseDataOptionalAttributes

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

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

**UserName:** A **UserNameType** that specifies the user name for the protocol client. **UserNameType** is defined in section 2.3.2.7.

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

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

**UserIsAnonymous:** A Boolean value that specifies whether the protocol client is an anonymous guest user. Protocol servers SHOULD return TRUE if the user is an anonymous guest user.<39>

**UserLogin:** A **UserLoginType** that specifies the user login alias of the protocol client. **UserLoginType** is defined in section 2.3.2.6. The **UserLogin** attribute MUST be specified in a WhoAmI subresponse that is generated in response to a WhoAmI subrequest.

2.3.3.7 EditorsTableSubRequestDataOptionalAttributes

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

```xml
<xs:attributeGroup name="EditorsTableSubRequestDataOptionalAttributes">
  <xs:attribute name="EditorsTableRequestType" type="tns:EditorsTableRequestTypes" use="optional"/>
</xs:attributeGroup>
```
EditorsTableRequestType: An EditorsTableRequestType that specifies the type of editors table subrequest. EditorsTableRequestTypes is defined in section 2.3.2.5.

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

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

In the following table, “Yes” signifies that the attribute MUST be specified as part of the SubRequestData element associated with the editors table subrequest.

<table>
<thead>
<tr>
<th>Value of EditorsTableRequestType</th>
<th>Timeout</th>
<th>ClientID</th>
<th>AsEditor</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;JoinEditingSession&quot;</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Join Editing Session)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;LeaveEditingSession&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Leave Editing Session)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;RefreshEditingSession&quot;</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Refresh Editing Session)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;UpdateEditorMetadata&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(Update Editor Metadata)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;RemoveEditorMetadata&quot;</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td>Yes</td>
</tr>
<tr>
<td>(Remove Editor Metadata)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2.3.3.8 FileOperationSubRequestDataOptionalAttributes

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

```
<xs:attributeGroup name="FileOperationSubRequestDataOptionalAttributes">
  <xs:attribute name="FileOperation" type="tns:FileOperationRequestTypes" use="optional"/>
</xs:attributeGroup>
```

FileOperation: A FileOperationRequestTypes that specifies the type of file operation subrequest. FileOperationRequestTypes is defined in section 2.3.2.8.
Depending on the type of the file operation subrequest, the following table shows a mapping between the type of file operation subrequest and the attributes that MUST be specified for that FileOperation, signified by “Yes”.

<table>
<thead>
<tr>
<th>Value of FileOperation</th>
<th>NewFileName</th>
<th>ExclusiveLockID</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Rename” (Rename)</td>
<td></td>
<td>Yes</td>
</tr>
</tbody>
</table>

2.3.3.9 VersioningSubRequestDataOptionalAttributes

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

```xml
<xs:attributeGroup name="VersioningSubRequestDataOptionalAttributes">
  <xs:attribute name="VersioningRequestType" type="tns:VersioningRequestTypes" use="optional"/>
</xs:attributeGroup>
```

VersioningRequestType: A VersioningRequestTypes that specifies the type of versioning subrequest. VersioningRequestTypes is defined in section 2.3.2.9.

Depending on the type of the versioning subrequest, the following table shows a mapping between the type of versioning subrequest and the attributes that MUST be specified for that VersioningRequestType, signified by “Yes”.

<table>
<thead>
<tr>
<th>Value of VersioningRequestType</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;GetVersionList&quot; (Get Version List)</td>
<td></td>
</tr>
<tr>
<td>&quot;RestoreVersion&quot; (Restore Version)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

2.3.3.10 PropertiesSubRequestDataOptionalAttributes

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

```xml
<xs:attributeGroup name="PropertiesSubRequestDataOptionalAttributes">
  <xs:attribute name="Properties" type="tns:PropertiesRequestTypes" use="optional"/>
</xs:attributeGroup>
```

Properties: A PropertiesRequestTypes (section 2.3.2.10) that specifies the type of Properties subrequest.
Depending on the type of the Properties subrequest, the following table shows a mapping between the type of Properties subrequest and the elements that MUST be specified for that Properties, signified by "Yes".

<table>
<thead>
<tr>
<th>Value of Properties</th>
<th>PropertyIds</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;PropertyEnumerate&quot;</td>
<td></td>
</tr>
<tr>
<td>&quot;PropertyGet&quot;</td>
<td>Yes</td>
</tr>
</tbody>
</table>
3 Protocol Details

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

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

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

3.1 Server Details

3.1.1 Abstract Data Model

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

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

- **Partition-block**: A partition that contains binary file data in the abstract data model as specified in [MS-FSSHTTPB]. A file is associated with one or more partitions. The server stores the binary file contents and file format–specific data in partitions. File contents are stored in the default partition. Each partition is uniquely identified by a partition identifier.

- **File coauthoring tracker**: A tracking mechanism that the protocol server uses for coauthoring. For each file, the protocol server keeps track of the protocol client identifier associated with each client that is coauthoring the file, the type of lock on the file—that is, an exclusive lock or a shared lock—and the timeout associated with each client’s shared lock on the coauthorable file.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Message Processing Events and Sequencing Rules

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

<table>
<thead>
<tr>
<th>Operation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cell subrequest</td>
<td>Retrieves or uploads a file’s binary contents or a file’s metadata contents.</td>
</tr>
<tr>
<td>Coauth subrequest</td>
<td>Gets a shared lock on a coauthorable file that allows for all clients with the same schema lock identifier to share the lock. The protocol server also keeps track</td>
</tr>
<tr>
<td>Operation</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td><strong>SchemaLock subrequest</strong></td>
<td>Gets a shared lock on a coauthorable file that allows all clients with the same schema lock identifier to share the lock.</td>
</tr>
<tr>
<td><strong>ExclusiveLock subrequest</strong></td>
<td>Gets an exclusive lock on the file, which ensures only one client edits the file at an instant in time.</td>
</tr>
<tr>
<td><strong>WhoAmI subrequest</strong></td>
<td>Retrieves the protocol client’s friendly name and other client-specific information for a client with a unique client identifier.</td>
</tr>
<tr>
<td><strong>ServerTime subrequest</strong></td>
<td>Retrieves the server time.</td>
</tr>
<tr>
<td><strong>EditorsTable subrequest</strong></td>
<td>Adds the protocol client to the editors table, which is accessible to all clients editing or reading a document.</td>
</tr>
<tr>
<td><strong>GetDocMetaInfo subrequest</strong></td>
<td>Retrieves various properties for the file and the parent folder as a series of string pairs.</td>
</tr>
<tr>
<td><strong>GetVersions subrequest</strong></td>
<td>Sends back information about the previous versions of a file.</td>
</tr>
<tr>
<td><strong>Versioning subrequest</strong></td>
<td>Requests information on versions of a file or different types of versioning related operations on a file.</td>
</tr>
<tr>
<td><strong>FileOperation subrequest</strong></td>
<td>Operates on a file.</td>
</tr>
<tr>
<td><strong>AmIAlone subrequest</strong></td>
<td>Finds out if the user is alone in a coauthoring session.</td>
</tr>
<tr>
<td><strong>LockStatus subrequest</strong></td>
<td>Gets the lock status.</td>
</tr>
<tr>
<td><strong>Properties subrequest</strong></td>
<td>Gets the information on properties.</td>
</tr>
</tbody>
</table>

3.1.4.1 Common Message Processing Rules and Events

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

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

- The **SubRequestToken** attribute in the cell storage service **SubRequest** element uniquely identifies a subrequest for a file. The **SubRequestToken** attribute is specified in section 2.2.4.5.

- The **Type** attribute in the cell storage service **SubRequest** element specifies the type of cell storage service subrequest. The protocol server uses the **Type** attribute to identify the type of cell storage service subrequest. The **Type** attribute is specified in section 2.2.4.4.

- The **DependencyType** attribute in the cell storage service **SubRequest** element specifies the type of dependency that a subrequest has on another subrequest. **DependencyType** attribute for a **SubRequest** element is specified in section 2.2.4.5. The protocol server identifies the dependency type to check if the subrequest is to be processed.

- The **DependsOn** attribute specifies the **SubRequestToken** of the subrequest that this subrequest depends on. The **DependsOn** attribute for a **SubRequest** element is specified in section 2.2.4.5.
The protocol server uses a combination of the \texttt{DependsOn} and \texttt{DependencyType} attributes for a specific subrequest to decide if a cell storage service subrequest will be executed. The protocol server sends a \texttt{Response} element for each \texttt{Request} element and a \texttt{SubResponse} element corresponding to each \texttt{SubRequest} element contained within a \texttt{Request} element.

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

- The coauthoring subrequest
- The schema lock subrequest

If the protocol server supports the coauthoring subrequest, it MUST support tracking the coauthoring transition. The coauthoring transition allows for the number of users editing the file to increase from 1 to \(n\) or to decrease from \(n\) to 1, where \(n\) is the maximum number of users who are allowed to edit a single file at an instant in time. If the protocol server supports the coauthoring subrequest, it MUST return a coauthoring status as specified in section 2.3.1.7.

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

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

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

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

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

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

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

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

3.1.4.2 Cell Subrequest

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

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

- The protocol client prepares a request containing a **URL** for the file, a unique **Request** token, and one or more **SubRequest** elements, as defined in section 2.2.3.2 and section 2.2.3.8. The **SubRequest** element is of type "Cell", and the **SubRequestData** element contains attributes that are input parameters used by the server when processing the cell subrequest. The **SubRequestData** element is of type **CellSubRequestDataType** and is defined in section 2.3.1.1.

- The binary data sent in the **SubRequestData** element indicates if this is a file content upload or download. The meaning of the data is as specified in [MS-FSSHTTPB] section 2.2.3.1.3. The cell subrequest for an upload of the file contents is processed as specified in [MS-FSSHTTPB] section 3.1.4.3. The cell subrequest for a download of the file contents is processed as specified in [MS-FSSHTTPB] section 3.1.4.2. The schema lock identifier MUST be specified in a cell subrequest if the current client already shared the lock on the file and is in a coauthoring session. The schema lock identifier is defined in section 2.3.1.1. When the schema lock identifier is specified in the cell subrequest, the protocol server returns a success error code if the file currently has a shared lock with the specified schema lock identifier.

- The bypass lock identifier MUST be present when uploading the binary or metadata contents of a file. It MUST NOT be present when retrieving the binary or metadata contents of a file. The bypass lock identifier is defined in section 2.3.3.1. When the bypass lock identifier is specified in a cell subrequest, exactly one of the following is true:
  - The file is schema locked. In this case, the bypass lock identifier MUST be the same as the schema lock identifier. They have the same semantics, in this case.
  - The file is exclusive locked. In this case, the bypass lock identifier MUST be the same as the exclusive lock identifier. They have the same semantics, in this case. The exclusive lock identifier is defined in section 2.3.3.1.

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

- If the **Coalesce** attribute is set to **true** in a cell subrequest, the protocol server persists all changes to the file in the underlying store. The **Coalesce** attribute is defined in section 2.3.3.1. For all first-time saves of the file's binary file contents in a partition, the following MUST be specified:
  - The **ExclusiveLockID** attribute (section 2.3.1.1) is set to a unique lock identifier.

- When the protocol server receives the **ExclusiveLockID** in a cell subrequest, the server performs the following steps as an atomic operation:
  1. Get an exclusive lock on the file.
  2. Commit the file’s binary contents, as specified in [MS-FSSHTTPB] section 3.1.4.3. (Note that for updates to other partitions that do not contain binary file contents, the **Coalesce** attribute is set to **false** in the cell subrequest.)

- The protocol server receives the request and parses the logic to send the information to the underlying store. If the request is a download request, the encoded file contents are sent back to
the protocol client. If file properties–specific information is requested, the file properties are prepared as a response and sent back to the protocol client.

- The **Response** element is defined in section 2.2.3.5, and the **SubResponse** element is defined in section 2.2.3.10. The **CellSubResponseDataType** specifies the type of the **SubResponseData** element inside the cell **SubResponse** element. The **CellSubResponseDataType** is defined in section 2.3.1.3. In the case where the protocol server finishes performing the request but in the middle of generating the **SubResponseData** data, another request changes the file content or metadata, the protocol server returns an additional **SubResponseStreamInvalid** element to indicate that the binary data in the **SubResponseData** is not valid. The protocol client can resend the request to get the data. The **SubResponseStreamInvalid** element is defined in 2.3.1.4.

- The protocol server returns results based on the following conditions:
  - If the protocol server was unable to find the URL for the file specified in the **Url** attribute, the protocol server reports a failure by returning an error code value set to "CellRequestFail" in the **ErrorCode** attribute sent back in the **SubResponse** element, and the binary data in the returned **SubRequestData** element indicates an HRESULT as described in [MS-FSSHTTPB] section 2.2.3.2.4. But for **Put Changes** subrequest, as described in [MS-FSSHTTPB] section 2.2.2.1.4, the protocol server creates a new file using the specified **Url**.
  - Depending on the type of error, the **ErrorCode** returned as an attribute of the **SubResponse** element is updated with a specific error code as specified in section 2.2.5.4.
  - **ErrorCode** includes "Success" to indicate success in processing the file upload or download request.

### 3.1.4.3 Coauth Subrequest

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

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

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

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

- The protocol client prepares a request containing a **URL** for the file, a unique **Request** token, and one or more **SubRequest** elements, as defined in section 2.2.3.2 and section 2.2.3.8.
**SubRequest** element is of type "Coauth", and the **SubRequestData** element contains attributes that are input parameters used by the server when processing the coauthoring subrequest. The **SubRequestData** element is of type **CoauthSubRequestDataType** and is defined in section 2.3.1.5.

- The protocol server receives the request and parses the logic. Depending on the type of coauthoring request, the protocol server processes the subrequest as specified in section 3.1.4.3.1, 3.1.4.3.2, 3.1.4.3.3, 3.1.4.3.4, 3.1.4.3.5, 3.1.4.3.6, or 3.1.4.3.7. The protocol server uses the **ClientID** attribute sent in a coauthoring subrequest to do the following:
  - Uniquely identify each client
  - Keep track of each client and its timeout on the shared lock for the file
  - Decide when to release the shared lock on the file

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

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

- The protocol server returns results based on the following conditions:
  - Depending on the type of error, **ErrorCode** is returned as an attribute of the **SubResponse** element. The **ErrorCode** attribute that is part of the **SubResponse** element is updated with a specific error code as specified in section 2.2.5.4.
  - If the protocol server was unable to find the URL for the file specified in the **Url** attribute, the protocol server reports a failure by returning an error code value set to "FileNotExistsOrCannotBeCreated" in the **ErrorCode** attribute sent back in the **SubResponse** element.
  - The protocol server returns an error code value set to "CoauthRefblobConcurrencyViolation" when there is a concurrency violation and the coauthoring subrequest is one of the following types:
    - Join coauthoring session
    - Exit coauthoring session
    - Refresh coauthoring session
    - Convert to an exclusive lock
    - Mark in transition complete
  - A concurrency violation happens when a current client’s request to save the file coauthoring tracker fails because another client’s request to edit and save the file coauthoring tracker is in progress on the server before the save is done by the current client. "CoauthRefblobConcurrencyViolation" is specified in section 2.2.5.8. The file coauthoring tracker is defined in section 3.1.1.
The protocol server returns an error code value set to "RequestNotSupported" if the server does not support this request type.

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

### 3.1.4.3.1 Join Coauthoring Session

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

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

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

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

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

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

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

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

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

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

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

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

### 3.1.4.3.2 Exit Coauthoring Session

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

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

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

- Return an error code of "InvalidCoauthSession"<41>, if there are other clients present in the coauthoring session.
- Return an error code of "FileNotLockedOnServer" if no clients are present in the coauthoring session.

The protocol server SHOULD<42> return an error code of "FileAlreadyLockedOnServer" if there is a current exclusive lock on the file or a shared lock on the file with a different schema lock identifier.

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

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

### 3.1.4.3.3 Refresh Coauthoring Session

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

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

- If the coauthoring feature is enabled on the protocol server, the server considers this coauthoring subrequest to be of type, "Join coauthoring session", and it gets a new shared lock on the file.
- If the coauthoring feature is disabled, then the protocol server returns an error code value set to "FileNotLockedOnServer".<43>

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

After the protocol server has ensured that there is a shared lock with the same schema lock identifier used by the current client and that the client is sharing the shared lock on the file, the protocol server MUST update the client’s timeout on the shared lock on the file. The new timeout value is the timeout value sent as part of the coauthoring subrequest. If the client is sending a request to refresh the shared lock with a timeout value less than or equal to the current timeout on the shared lock, the protocol server considers the coauthoring subrequest of type "Refresh coauthoring session" to be a no-operation instruction and does nothing. If the client is sending a request to refresh the shared lock with a timeout value greater than the current timeout on the shared lock, the protocol server updates the file coauthoring tracker with the new timeout value for that specific client and that specific file. The **Timeout** attribute is defined in section 2.3.1.5, and the file coauthoring tracker is defined in section 3.1.1.

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

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

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

### 3.1.4.3.4 Convert to Exclusive Lock

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

The protocol server performs the following two operations:

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

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

- There is no shared lock.
- There is no coauthoring session for the file.
- The current client is not present in the coauthoring session.
If there is a current exclusive lock on the file or if there is a shared lock on the file from another client with a different schema lock identifier, the request cannot be processed successfully so the protocol server returns "InvalidCoauthSession" as defined in section 2.2.5.8.

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

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

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

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

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

3.1.4.3.5 Check Lock Availability

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

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

3.1.4.3.6 Mark Transition to Complete

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

A coauthoring subrequest of type "Mark transition to complete" is requested by a protocol client that first joined a coauthoring session or by a protocol client that has a coauthoring status of "Alone". The request is sent by the protocol client when it receives "false" from the protocol server as a response to the IsOnlyClient web service request. The IsOnlyClient web service request is as specified in [MS-SHDACCWS].
The following diagram specifies a sequence in which a coauthoring subrequest of type "Mark transition to complete" is sent by Client1 (C1). C1 is the first client that joined the coauthoring session to edit the file. Client2 (C2) is the second client sharing the lock and joining the coauthoring session.

![Diagram of coauthoring subrequest sequence]

**Figure 3: Sequence of coauthoring subrequest types**

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

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

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

- There is no shared lock.<44>
- There is no coauthoring session for the file.<45>
- The current client is not present in the coauthoring session.

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

**3.1.4.3.7 Get Coauthoring Session**
If the CoauthRequestType attribute is set to "GetCoauthoringStatus", the protocol server considers this coauthoring subrequest to be of type, "Get coauthoring status". The protocol server checks the coauthoring status of the coauthorable file. The protocol client keeps sending coauthoring subrequests of type, "Get coauthoring status" while it is actively coauthoring to check if it is the only client editing the file. If the number of clients editing the file goes back to one, the protocol client stops sending the "Get coauthoring status" request and sends the IsOnlyClient web service request as specified in [MS-SHDAACCWS].

If the coauthoring session does not exist or the current client is not present in the coauthoring session, the protocol server returns an error code value set to "Success".<46>. If the current client is the only client editing the coauthorable file, the protocol server MUST set the CoauthStatus attribute value to "Alone", indicating that no one else is editing the file. If the current client is the second, third, or later client trying to edit the document, the protocol server MUST return a CoauthStatus set to "Coauthoring", which indicates that the current client is coauthoring when editing the document.

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

### 3.1.4.4 SchemaLock Subrequest

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

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

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

The SchemaLockRequestType attribute is defined in section 2.3.2.4. The SubRequestData element for the schema lock subrequest is of type SchemaLockSubRequestDataType and is defined in section 2.3.1.13.

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

- The protocol client prepares a request containing a URL for the file, a unique Request token, and one or more SubRequest elements, as defined in section 2.2.3.2 and section 2.2.3.8. The SubRequest element is of type "SchemaLock" and the SubRequestData element contains attributes that are input parameters used by the protocol server when processing the schema lock subrequest. The SubRequestData element is of type, SchemaLockSubRequestDataType and is defined in section 2.3.1.13.
The protocol server receives the request and parses the logic, and depending on the type of schema lock subrequest, the protocol server processes the request as specified in section 3.1.4.4.1, 3.1.4.4.2, 3.1.4.4.3, 3.1.4.4.4, or 3.1.4.4.5. The protocol server uses the ClientID attribute sent in a schema lock subrequest to uniquely identify each client and keep track of each client’s timeout on the shared lock for the file. The protocol server also uses the ClientID sent in the schema lock subrequest to decide when to release the shared lock on the file. The protocol server does the following to decide when to release the shared lock on the file:

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

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

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

The protocol server returns results based on the following conditions:

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

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

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

A concurrency violation happens when a current client’s request to save the file coauthoring tracker fails because another client’s request to edit and save the file coauthoring tracker is in progress on the server before the save is done by the current client. "CoauthRefblobConcurrencyViolation" is specified in section 2.2.5.8. The file coauthoring tracker is defined in section 3.1.1.

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

An ErrorCode value of "Success" indicates success in processing the schema lock request.
3.1.4.4.1 Get Lock

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

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

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

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

- If the AllowFallbackToExclusive attribute is set to true, the protocol server gets an exclusive lock on the file.
- If the AllowFallbackToExclusive attribute is set to false, the protocol server returns an error code value set to "FileNotLockedOnServer".<47>

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

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

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

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

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

3.1.4.4.2 Release Lock

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

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

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

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

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

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

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

**LockCoauthRelatedErrorCodeTypes** is defined in section 2.2.5.8, and generic error code types are defined in section 2.2.5.6. For other unknown error types, the protocol server returns an error code value set to "LockRequestFail".

### 3.1.4.4.3 Refresh Lock

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

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

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

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

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

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

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

The protocol server performs the following operations:

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

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

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

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

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

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

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

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

### 3.1.4.4.5 Check Lock Availability

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

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

3.1.4.5 ExclusiveLock Subrequest

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

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

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

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

- The protocol client prepares a request containing a URL for the file, a unique Request token, and one or more SubRequest elements, as defined in section 2.2.3.2 and section 2.2.3.8. The SubRequest element is of type "ExclusiveLock" and the SubRequestData element contains attributes that are input parameters used by the protocol server when processing the exclusive lock subrequest. The SubRequestData element is of type ExclusiveLockSubRequestDataType and is defined in section 2.3.1.9.

- The protocol server receives the request, parses the logic, and depending on the type of exclusive lock subrequest, processes the request as specified in section 3.1.4.5.1, 3.1.4.5.2, 3.1.4.5.3, 3.1.4.5.4, 3.1.4.5.5, or 3.1.4.5.6.

- The Response element is defined in section 2.2.3.5, and the SubResponse element is defined in section 2.2.3.10. The ExclusiveLockSubResponseType defines the type of the SubResponseData element inside the exclusive lock SubResponse element. The ExclusiveLockSubResponseType is defined in section 2.3.1.11.

- The protocol returns results based on the following conditions:

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

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

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

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

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

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

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

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

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

`DependencyCheckRelatedErrorCodeTypes` is defined in section 2.2.5.2. `LockCoauthRelatedErrorCodeTypes` is defined in section 2.2.5.8.

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

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

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

- The file is already locked with a shared lock.

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

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

### 3.1.4.5.3 Refresh Lock

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

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

- The protocol server is unable to refresh the lock on the file because a shared lock already exists on the file.

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

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

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

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

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

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

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

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

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

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

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

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

3.1.4.5.5 Convert to Schema Lock

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

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

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

- If the feature of transitioning a file that currently has an exclusive lock to one that has a shared
  lock is not supported by the protocol server, the protocol server returns an error code value set to
  "RequestNotSupported".

- If the feature of coauthoring is not completely supported by the protocol server, the protocol
  server returns an error code value set to "LockNotConvertedAsCoauthDisabled".

- If the protocol server is unable to convert the exclusive lock to a shared lock on the file because
  the file is checked out by the current user, the protocol server returns an error code value set to
  "ConvertToSchemaFailedFileCheckedOutByCurrentUser".

- If the protocol server is unable to convert the lock because either there is an exclusive lock with a
different exclusive lock identifier or there is a shared lock already present on the file, the protocol
  server returns an error code value set to "FileAlreadyLockedOnServer".

- If the protocol server is unable to convert the lock on the file because no lock exists on the server,
  the protocol server returns an error code value set to "FileNotLockedOnServer".

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

- If the protocol server encounters unknown exceptions or failures when converting the lock on the
  file, the protocol server returns an error code value set to "LockRequestFail".

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

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

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

- If there is a current exclusive lock on the file with a different exclusive lock identifier than the one specified by the current client or if there is a shared lock on the file, the protocol server returns an error code value set to "FileAlreadyLockedOnServer".

- If the file is checked out on the server, but it is checked out by a client with a different user name than that of the current client, the protocol server returns an error code value set to "FileAlreadyCheckedOutOnServer".

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

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

3.1.4.6 WhoAmI Subrequest

This operation is used to retrieve the current client’s user information, which helps in showing a client’s friendly name when a coauthorable file is being edited by more than one client.

The protocol client sends a WhoAmI SubRequest message, which is of type WhoAmISubRequestType as specified in section 2.3.1.20. The protocol server responds with a WhoAmI SubResponse message, which is of type WhoAmISubResponseType as specified in section 2.3.1.22. This is done as follows:

- 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".

- The protocol server receives the request and parses the logic to request the client-specific user information. The requested client-specific user information is prepared as a response and sent back to the protocol client.

The Response element is defined in section 2.2.3.5 and the SubResponse element is defined in section 2.2.3.10. The WhoAmISubResponseData element is defined in section 2.3.1.21. The protocol server sends the requested client-specific user information as attributes in the WhoAmI SubResponseData element. The attributes sent by the protocol server are defined in section 2.3.3.6.

The protocol server returns results based on the following conditions:

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

### 3.1.4.7 ServerTime Subrequest

This operation is used to retrieve the server time, which is expressed as the number of ticks (that is, the number of 100-nanosecond intervals) that have elapsed since 00:00:00, January 1, 0001. The server time SHOULD<51> be expressed in **Coordinated Universal Time (UTC)**.

The protocol client sends a **ServerTime SubRequest** message, which is of **ServerTimeSubRequestType** as specified in section 2.3.1.17. The protocol server responds with a **ServerTime SubResponse** message, which is of type **ServerTimeSubResponseType** as specified in section 2.3.1.19. This is done as follows:

- 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".
- The protocol server receives the request and gets the server time information from the server. The requested server time information is prepared as a response and sent back to the protocol client.

The **Response** element is defined in section 2.2.3.5, and the **SubResponse** element is defined in section 2.2.3.10. The **ServerTimeSubResponseDataType** defines the type of the **SubResponseData** element that is sent in a server time **SubResponse** element. The **ServerTimeSubResponseDataType** is defined in section 2.3.1.18. The protocol server sends the server time as a **ServerTime** attribute in the **ServerTime SubResponseData** element. The **ServerTime** attribute is defined in section 2.3.1.18.

The protocol results based on the following conditions:

- If the processing of the **ServerTime subrequest** by the server fails to get the server time or encountered an unknown exception, the protocol server returns an error code value set to "SubRequestFail".
- Otherwise, the protocol server sets the error code value to "Success" to indicate success in processing the **ServerTime** subrequest.

### 3.1.4.8 EditorsTable Subrequest

This operation is used to request that the protocol server store an editors table entry on the file for the client or perform other editors table operations on a file.<52><53> Depending on the **EditorsTableRequestType** attribute value, the protocol server interprets the **subrequest** as one of the following types of editors table operations:

- Join editing session
- Leave editing session
- Refresh editing session
- Update editor metadata
- Remove editor metadata

The **EditorsTableRequestType** attribute is defined in section 2.3.3.7. The **EditorsTableRequestType** attribute is one of the attributes for the **EditorsTableSubRequestData** element, which is of type **EditorsTableSubRequestDataType**. The **EditorsTableSubRequestDataType** is defined in section 2.3.1.23.
The protocol client sends an editors table SubRequest message, which is of type EditorsTableSubRequestType as specified in section 2.3.1.24. The protocol server responds with an editors table SubResponse message, which is of type EditorsTableSubResponseType as specified in section 2.3.1.25. This is done as follows:

- The protocol client prepares a request containing a URL for the file, a unique request token, and one or more SubRequest elements, as defined in section 2.2.3.2 and section 2.2.3.8. The SubRequest element is of type "EditorsTable", and the SubRequestData element contains attributes that are input parameters used by the server when processing the editors table subrequest. The SubRequestData element is of type EditorsTableSubRequestDataType and is defined in section 2.3.1.23.

- The protocol server receives the request and parses the logic. Depending on the type of editors table request, the protocol server processes the subrequest as specified in section 3.1.4.8.1, 3.1.4.8.2, 3.1.4.8.3, 3.1.4.8.4 or 3.1.4.8.5. (The Response element is defined in section 2.2.3.5, and the SubResponse element is defined in section 2.2.3.10.) The protocol server uses the ClientID attribute sent in an editors table subrequest to do the following:
  - Uniquely identify each client
  - Keep track of each client and its timeout on its entry in the editors table
  - Optionally store information about the user, such as a user name or email address

The protocol server returns results based on the following conditions:

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

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

- If the ClientID does not currently exist in the editors table, the protocol server returns an error code value set to "EditorClientIdNotFound" for the Update editor metadata request.

- The protocol server returns an error code value set to "EditorMetadataQuotaReached" for an "Update editor metadata" request if the client has already exceeded its quota for key/value pairs. <54>

- The protocol server returns an error code value set to "EditorMetadataStringExceedsLengthLimit" for an "Update editor metadata" request if the key exceeds the server’s length limit.

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

- The protocol server returns an error code value set to "Success" to indicate success in processing the EditorsTable request.

To retrieve an editors table, a protocol client MUST send a QueryChanges request as specified in [MS-FSSHTTPP], with the PartitionID attribute set to the corresponding editors table partition on the server. <55>

### 3.1.4.8.1 Join Editing Session

If the EditorsTableRequestType attribute is set to "JoinEditingSession", the protocol server considers the editors table subrequest to be of type "Join editing session". The protocol server
processes this request to add an entry to the editors table associated with the coauthorable file by adding the client’s associated ClientID, Timeout, and AsEditor status in an entry.

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

3.1.4.8.2 Leave Editing Session

If the EditorsTableRequestType attribute is set to "LeaveEditingSession", the protocol server considers the editors table subrequest to be of type "Leave editing session." The protocol server processes this request to remove the entry in the editors table associated with the coauthorable file corresponding to the client with the given ClientID.

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

3.1.4.8.3 Refresh Editing Session

If the EditorsTableRequestType attribute is set to "RefreshEditingSession", the protocol server considers this editors table subrequest to be of type "Refresh editing session." The protocol server processes this request to refresh the Timeout value in the entry in the editors table associated with the coauthorable file corresponding to the client with the given ClientID.

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

3.1.4.8.4 Update Editor Metadata

If the EditorsTableRequestType attribute is set to "UpdateEditorMetadata", the protocol server considers this editors table subrequest to be of type "Update editor metadata." The protocol server processes this request to add or update the client-supplied key/value pair in the entry in the editors table associated with the coauthorable file corresponding to the client with the given ClientID.

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

3.1.4.8.5 Remove Editor Metadata

If the EditorsTableRequestType attribute is set to "RemoveEditorMetadata", the protocol server considers this editors table subrequest to be of type "Remove editor metadata." The protocol server processes this request to remove the client-supplied key/value pair for the given key in the entry in the editors table associated with the coauthorable file corresponding to the client with the given ClientID.

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

3.1.4.9 GetDocMetaInfo Subrequest

This operation is used to retrieve server metadata properties pertaining to the server file and its parent folder<56>. Valid metadata properties are as specified in [MS-FPSE] section 2.2.4.

The protocol client MUST send the GetDocMetaInfo SubRequest message only if the X-MSFSSHTTP header, as described in [MS-OCPROTO] section 2.1.2.1.2, has a value of 1.1 or greater. The protocol client sends a GetDocMetaInfo SubRequest message, which is of type GetDocMetaInfoSubRequestType as specified in section 2.3.1.26. The protocol server responds
with a `GetDocMetaInfo SubResponse` message, which is of type `GetDocMetaInfoSubResponseType` as specified in section 2.3.1.30. This is done as follows:

- The protocol client prepares a request containing a `URL` for the file, a unique request token, and one or more `SubRequest` elements, as defined in section 2.2.3.2 and section 2.2.3.8. The `SubRequest` element is of type "GetDocMetaInfo".

- The protocol server receives the request and parses the logic to request the metadata. The requested document and parent directory metadata are prepared as a response and sent back to the protocol client.

The `Response` element is defined in section 2.2.3.5, and the `SubResponse` element is defined in section 2.2.3.10. `GetDocMetaInfoSubResponseDataType` defines the type of the `SubResponseData` element inside the `GetDocMetaInfo SubResponse` element. `GetDocMetaInfoSubResponseDataType` is defined in section 2.3.1.27. The protocol server sends the requested metadata as child elements of the `GetDocMetaInfo SubResponseData` element. Document metadata and directory metadata are specified in the `DocProps` and the `FolderProps` element, respectively, both of which are of type `GetDocMetaInfoPropertySetType` as defined in section 2.3.1.28. The `GetDocMetaInfoPropertySetType` elements have one `Property` element per metadata item of type `GetDocMetaInfoPropertyType` as defined in section 2.3.1.29.

The protocol returns results based on the following conditions:

- If the processing of the `GetDocMetaInfo subrequest` by the protocol server failed to get the requested metadata or encountered an unknown exception, the protocol server returns an error code value set to "SubRequestFail".

- Otherwise, the protocol server sets the error code value to "Success" to indicate success in processing the `GetDocMetaInfo` subrequest.

### 3.1.4.10 GetVersions Subrequest

This operation is used to retrieve information about a file’s versions.<57>.

The protocol client MUST send the `GetVersions SubRequest` message only if the `X-MSFSSHTTP` header, as described in [MS-OCPROTO] section 2.1.2.1.2, has a value of 1.1 or greater. The protocol client sends a `GetVersions SubRequest` message, which is of type `GetVersionsSubRequestType` as specified in section 2.3.1.31. The protocol server responds with a `GetVersions SubResponse` message, which is of type `GetVersionsSubResponseType` as specified in section 2.3.1.32. This is done as follows:

- The protocol client prepares a request containing a `URL` for the file, a unique request token, and one or more `SubRequest` elements, as defined in section 2.2.3.2 and section 2.2.3.8. The `SubRequest` element is of type "GetVersions".

- The protocol server receives the request and parses the logic to request information about a file’s versions. The requested file version data is prepared as a response and sent back to the protocol client.

The `Response` element is defined in section 2.2.3.5, and the `SubResponse` element is defined in section 2.2.3.10. The `Results` element, as specified in [MS-VERSS] section 2.2.4.1, is a complex type that specifies information about the file’s versions.

The protocol returns results based on the following conditions:

- If the processing of the `GetVersions subrequest` by the protocol server failed to get the requested versions information or encountered an unknown exception, the protocol server returns an error code value set to "SubRequestFail".
Otherwise, the protocol server sets the error code value to "Success" to indicate success in processing the **GetVersions** subrequest.

### 3.1.4.11 Versioning Subrequest

This operation is used to request information on versions of a file or different types of versioning related operations on a file from the protocol server. Depending on the **VersioningRequestType** attribute value, the protocol server interprets the request as one of the following types of versioning operations:

- Get Version List
- Restore Version

The **VersioningRequestType** attribute is defined in section 2.3.3.9. The **SubRequestData** element for a versioning **subrequest** is of type **VersioningSubRequestDataType** and is defined in section 2.3.1.36.

The protocol client sends a versioning **SubRequest** message, which is of type **VersioningSubRequestType** as specified in section 2.3.1.36. The protocol server responds with a versioning **SubResponse** message, which is of type **VersioningSubResponseType** as specified in section 2.3.1.39. 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 "Versioning" and the **SubRequestData** element contains attributes that are input parameters used by the protocol server when processing the versioning subrequest. The **SubRequestData** element is of type **VersioningSubRequestDataType** and is defined in section 2.3.1.36.

- The protocol server receives the request, parses the logic, and depending on the type of versioning subrequest, processes the request as specified in section 3.1.4.11.1 or section 3.1.4.11.2.

- The **Response** element is defined in section 2.2.3.5, and the **SubResponse** element is defined in section 2.2.3.10. The **VersioningSubResponseDataType** defines the type of the **SubResponseData** element inside the versioning **SubResponse** element. The **VersioningSubResponseDataType** is defined in section 2.3.1.38.

- The protocol server returns results based on the following conditions:
  - Depending on the type of error, the **ErrorCode** is returned as an attribute of the **SubResponse** element. The **ErrorCode** attribute that is part of the **SubResponse** element is updated with a specific error code as specified in section 2.2.5.4.
  - If the protocol server was unable to find the URL for the file specified in the **Url** attribute, the protocol server reports a failure by returning an error code value set to "FileNotFoundExceptionOrCannotBeCreated" in the **ErrorCode** attribute sent back in the **SubResponse** element.
  - If the protocol server gets a versioning subrequest of type "Restore version" and the restore fails because the version number specifies a non-existent version, the protocol server returns an error code value set to "VersionNotFound".
  - An **ErrorCode** value of "Success" indicates success in processing the versioning request.

#### 3.1.4.11.1 Get Version List
If the `VersioningRequestType` attribute is set to "GetVersionList", the protocol server considers the versioning subrequest to be of type "Get Version List". The protocol server processes this request to return a list of the most recent versions of the file. The number of versions returned is decided by the protocol server.

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

### 3.1.4.11.2 Restore Version

If the `VersioningRequestType` attribute is set to "RestoreVersion", the protocol server considers the versioning subrequest to be of type 'Restore Version". The protocol server processes this request by restoring the file to its state in the version specified by the `Version` attribute.

If the `Version` attribute specifies a version that doesn’t exist, the protocol server returns an error status set to "VersionNotFound".

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

### 3.1.4.12 FileOperation Subrequest

This operation is used to make a request to the protocol server for a file operation.<ref>Depending on the `FileOperation` attribute value, the protocol server interprets the request as the following type of file operation:

- Rename

The `FileOperation` attribute is defined in section 2.3.3.8. The `SubRequestData` element for a file operation subrequest is of type `FileOperationSubRequestDataType` and is defined in section 2.3.1.33.

The protocol client sends a `FileOperation SubRequest` message, which is of type `FileOperationSubRequestType` as specified in section 2.3.1.34. The protocol server responds with a `FileOperation SubResponse` message, which is of type `FileOperationSubResponseType` as specified in section 2.3.1.35. 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 "FileOperation" and the `SubRequestData` element contains attributes that are input parameters used by the protocol server when processing the file operation subrequest. The `SubRequestData` element is of type `FileOperationSubRequestDataType` and is defined in section 2.3.1.33.

- The protocol server receives the request, parses the logic, and depending on the type of file operation subrequest, processes the file operation request.

The `Response` element is defined in section 2.2.3.5 and the `SubResponse` element is defined in section 2.2.3.10.

The protocol server returns results based on the following conditions:

- If the processing of the `FileOperation` subrequest by the protocol server failed to perform the operation or encountered an unknown exception, the protocol server returns an error code value set to "SubRequestFail".

- Otherwise, the protocol server sets the error code value to "Success" to indicate success in processing the `FileOperation` subrequest.
If the `FileOperation` attribute is set to "Rename", the protocol server considers the file operation subrequest to be of type "Rename". The protocol server processes this request to request a name change of a file on the server. The requested new name for the file is specified by the `NewFileName` attribute.

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

### 3.1.4.13 AmIAlone Subrequest

This operation is used to find out if the user is alone in a coauthoring session.\(^{<60>}\)

The protocol client sends an `AmIAlone SubRequest` message, which is of type `AmIAloneSubRequestType` as specified in section 2.3.1.46. The protocol server responds with an `AmIAlone SubResponse` message, which is of type `AmIAloneSubResponseType` as specified in section 2.3.1.48. 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 "AmIAlone".
- The protocol server receives the request and parses the logic to request the information about whether the user is alone. The requested information is prepared as a response and sent back to the protocol client.

The `Response` element is defined in section 2.2.3.5, and the `SubResponse` element is defined in section 2.2.3.10. The `AmIAloneSubResponseDataType` defines the type of the `SubResponseData` element inside the `AmIAloneSubResponse` element. The `AmIAloneSubResponseDataType` is defined in section 2.3.1.47. The protocol server sends the requested information as `AmIAlone SubResponseData` element. The `AmIAlone` attribute sent by the protocol server are defined in section 2.3.1.47.

The protocol returns results based on the following conditions:

- If the processing of the `AmIAlone` subrequest by the protocol server failed to get the requested versions information or encountered an unknown exception, the protocol server returns an error code value set to "SubRequestFail".

Otherwise, the protocol server sets the error code value to "Success" to indicate success in processing the `AmIAlone` subrequest.

### 3.1.4.14 LockStatus Subrequest

This operation is used to retrieve information about the lock state of a file.\(^{<61>}\)

The protocol client sends a `LockStatus SubRequest` message, which is of type `LockStatusSubRequestType` as specified in section 2.3.1.49. The protocol server responds with a `LockStatus SubResponse` message, which is of type `LockStatusSubResponseType` as specified in section 2.3.1.51. 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 "LockStatus".
- The protocol server receives the request and parses the logic to request the information about the lock status on a file. The requested information is prepared as a response and sent back to the protocol client.
The `Response` element is defined in section 2.2.3.5, and the `SubResponse` element is defined in section 2.2.3.10. The `LockStatusSubResponseDataType` defines the type of the `SubResponseData` element inside the `LockStatusSubResponse` element. The `LockStatusSubResponseDataType` is defined in section 2.3.1.50. The protocol server sends the requested information as attributes in the `LockStatusSubResponseData` element. The attributes sent by the protocol server are defined in section 2.3.1.50.

The protocol returns results based on the following conditions:

- If the processing of the `LockStatus subrequest` by the protocol server failed to get the requested information about lock status or encountered an unknown exception, the protocol server returns an error code value set to "SubRequestFail".

Otherwise, the protocol server sets the error code value to "Success" to indicate success in processing the `LockStatus` subrequest.

### 3.1.4.15 Properties Subrequest

This operation is used to request information on properties. Depending on the `Properties` attribute value, the protocol server interprets the request as one of the following types of Properties operations:

- Property Enumerate
- Property Get

The `Properties` attribute is defined in section 2.3.3.10. The `SubRequestData` element for a Properties subrequest is of type `PropertiesSubRequestDataType` and is defined in section 2.3.1.52.

The protocol client sends a Properties `SubRequest` message, which is of type `PropertiesSubRequestType` as specified in section 2.3.1.53. The protocol server responds with a Properties `SubResponse` message, which is of type `PropertiesSubResponseType` as specified in section 2.3.1.55. 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 "Properties" and the `SubRequestData` element contains attributes and elements that are input parameters used by the protocol server when processing the Properties subrequest. The `SubRequestData` element is of type `PropertiesSubRequestDataType` and is defined in section 2.3.1.52.

- The protocol server receives the request, parses the logic, and depending on the type of Properties subrequest, processes the request as specified in section 3.1.4.15.1 or section 3.1.4.15.2.

- The `Response` element is defined in section 2.2.3.5, and the `SubResponse` element is defined in section 2.2.3.10. The `PropertiesSubRequestDataType` defines the type of the `SubResponseData` element inside the versioning `SubResponse` element. The `PropertiesSubRequestDataType` is defined in section 2.3.1.52.

- The protocol server returns results based on the following conditions:
  - Depending on the type of error, the `ErrorCode` is returned as an attribute of the `SubResponse` element. The `ErrorCode` attribute that is part of the `SubResponse` element is updated with a specific error code as specified in section 2.2.5.4.
  - An `ErrorCode` value of "Success" indicates success in processing the Properties request.

### 3.1.4.15.1 Property Enumerate
If the `Properties` attribute is set to "PropertyEnumerate", the protocol server considers the `Properties subrequest` to be of type "Property Enumerate". The protocol server processes this request to return a list of Ids of the properties available for the specified resource.

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

### 3.1.4.15.2 Property Get

If the `Properties` attribute is set to "PropertyGet", the protocol server considers the `Properties subrequest` to be of type "Property Get". The protocol server processes this request to return a list of properties available for the specified property Ids. The properties to be retrieved is specified by the `PropertyIds` element.

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

### 3.1.5 Timer Events

None.

### 3.1.6 Other Local Events

None.
4 Protocol Examples
This section provides common scenarios involving the use of the different types of subrequest operations for synchronizing a file's contents or metadata contents.

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

4.1 Successful File Open of a Coauthorable Document
A client wants to open a file on a protocol server. This file is a coauthorable document. The client successfully opens the coauthorable document by sending a series of subrequests to initiate a download of the file contents for shared editing.

The protocol server is named Example.

The source file to be opened is http://Example/shared%20documents/test1.docx.

4.1.1 Request
<Envelope xmlns="http://schemas.xmlsoap.org/soap/envelope/">
  <Body>
    <RequestVersion Version="2" MinorVersion="0"
                     xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <RequestCollection CorrelationId="{A2FFBFA0-50BA-47EC-81CB-D562A74A58768}" 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-ED034E7B73C" 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>
        </RequestCollection>
      </Request>
    </Body>
  </Envelope>
The protocol client sends seven **SubRequest** elements as part of the **Request** element in the cell storage service request message for opening the file contents. The file whose contents need to be opened is uniquely identified by the **URL** for the file. The URL for the file is specified as part of the **Url** attribute of the **Request** element, as specified in section 2.2.3.2. Each **SubRequest** element specifies a type of **subrequest** to the protocol server and is uniquely identified by the **SubRequestToken** attribute. Details about the **SubRequestToken** attribute are specified in section 2.2.4.5. The type of the subrequest is specified as part of the **Type** attribute for each **SubRequest** element, as specified in section 2.2.4.4.

The first **SubRequest** element is a coauthoring subrequest of type "Join coauthoring session" that requests a shared lock on the file and the coauthoring status. Details about the coauthoring subrequest of type "Join coauthoring session" are specified in section 3.1.4.3.1.

The second **SubRequest** element is a schema lock subrequest of type "Get lock" that also requests a shared lock on the file. Unlike the first **SubRequest** element, the coauthoring subrequest, the schema lock subrequest does not get the coauthoring status. This schema lock subrequest is executed only if the first **SubRequest** element, the coauthoring subrequest, is not supported by the protocol server. The dependency of the schema lock subrequest on the first **SubRequest** element is defined by the **DependsOn** attribute and the **DependencyType** attribute of the schema lock subrequest. In this case, the **DependsOn** value of 1 specifies the **SubRequestToken** of the first **SubRequest** element, which the second element is dependent on. The **DependencyType** value of "OnNotSupported" specifies that the second schema lock subrequest gets called only if the first **SubRequest** element is not supported. Details about the **DependsOn** and **DependencyType** attributes are specified in section 2.2.4.5. Details about the schema lock subrequest of type "Get lock" are specified in section 3.1.4.1.

The third, fourth, and fifth **SubRequest** elements are cell subrequests that request the download of file contents or file metadata contents. All three cell subrequests are dependent on the second **SubRequest** element, as defined by the **DependsOn** attribute with a value of 2, which specifies the **SubRequestToken** of the second **SubRequest** element, and the **DependencyType** attribute with a value of "OnExecute", which specifies that the cell subrequest gets executed only after the second **SubRequest** element is executed.

The third and fifth **SubRequest** elements specify the **PartitionID** attribute, while the fourth **SubRequest** element specifies the **GetFileProps** attribute. The **PartitionID** attribute specifies the **Partition-block** subfile contents or file metadata contents that are downloaded. The binary data sent in the **SubRequestData** element of a cell subrequest indicates if this is a file content upload or download request. The cell subrequest for a download of file contents is processed as described in [MS-FSSHTTP] section 3.1.4.2. In the fourth **SubRequest** element, the **GetFileProps** attribute is set to "true" to request the properties of the file. Details about the **PartitionID** attribute and **GetFileProps** attribute are specified in section 2.3.3.1. Details about the cell subrequest are specified in section 3.1.4.2.

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

The seventh **SubRequest** element is a **WhoAmI** subrequest that requests the current client user information. This client user information helps in showing a client's friendly name when a coauthorable file is edited by more than one client. Details about the **WhoAmI** subrequest are specified in section 3.1.4.6.

In **SubRequest** elements where the **SchemaLockID** attribute string is present, the string is "29358EC1-E813-4793-8E70-ED0344E7B73C".

### 4.1.2 Response

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
```

[MS-FSSHTTP] - v20190924
File Synchronization via SOAP over HTTP Protocol
Copyright © 2019 Microsoft Corporation
Release: September 24, 2019
For each `SubRequest` element that is in a `Request` element of a cell storage service request message, the protocol server sends a corresponding `SubResponse` element in a `Response` element as part of the cell storage service response message. Each `SubResponse` element contains the `SubRequestToken` attribute and the `ErrorCode` attribute. The `SubRequestToken` attribute identifies the `SubRequest` element for which the `SubResponse` was generated by the protocol server. The `ErrorCode` attribute specifies the error code result for the specific `subrequest`. The `SubRequestToken` attribute and the `ErrorCode` attribute that are part of a `SubResponse` element are specified in section 2.2.4.8.
The **SubResponse** element with a **SubRequestToken** set to 1 in this example is the response from the protocol server for the coauthoring subrequest of type "Join coauthoring session". The **ErrorCode** attribute of "Success" indicates that the coauthoring subrequest was successfully processed, as specified in section 2.2.5.6. The **SubResponseData** element of the first **SubResponse** element specifies the type of lock granted in the **LockType** attribute and the coauthoring status in the **CoauthStatus** attribute. In this **SubResponse** element, the **LockType** attribute of "SchemaLock" indicates a shared lock on the file, and the **CoauthStatus** attribute of "Alone" indicates that only the current client is in the coauthoring session and editing the file. Details about the **LockType** attribute and **CoauthStatus** attribute are specified in section 2.3.1.7.

The **SubResponse** element with a **SubRequestToken** set to 2 is the response for the schema lock subrequest of type "Get Lock". The **ErrorCode** attribute of "DependentOnlyOnNotSupportedRequestGetSupported", defined in section 2.2.5.2, indicates that the coauthoring subrequest on which this schema lock subrequest is dependent was supported by the protocol server, so the schema lock subrequest was not executed.

The **SubResponse** elements with **SubRequestToken** values of 3, 4, and 6 are the responses from the protocol server for the corresponding cell subrequests. The **ErrorCode** attributes in each **SubResponse** element of "Success" indicate that all three cell subrequests were successfully processed. The third and fifth **SubResponse** elements contain the requested file contents or file metadata contents. The fourth **SubResponse** element contains the file properties requested as part of the corresponding cell subrequest. The **Etag** attribute in the fourth **SubResponse** element specifies the file version so that the protocol client knows which version of the file contents it is receiving. Details about the **Etag** attribute are specified in section 2.3.3.2.

The format of the binary data values of the **SubResponseData** elements is described in [MS-FSSHTTPB] section 2.2.3.1.3.

The **SubResponse** element with a **SubRequestToken** set to 5 is the response for the **ServerTime** subrequest. The **ErrorCode** attribute of "Success" indicates that the **ServerTime** subrequest was successfully processed. The **SubResponseData** element specifies the server time with the **ServerTime** attribute, as specified in section 2.3.1.18.

The seventh **SubResponse** element is the response for the **WhoAmI** subrequest. The **ErrorCode** attribute of "Success" indicates that the **WhoAmI** subrequest was successfully processed. The **SubResponseData** element specifies the requested client specific information in the **UserName**, **UserLogin**, **UserEmailAddress**, and **UserSIPAddress** attributes, which are specified in section 2.3.3.6.

### 4.2 Successful File Save of a Coauthorable Document

A client wants to save a file that has been edited back to the protocol server. This file is a coauthorable document. The client successfully saves the coauthorable document by sending a series of **subrequests** to initiate an upload of the file contents.

The protocol server is named Example.

The source file to be saved is http://Example/shared%20documents/test1.docx.

#### 4.2.1 Request

```xml
<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">
```
The protocol client sends three SubRequest elements as part of the Request element in the cell storage service request message for uploading the file contents.

The first SubRequest element is a coauthoring subrequest of type "Refresh coauthoring session", which requests a refresh of the client’s timeout of the shared lock and an update of the coauthoring status. Details about the coauthoring subrequest of type "Refresh coauthoring session" are specified in section 3.1.4.3.3.

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

The third SubRequest element is a cell subrequest that requests the upload of file contents or file metadata contents. Because the DependencyType attribute for this subrequest is set to "OnSuccessOrNotSupported", it is executed if either the coauthoring subrequest or the schema lock subrequest was successfully executed or if both these subrequests are not supported by the server. The Coalesce attribute of "true" requests that the protocol server persist all changes to the file with the underlying store. The CoauthVersioning attribute of "true" requests that the protocol server optimize the versioning of the file for coauthoring. Details about the Coalesce attribute and CoauthVersioning attribute are specified in section 2.3.3.1.

The Include element within the SubRequestData element of the cell subrequest is used for encapsulating and sending large amounts of binary data. Details about the Include element are specified in section 2.2.3.1. The cell subrequest for upload of file contents is processed as described in [MS-FSSHTTPB] section 3.1.4.3. Details about the cell subrequest are specified in section 3.1.4.2.

4.2.2 Response

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
      <Response RequestToken="1" HealthScore="0"><SubResponse SubRequestToken="1" ErrorCode="Success" HResult="0"></SubResponse>
    </ResponseCollection>
  </s:Body>
</s:Envelope>
The first SubResponse element in this example is the response from the protocol server to the first subrequest from the protocol client, which is of type "Refresh coauthoring session". The ErrorCode attribute of "Success" indicates that the coauthoring subrequest was successfully processed, as specified in section 2.2.5.6. The SubResponseData element of the first SubResponse element specifies that the type of lock that was refreshed in the LockType attribute is "SchemaLock" and that the current coauthoring status in the CoauthStatus attribute is "Alone".

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

The third SubResponse element is the response from the protocol server to the cell subrequest. The ErrorCode attribute of "Success" indicates that the cell subrequest for the uploading of file contents or file metadata contents was successfully processed. The other attributes and elements of this SubResponse element are described in section 4.2.1. The format of the value of the SubResponseData element is as described in [MS-FSSHTTPB] section 2.2.3.1.3.

4.3 Successful File Open of a Document that Is Not Coauthorable

A client wants to open a file on a protocol server. This file is not a coauthorable document. The client successfully opens the document by sending two subrequests to initiate a download of the file contents for exclusive editing.

The protocol server is named Example.

The source file to be opened is http://Example/shared%20documents/test2.xlsx.

4.3.1 Request

<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <RequestVersion Version="2" MinorVersion="0" xmlns="http://schemas.microsoft.com/sharepoint/soap/"/>
    <RequestCollection CorrelationId="{E07FCF1D-AD34-403F-9F77-D31B8225C8C5}" xmlns="http://schemas.microsoft.com/sharepoint/soap/">
      <Request Url="http://Example/Shared%20Documents/test2.xlsx" RequestToken="1">"/>
      <SubRequestType Type="ExclusiveLock" SubRequestToken="1">
        <SubRequestData ExclusiveLockRequestType="GetLock" ExclusiveLockID="{9BCE3023-0F1F-496B-A561-610144854040}" Timeout="3600"/>
      </SubRequest>
      <SubRequest Type="Cell" SubRequestToken="2" DependsOn="1" DependencyType="OnExecute">"/>
    </RequestCollection>
  </s:Body>
</s:Envelope>
The protocol client sends two **SubRequest** elements as part of the **Request** element in the cell storage service request message for opening the file contents.

The first **SubRequest** element is an exclusive lock **subrequest** of type "Get lock" that requests an exclusive lock on the file. The **ExclusiveLockID** attribute in the **SubRequestData** element specifies a unique identifier for the exclusive lock on the file. Details about the **ExclusiveLockID** attribute are specified in section 2.3.1.9. Details about the exclusive lock subrequest of type "Get lock" are specified in section 3.1.4.5.1.

The second **SubRequest** element is a cell subrequest that requests the download of file contents or file metadata contents and file properties. Details about the cell subrequest are specified in section 3.1.4.2.

### 4.3.2 Response

The first **SubResponse** element in this example is the response from the protocol server to the exclusive lock **subrequest** of type "Get lock". The **ErrorCode** attribute of "Success" indicates that the coauthoring subrequest was successfully processed, as specified in section 2.2.5.6.

The second **SubResponse** element is the response from the protocol server for the cell subrequest. The **ErrorCode** of "Success" indicates that the cell subrequest for the downloading of file contents or file metadata contents and file properties was successfully processed. The **Include** element within the **SubResponseData** element of the response to the cell subrequest is used for encapsulating and sending large amounts of binary data. Details about the **Include** element are specified in section 2.2.3.1.
4.4 Unsuccessful File Open of a Document that Is Not Coauthorable

A client wants to open a file on the protocol server. This file is a not a coauthorable document. The client sends two subrequests to initiate a download of the file contents for exclusive editing. The client fails to open the document for exclusive editing because the file is already exclusively locked by another client.

The protocol server is named Example.

The source file to be opened is http://Example/shared%20documents/test2.xlsx.

4.4.1 Request

```xml
<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-CF31FDA0F0A}" 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+9FTDnCCAD0J0fdwENAgYA AwUAigIACAADqAgYAAAYaAYgIAAgAgAOEACYCId2NkoYQcURJaGemzypNa5pBA4JET1xqOXsRdOgqho/p1URQaAcZku wo5XEwy6lmqqGJ+8VRGqC5UMPJ+gIgkArkCcCrcPyAmiZLM/k0whlsAXatDFkLQT4v0ZlEpeMnIy?cpxEjMwAALUTAS YCIAAO6XY6MoAMTvn88ZKU+M+aEgJgqX9caja80XRqqaop6dVEaprwC1EwFCWGeAgBAVEe=E="/SubRequestData>
        </SubRequest>
      </Request>
    </RequestCollection>
  </s:Body>
</s:Envelope>
```

The protocol client sends two SubRequest elements as part of the Request element in the cell storage service request message for opening the file contents.

The first SubRequest element is an exclusive lock subrequest of type "Get lock" that requests an exclusive lock on the file and provides an ExclusiveLockID of "{9BCE3023-0F1F-496B-A561-610144B54040}". Details about the exclusive lock subrequest of type "Get lock" are specified in section 3.1.4.5.1.

The second SubRequest element is a cell subrequest that requests the download of file contents or file metadata contents and file properties. Details about the cell subrequest are specified in section 3.1.4.2.

4.4.2 Response

```xml
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
      <Response Url="http://Example/Shared%20Documents/test2.xlsx" UrlIsEncoded="true" RequestToken="1" HealthScore="0">
        <SubResponse SubRequestToken="1" ErrorCode="FileAlreadyLockedOnServer" ErrorMessage="EXAMPLE\jdarcy" HRESULT="2147500037">
          <SubResponseData/>
        </SubResponse>
      </Response>
    </ResponseCollection>
  </s:Body>
</s:Envelope>
```
The first **SubResponse** element in this example is the response from the protocol server to the exclusive lock **subrequest** of type "Get lock". The **ErrorCode** attribute of "FileAlreadyLockedOnServer", defined in section 2.2.5.8, indicates that there is an existing exclusive lock on the targeted file or an existing schema lock on the targeted file but with a schema lock identifier different from the one provided in the client request. The **ErrorMessage** attribute of "EXAMPLE\jdarcy" specifies the identity of the user who is currently holding the lock on the file. The **HResult** attribute set to 2,147,500,037 specifies an error code specific to the exclusive lock subrequest that failed and gives more hints about the cause of the failure. Details about the **ErrorMessage** attribute and **HResult** attribute are specified in section 2.2.4.8.

Although the first subrequest failed, the second subrequest is executed by the server because the **DependencyType** attribute in the second subrequest is set to "OnExecute". The second **SubResponse** element is the response from the protocol server for the cell subrequest. The **ErrorCode** of "Success" indicates that the cell subrequest for the downloading of file contents and file properties was successfully processed. In this example, because the exclusive lock subrequest of type "Get lock" failed but the cell subrequest for download succeeded, the protocol client is allowed to open the file contents in read-only mode. The protocol client will not be allowed to make edits on the file.

### 4.5 Successful File Save of a Document that Is Not Coauthorable

A client wants to save a file that it has edited back to the protocol server. This file is a not a coauthorable document. The client successfully saves the document by sending two **subrequests** to initiate an upload of the file contents.

The protocol server is named Example.

The source file to be saved is http://Example/shared%20documents/test2.xlsx.

#### 4.5.1 Request

```xml
<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-496A-A561-610144B54040}" Timeout="3600"/>
        </SubRequest>
        <SubRequest Type="Cell" SubRequestToken="2" DependsOn="1" DependencyType="OnSuccessOrNotSupported">
          <SubRequestData Coalesce="true" CoauthVersioning="true" BypassLockID="{9BCE3023-0F1F-496A-A561-610144B54040}" BinaryDataSize="14972"/>
        </SubRequest>
      </Request>
    </RequestCollection>
  </s:Body>
</s:Envelope>
```
The protocol client sends two SubRequest elements as part of the Request element in the cell storage service request message for uploading the file contents.

The first SubRequest element is an exclusive lock subrequest of type "Refresh lock" that requests a refresh of the client’s timeout of the exclusive lock on the file. Details about the exclusive lock subrequest of type "Refresh lock" are specified in section 3.1.4.5.3.

The second SubRequest element is a cell subrequest that requests the upload of file contents or file metadata contents. This cell subrequest is executed only if the first SubRequest element, the exclusive lock subrequest, succeeded or is not supported by the protocol server. The dependency of the cell subrequest on the first SubRequest element is defined by the DependsOn attribute and the DependencyType attribute of the cell subrequest. In this case, the DependsOn value of 1 specifies the SubRequestToken of the first SubRequest element, which the second element is dependent on. The DependencyType value of "OnSuccessOrNotSupported" specifies that the second cell subrequest gets called only if the first SubRequest element succeeded or is not supported, but not if it failed for a different reason. Details about the cell subrequest are specified in section 3.1.4.2.

4.5.2 Response

The first SubResponse element in this example is the response from the protocol server to the exclusive lock subrequest of type "Refresh lock". The ErrorCode attribute of "Success" indicates that the exclusive lock subrequest was successfully processed, as specified in section 2.2.5.6.

The second SubResponse element is the response from the protocol server for the cell subrequest. The ErrorCode attribute of "Success" indicates that the cell subrequest for the uploading of file
contents or file metadata contents was successfully processed. Other attributes and elements of this SubResponse are described in previous examples.

### 4.6 Unsuccessful File Open of a Coauthorable Document

A client wants to open a coauthorable document for coauthoring from a protocol server. The client fails to open the document for coauthoring because the number of coauthors has already reached the maximum. This section is almost identical to section 4.1, with the exception that the first subrequest ("Coauth") fails because the maximum number of coauthors has been reached.

The protocol server is named Example.

The source file to be opened is http://Example/Shared%20Documents/hello.docx.

#### 4.6.1 Request

```xml
<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-D024-42C2-9A22-46BADD853FD1}" xmlns="http://schemas.microsoft.com/sharepoint/soap/">
        <Request Url="http://Example/Shared%20Documents/hello.docx" RequestToken="1">
          <SubRequest Type="Coauth" SubRequestToken="1">
          </SubRequest>
          <SubRequest Type="SchemaLock" SubRequestToken="3" DependsOn="1" DependencyType="OnNotSupported">
          </SubRequest>
          <SubRequest Type="Cell" SubRequestToken="6" DependsOn="3" DependencyType="OnExecute">
            <SubRequestData GetFileProps="true" BinaryDataSize="88">DAALAJzPKfM5lAabBgIAAO4CAACqAiAAfrgx50XdqkSrgAx1+9FTDn0CCB000UudEwAgYAAwUAigICAADaAgYAAwAAygIIAAgAgAOEAEELAawCAFUDQ==</SubRequestData>
          </SubRequest>
          <SubRequest Type="Cell" SubRequestToken="5" DependsOn="3" DependencyType="OnExecute">
            <SubRequestData PartitionID="383ADC0B-e66e-4438-95e6-e39f9720120" BinaryDataSize="88">DAALAJzK8M51AabBgIAAO4CAACqA1Afgrx50XdqkSrgAx1+9FTDn0CCB000UudEwAgYAAwUAigICAADaAgYAAwAAygIIAAgAgAOEAEELAawCAFUDQ==</SubRequestData>
          </SubRequest>
          <SubRequest Type="Cell" SubRequestToken="4" DependsOn="3" DependencyType="OnExecute">
            <SubRequestData PartitionID="383ADC0B-e66e-4438-95e6-e39f9720120" BinaryDataSize="88">DAALAJzK8M51AabBgIAAO4CAACqA1Afgrx50XdqkSrgAx1+9FTDn0CCB000UudEwAgYAAwUAigICAADaAgYAAwAAygIIAAgAgAOEAEELAawCAFUDQ==</SubRequestData>
          </SubRequest>
          <SubRequest Type="WhoAmI" SubRequestToken="2"/>
          <SubRequest Type="ServerTime" SubRequestToken="7"/>
        </RequestCollection>
      </Request>
    </s:Body>
  </s:Envelope>
```

The protocol client sends seven SubRequest elements as part of the Request element in the cell storage service request message for opening the document. The seven subrequests are identical to those in section 4.1.1.
4.6.2 Response

```
<s:Envelope xmlns:s="http://schemas.xmlsoap.org/soap/envelope/">
  <s:Body>
    <ResponseCollection WebUrl="http://Example" WebUrlIsEncoded="false">
      <Response RequestToken="1" HealthScore="0" RequestUrl="http://Example/Shared%20Documents/hello.docx" UrlIsEncoded="true">91
        <SubResponse SubRequestToken="1" ErrorCode="NumberOfCoauthorsReachedMax" HResult="2147500037"/>
      91
        <SubResponse SubRequestToken="3" ErrorCode="DependentOnlyOnNotSupportedRequestGetSupported" HResult="2147500037">
          <SubResponseData/>
          1d4
        </SubResponse>
        <SubResponse SubRequestToken="6" ErrorCode="Success" HResult="0">
          <SubResponseData Etag="" CoalesceHResult="0" ContainsHotboxData="False" HaveOnlyDemotionChanges="False" LastModifiedTime="1292171159700000000" ModifiedBy="User" xmlns:xop="http://www.w3.org/2004/08/xop/include">f9</SubResponseData>
          13d
        </SubResponse>
        <SubResponse SubRequestToken="5" ErrorCode="Success" HResult="0">
          <SubResponseData CoalesceHResult="0" ContainsHotboxData="True" HaveOnlyDemotionChanges="False">DAALAJ3PKfM5lAabFgMCACsAgMVwiNmq4KENGARZbd2YMxEO2gNV311VftJZJo4z7N5gOMcBAABAAAIA0INCAYc8l3OCE9ED71CoJvYCAEAAAAAAAABVUOAyAaWOA+gIkaAW1Nmq4KE NAR2zfZIMxEO2aA1QAQQcBwE="</SubResponseData>
          170
        </SubResponse>
        <SubResponse SubRequestToken="4" ErrorCode="Success" HResult="0">
          <SubResponseData CoalesceHResult="0" ContainsHotboxData="True" HaveOnlyDemotionChanges="False">DAALAJ3PKfM51AabFgMCACsAgMVwiNmq4KENGARZbd2YMxEO2gNV311VftJZJo4z7N5gOMcBAABAAAIA0INCAYc8l3OCE9ED71CoJvYCAEAAAAAAAABVUOAyAaWOA+gIkaAW1Nmq4KE NAR2zfZIMxEO2aA1QAQQcBwE="</SubResponseData>
          91
        </SubResponse>
        <SubResponse SubRequestToken="2" ErrorCode="Success" HResult="0" UserLogin="DOMAIN\User" UserEmailAddress="user@example.com" UserName="The Automation Net Client" UserSIFAddress="user@example.com"/>
          81
        </SubResponse>
        <SubResponse SubRequestToken="7" ErrorCode="Success" HResult="0">
          <SubResponseData ServerTime="6341287002500000000"/>
          36
        </SubResponse>
      </ResponseCollection>
    </s:Body>
  </s:Envelope>
```

With one exception, the **SubResponse** elements in the **Response** element are identical to those of section 4.1.2. The one exception is the **SubResponse** element for the "Coauth" subrequest. It shows that the "Coauth" subrequest failed with an **ErrorCode** of "NumberOfCoauthorsReachedMax", defined in section 2.2.5.8, indicating that no more coauthors are allowed because the maximum number of coauthors has already been reached on the server.
5 Security

5.1 Security Considerations for Implementers
None.

5.2 Index of Security Parameters
None.
6 Appendix A: Full XML Schema

For ease of implementation, the following sections provide the full XML schemas for this protocol.

<table>
<thead>
<tr>
<th>Schema name</th>
<th>Prefix</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request message</td>
<td>None</td>
<td>6.1</td>
</tr>
<tr>
<td>Response message</td>
<td>None</td>
<td>6.2</td>
</tr>
</tbody>
</table>

6.1 Request Message Schema

```xml
<?xml version="1.0" encoding="utf-8"?>
 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
<?xml version="1.0" encoding="utf-8"?>
 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>
```
<xs:complexType name="EditorsTableSubRequestDataType" mixed="true">
  <xs:attributeGroup ref="tns:EditorsTableSubRequestDataOptionalAttributes" />
  <xs:attribute name="ClientID" type="xs:string" use="optional" />
  <xs:attribute name="AsEditor" type="xs:boolean" use="optional" />
  <xs:attribute name="Timeout" type="xs:integer" use="optional" />
  <xs:attribute name="Key" type="xs:string" use="optional" />
  <xs:attribute name="Value" type="xs:base64Binary" use="optional" />
</xs:complexType>

<xs:complexType name="FileOperationSubRequestDataType">
  <xs:attributeGroup ref="tns:FileOperationSubRequestDataOptionalAttributes" />
  <xs:attribute name="NewFileName" type="xs:string" use="optional" />
  <xs:attribute name="ExclusiveLockID" type="xs:string" use="optional" />
</xs:complexType>

<xs:complexType name="VersioningSubRequestDataType">
  <xs:attributeGroup ref="tns:VersioningSubRequestDataOptionalAttributes" />
  <xs:attribute name="Version" type="tns:FileVersionNumberType" use="optional" />
</xs:complexType>

<xs:complexType name="AmIAloneSubRequestDataType">
  <xs:attribute name="TransitionID" type="tns:guid" use="optional" />
</xs:complexType>

<xs:complexType name="PropertiesSubRequestDataType">
  <xs:sequence>
    <xs:element name="PropertyIds" minOccurs="0" maxOccurs="1" type="tns:PropertyIdsType" />
  </xs:sequence>
  <xs:attributeGroup ref="tns:PropertiesSubRequestDataOptionalAttributes" />
</xs:complexType>

<xs:complexType name="PropertyIdsType">
  <xs:sequence>
    <xs:element name="PropertyId" minOccurs="0" maxOccurs="unbounded" type="tns:PropertyIdType" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="PropertyIdType">
  <xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="SubRequestDataGenericType" mixed="true">
  <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="Timeout" type="xs:integer" use="optional" />
  <xs:attribute name="ExclusiveLockID" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="EditorsTableSubRequestDataType" mixed="true">
  <xs:attribute name="ClientID" type="xs:string" use="required" />
  <xs:attribute name="AsEditor" type="xs:boolean" use="optional" />
  <xs:attribute name="Timeout" type="xs:integer" use="optional" />
  <xs:attribute name="Key" type="xs:string" use="optional" />
  <xs:attribute name="Value" type="xs:base64Binary" use="optional" />
</xs:complexType>

<xs:complexType name="FileOperationSubRequestDataType">
  <xs:attribute name="NewFileName" type="xs:string" use="optional" />
  <xs:attribute name="ExclusiveLockID" type="xs:string" use="optional" />
</xs:complexType>

<xs:complexType name="VersioningSubRequestDataType">
  <xs:attribute name="Version" type="tns:FileVersionNumberType" use="optional" />
</xs:complexType>

<xs:complexType name="AmIAloneSubRequestDataType">
  <xs:attribute name="TransitionID" type="tns:guid" use="optional" />
</xs:complexType>

<xs:complexType name="PropertiesSubRequestDataType">
  <xs:sequence>
    <xs:element name="PropertyIds" minOccurs="0" maxOccurs="1" type="tns:PropertyIdsType" />
  </xs:sequence>
  <xs:attributeGroup ref="tns:PropertiesSubRequestDataOptionalAttributes" />
</xs:complexType>

<xs:complexType name="PropertyIdsType">
  <xs:sequence>
    <xs:element name="PropertyId" minOccurs="0" maxOccurs="unbounded" type="tns:PropertyIdType" />
  </xs:sequence>
</xs:complexType>

<xs:complexType name="PropertyIdType">
  <xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="SubRequestDataGenericType" mixed="true">
  <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:element ref="i:Include" minOccurs="0" maxOccurs="1" />
<xs:element name="PropertyIds" minOccurs="0" maxOccurs="1" type="tns:PropertyIdsType"/>
</xs:all>
<xs:attributeGroup ref="tns:SubRequestDataOptionalAttributes"/>
</xs:complexType>

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

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

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

<xs:complexType name="CellSubRequestType">
  <xs:complexContent>
    <xs:extension base="tns:SubRequestType">
      <xs:sequence minOccurs="0" maxOccurs="1">
        <xs:element name="SubRequestData" type="tns:CellSubRequestDataType" />
      </xs:sequence>
      <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required" fixed="Cell" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

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

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

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

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

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

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

<xs:complexType name="FileOperationSubRequestType">
  <xs:complexContent>
    <xs:extension base="tns:SubRequestType">
      <xs:sequence minOccurs="1" maxOccurs="1">
        <xs:element name="SubRequestData" type="tns:FileOperationSubRequestDataType" />
      </xs:sequence>
      <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required" fixed="FileOperation" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="VersioningSubRequestType">
  <xs:complexContent>
    <xs:extension base="tns:SubRequestType">
      <xs:sequence minOccurs="1" maxOccurs="1">
        <xs:element name="SubRequestData" type="tns:VersioningSubRequestDataType" />
      </xs:sequence>
      <xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required" fixed="Versioning" />
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

<xs:complexType name="AmIAloneSubRequestType">
  <xs:complexContent>
    <xs:extension base="tns:SubRequestType">
      <xs:sequence minOccurs="1" maxOccurs="1">
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>


<xs:element name="SubRequestData" type="tns:AmIAloneSubRequestDataType" />
</xs:sequence>
<xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="AmIAlone" />
</xs:extension>
</xs:complexType>

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

<xs:complexType name="PropertiesSubRequestType">
<xs:complexContent>
<xs:extension base="tns:SubRequestType">
<xs:sequence minOccurs="1" maxOccurs="1">
<xs:element name="SubRequestData" type="tns:PropertiesSubRequestDataType" />
</xs:sequence>
<xs:attribute name="Type" type="tns:SubRequestAttributeType" use="required"
fixed="Properties" />
</xs:extension>
</xs:complexContent>
</xs:complexType>

<!--[1--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>

<!--[1--definition of simple elements -->]
<!--[1--definition of complex elements -->]
<xs:element name="RequestVersion" type="tns:VersionType" />
<xs:element name="Request">
<xs:complexType>
<xs:sequence>
<xs:element name="GenericProperties" type="tns:GenericPropertiesType" minOccurs="0" />
<xs:sequence minOccurs="1" maxOccurs="unbounded">
<xs:element name="SubRequest" type="tns:SubRequestElementGenericType" />
</xs:sequence>
</xs:sequence>
<xs:attribute name="Url" type="xs:string" use="required" />
<xs:attribute name="Interval" type="xs:nonNegativeInteger" use="optional" />
<xs:attribute name="MetaData" type="xs:integer" use="optional" />
<xs:attribute name="RequestToken" type="xs:nonNegativeInteger" use="required" />
<xs:attribute name="UserAgent" type="tns:guid" use="optional" />
<xs:attribute name="UserAgentClient" type="xs:string" use="optional" />
<xs:attribute name="UserAgentPlatform" type="xs:string" use="optional" />
<xs:attribute name="Build" type="xs:string" use="optional" />
</xs:complexType>
</xs:element>
6.2 Response Message Schema

The referenced child elements of the Body element are specified in the following schema:

```xml
<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>
```
<xs:simpleType name="LockTypes">
<xs:union>
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:enumeration value="None"/>
<xs:enumeration value="SchemaLock"/>
<xs:enumeration value="ExclusiveLock"/>
</xs:restriction>
</xs:simpleType>
<xs:simpleType>
<xs:restriction base="xs:integer">
<xs:enumeration value="0"/>
<xs:enumeration value="1"/>
<xs:enumeration value="2"/>
</xs:restriction>
</xs:simpleType>
</xs:union>
</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="3"/>
</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:union memberTypes="tns:GenericErrorCodeTypes
</xs:simpleType>

<xs:simpleType name="GenericErrorCodeTypes">
<xs:restriction base="xs:string">
<xs:enumeration value="Success"/>
<xs:enumeration value="IncompatibleVersion"/>
<xs:enumeration value="InvalidUrl"/>
</xs:restriction>
</xs:simpleType>
<xs:enumeration value="Coauthoring"/>
</xs:restriction>
</xs:simpleType>

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

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

<xs:simpleType name="FileVersionNumberType">
<xs:restriction base="xs:string">
<xs:pattern value="[0-9]+.[0-9]+" />
</xs:restriction>
</xs:simpleType>

<xs:simpleType name="VersioningRelatedErrorCodeTypes">
<xs:restriction base="xs:string">
<xs:enumeration value="VersionNotFound"/>
</xs:restriction>
</xs:simpleType>

<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:attribute name="AmIAlone" type="xs:boolean" use="optional"/>
<xs:attribute name="LockedBy" type="xs:string" use="optional"/>
</xs:attributeGroup>

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

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

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

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

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

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

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

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

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

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

<xs:complexType name="VersioningSubResponseDataType">
    <xs:sequence minOccurs="0" maxOccurs="1">
        <xs:sequence minOccurs="0" maxOccurs="1">
            <xs:element name="UserTable" type="tns:VersioningUserTableType"/>
        </xs:sequence>
        <xs:element name="Versions" type="tns:VersioningVersionListType"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="VersioningUserTableType">
    <xs:sequence>
        <xs:element name="User" maxOccurs="unbounded" minOccurs="1" type="tns:UserDataType"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="VersioningVersionListType">
    <xs:sequence>
        <xs:element name="Version" maxOccurs="unbounded" minOccurs="1" type="tns:FileVersionDataType"/>
    </xs:sequence>
</xs:complexType>

<xs:complexType name="UserDataType"/>
<xs:attribute name="UserId" type="xs:integer" use="required" />
<xs:attribute name="UserLogin" type="xs:UserLoginType" use="required" />
<xs:attribute name="UserName" type="xs:UserNameType" use="optional" />
<xs:attribute name="UserEmailAddress" type="xs:string" use="optional" />
</xs:complexType>

<xs:complexType name="FileVersionDataType">
<xs:sequence>
<xs:element name="Events" minOccurs="0" maxOccurs="1" >
<xs:complexType>
<xs:sequence>
<xs:element name="Event" minOccurs="1" maxOccurs="unbounded" type="tns:FileVersionEventDataType" />
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>

<xs:complexType name="FileVersionEventDataType">
<xs:attribute name="Id" type="xs:integer" use="required" />
<xs:attribute name="Type" type="xs:integer" use="required" />
<xs:attribute name="CreateTime" type="xs:positiveInteger" use="optional" />
<xs:attribute name="UserId" type="xs:integer" use="optional" />
</xs:complexType>

<xs:complexType name="AmIAloneSubResponseDataType">
<xs:attribute name="AmIAlone" type="xs:boolean" use="optional" />
</xs:complexType>

<xs:complexType name="LockStatusSubResponseDataType">
<xs:attribute name="LockType" type="tns:LockTypes" use="optional" />
<xs:attribute name="LockID" type="tns:guid" use="optional" />
<xs:attribute name="LockedBy" type="xs:string" use="optional" />
</xs:complexType>

<xs:complexType name="PropertiesSubResponseDataType">
<xs:sequence>
<xs:element name="PropertyIds" minOccurs="0" maxOccurs="1" type="tns:PropertyIdsType" />
<xs:element name="PropertyValues" minOccurs="0" maxOccurs="1" type="tns:PropertyValuesType" />
</xs:sequence>
</xs:complexType>

<xs:complexType name="PropertyIdsType">
<xs:sequence>
<xs:element name="PropertyId" minOccurs="0" maxOccurs="unbounded" type="tns:PropertyIdType" />
</xs:sequence>
</xs:complexType>

<xs:complexType name="PropertyIdType">
<xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType name="PropertyValuesType">
<xs:sequence>
<xs:element name="PropertyValue" minOccurs="0" maxOccurs="unbounded" type="tns:PropertyValueType" />
</xs:sequence>
</xs:complexType>

<xs:complexType name="PropertyValueType">
<xs:attribute name="id" type="xs:string" use="required" />
<xs:attribute name="value" type="xs:string" use="required" />
</xs:complexType>
<xs:complexType name="SchemaLockSubResponseType">
  <xs:complexContent>
    <xs:extension base="tns:SubResponseType">
      <xs:sequence minOccurs="1" maxOccurs="1">
        <xs:element name="SubResponseData" type="tns:SchemaLockSubResponseDataType" />
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>

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

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

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

<xs:element name="GetVersionsResponse">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="1" maxOccurs="1" name="GetVersionsResult">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="results" minOccurs="1" maxOccurs="1" type="tns:Results" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="versioning" maxOccurs="1" minOccurs="1">
        <xs:complexType>
          <xs:attribute name="enabled" type="xs:unsignedByte" use="required" />
        </xs:complexType>
      </xs:element>
      <xs:element name="settings" maxOccurs="1" minOccurs="1">
        <xs:complexType>
          <xs:attribute name="id" type="xs:string" use="required" />
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:complexType>
  <xs:attribute name="url" type="xs:string" use="required" />
</xs:complexType>

<xs:complexType>
  <xs:element name="result" maxOccurs="unbounded" minOccurs="1" type="tns:VersionData"/>
</xs:complexType>

<xs:complexType name="VersionData">
  <xs:attribute name="version" type="xs:string" use="required" />
  <xs:attribute name="created" type="xs:string" use="required" />
  <xs:attribute name="createdRaw" type="xs:string" use="required" />
  <xs:attribute name="createdBy" type="xs:string" use="required" />
  <xs:attribute name="createdByName" type="xs:string" use="optional" />
  <xs:attribute name="size" type="xs:unsignedLong" use="required" />
  <xs:attribute name="comments" type="xs:string" use="required" />
</xs:complexType>

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

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

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

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

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

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

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

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

<xs:element name="Response">
<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="UrlIsEncoded" type="tns:TRUEFALSE" use="required"/>
<xs:attribute name="RequestToken" type="xs:nonNegativeInteger" use="optional" />
<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:attribute name="SuggestedFileName" type="xs:string" use="optional" />
<xs:attribute name="ResourceID" type="xs:string" use="optional" />
<xs:attribute name="IntervalOverride" type="xs:nonNegativeInteger" 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:complexType>
</xs:element>
<xs:element name="WebUrl" type="xs:string" use="required"/>
<xs:attribute name="WebUrlIsEncoded" type="tns:TRUEFALSE" 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 updates to those products.

- Microsoft Office 2010 suites
- Microsoft Office 2013
- Microsoft Office 2016
- Microsoft Office 2019
- Microsoft SharePoint Workspace 2010
- Microsoft SharePoint Foundation 2010
- Microsoft SharePoint Foundation 2013
- Microsoft SharePoint Server 2010
- Microsoft SharePoint Server 2013
- Microsoft SharePoint Server 2016
- Microsoft SharePoint Server 2019
- Windows 8.1 Update
- Windows 10 operating system

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

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

<1> Section 2.2.3.2: Attribute not supported by Microsoft SharePoint Server 2010.
<2> Section 2.2.3.2: Attribute not supported by Office 2010.
<3> Section 2.2.3.2: Office 2010 does not support UserAgent attribute.
<4> Section 2.2.3.2: Office 2010 does not support UserAgentClient attribute.
<5> Section 2.2.3.2: Office 2010 does not support UserAgentPlatform attribute.
<6> Section 2.2.3.2: Office 2010 does not support Build attribute.
<7> Section 2.2.3.2: Office 2010 and Office 2013 do not support ParentFolderResourceID attribute.
<8> Section 2.2.3.2: Office 2010 and Office 2013 do not support ShouldReturnDisambiguatedFileName attribute.
<9> Section 2.2.3.2: Office 2010 and Office 2013 do not support ResourceID attribute.
<10> Section 2.2.3.2: Office 2010 and Office 2013 do not support UseResourceID attribute.
<11> Section 2.2.3.5: SharePoint Server 2010 will return 2 ErrorCode attributes in Response element. SharePoint Server 2013 will not return Response element.
Section 2.2.3.5: The RequestToken attribute is ignored by SharePoint Foundation 2013, SharePoint Server 2013 and SharePoint Server 2016.

Section 2.2.3.5: The SuggestedFileName attribute is not supported by SharePoint Server 2010 and SharePoint Server 2013.

Section 2.2.3.5: The ResourceID attribute is not supported by SharePoint Server 2010 and SharePoint Server 2013.

Section 2.2.3.7: In SharePoint Server 2016, ErrorCode attribute is not present if this protocol is not enabled on the protocol server.

Section 2.2.4.8: SharePoint Server 2010 and SharePoint Server 2013 do not support the ServerCorrelationId attribute.

Section 2.2.5.5: SharePoint Server 2010, SharePoint Server 2013 and SharePoint Server 2016 will never return this value to the client.

Section 2.2.5.6: SharePoint Server 2010 and SharePoint Server 2013 do not support this error code type.

Section 2.2.5.6: SharePoint Server 2010 does not support this error code type.

Section 2.2.5.14: SharePoint Server 2010 does not support this simple type.

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

Section 2.2.8.2: Servers running Microsoft Office 2010 suites return a time value that is not the current time. The protocol client does not change its behavior because the protocol client uses the difference between ServerTime values, not the difference between ServerTime and the time at the client.

Section 2.2.8.2: SharePoint Server 2010 will not return CoauthStatus attribute when client tries to join the coauthoring session and the subrequest falls back to an exclusive lock subrequest.

Section 2.3.1.1: Word 2010 and PowerPoint 2010 use string "29358EC1-E813-4793-8E70-ED0344E7B73C" for the SchemaLockID attribute sent in the different subrequests of type Coauthoring, ExclusiveLock, and SchemaLock.

Section 2.3.1.5: In SharePoint Server 2010, if the CoauthRequestType attribute is not provided, an "InvalidArgument" error code will be returned as part of the SubResponseData element associated with the coauthoring subresponse.

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

Section 2.3.1.7: SharePoint Server 2010 will not return CoauthStatus attribute when client tries to join the coauthoring session and the subrequest falls back to an exclusive lock subrequest.

Section 2.3.1.9: In SharePoint Server 2010, if the ExclusiveLockRequestType attribute is not provided, an "InvalidArgument" error code will be returned as part of the SubResponseData element associated with the exclusive lock subresponse.

Section 2.3.1.9: Word 2010 and PowerPoint 2010 use the string "29358EC1-E813-4793-8E70-ED0344E7B73C" for the SchemaLockID attribute sent in the different subrequests of type Coauthoring, ExclusiveLock, and SchemaLock.
In SharePoint Server 2010, if the SchemaLockRequestType attribute is not provided, an "InvalidArgument" error code will be returned as part of the SubResponseData element associated with the schema lock subresponse.


Servers running Office 2010 return a time value that is not the current time. The protocol client does not change its behavior because the protocol client uses the difference between ServerTime values, not the difference between the ServerTime and the time at the client.

In SharePoint Server 2010, if the FileOperation attribute is not provided, an "InvalidArgument" error code will be returned as part of the SubResponseData element associated with the file operation subresponse.

There is an additional authentication prefix if claim-based authentication mode is enabled. For example: "i:0#.w|contoso\user01".

Servers running Office 2010 save all changes to the underlying store at the end of processing a subrequest, with the following four exceptions:

1. The subrequest is a Put Changes subrequest, as described in [MS-FSSHTTPB] section 2.2.2.1.4, and the Partial bit, as described in [MS-FSSHTTPB] section 2.2.2.1.4, of the Put Changes request is 1. In this case, servers running Office 2010 do not save changes to the underlying store. Instead, the changes are stored in a write cache.

2. Servers running Office 2010 write to an intermediate write cache while processing a subrequest if the subrequest is a Put Changes subrequest, as described in [MS-FSSHTTPB] section 2.2.2.1.4. If writing to the write cache fails, servers running Office 2010 do not save changes to the underlying store.

3. If prior to the current attempt to save changes to the underlying store, there have been from 26 through 73 consecutive failed attempts immediately preceding the current attempt, and the most recent failed attempt is within 60 minutes of the current subrequest, servers running Office 2010 do not save changes to the underlying store. Instead, the changes are stored in a write cache.

4. If prior to the current attempt to save changes to the underlying store there have been more than 73 consecutive failed attempts immediately preceding and the most recent failed attempt is within 24 hours of the current subrequest then servers running Office 2010 will not save changes to the underlying store. Instead the changes will be stored in a write cache.

Attribute not supported by SharePoint Foundation 2013 and SharePoint Server 2013 in Put Changes subrequest, as described in [MS-FSSHTTPB] section 2.2.1.1.4.

Attribute not supported by SharePoint Server 2013.

Attribute not supported by SharePoint Server 2013.

Attribute not supported by Office 2010.

SharePoint Server 2010 will return an error code value "FileNotLockedOnServerAsCoauthDisabled" if the AllowFallbackToExclusive attribute is set to false.

SharePoint Server 2010 will return an error code of "Success" if there are other clients present in the coauthoring session.

SharePoint Server 2013 and SharePoint Server 2010 return an error code of "Success" if there is an exclusive lock on the file or if there is a shared lock with a different shared lock identifier and a valid coauthoring session containing more than one clients. SharePoint Server
2013 and SharePoint Server 2010 return an error code of "FileAlreadyLockedOnServer" if there is a shared lock with a different shared lock identifier and a coauthoring session containing one client.

<43> Section 3.1.4.3.3: SharePoint Server 2010 will return an error code value "FileNotLockedOnServerAsCoauthDisabled".

<44> Section 3.1.4.3.6: SharePoint Server 2010 returns an error code value set to "LockRequestFail" if there is no shared lock.

<45> Section 3.1.4.3.6: SharePoint Server 2010 returns an error code value set to "LockRequestFail" if there is no coauthoring session for the file.

<46> Section 3.1.4.3.7: In SharePoint Server 2010, the protocol server returns an error code value set to "InvalidCoauthSession".

<47> Section 3.1.4.4.1: SharePoint Server 2010 will return an error code value "FileNotLockedOnServerAsCoauthDisabled" if the AllowFallbackToExclusive attribute is set to false.

<48> Section 3.1.4.4.2: SharePoint Server 2010 will return an error code of "Success" if there are other clients present in the coauthoring session.

<49> Section 3.1.4.4.2: SharePoint Server 2013 and SharePoint Server 2010 return error code "Success" if there is an exclusive lock on the file or there is a shared lock with a different shared lock identifier and valid coauthoring session with more than one clients in it. SharePoint Server 2013 and SharePoint Server 2010 return an error code "FileAlreadyLockedOnServer" if there is a shared lock with a different shared lock identifier and a coauthoring session with one client in it.

<50> Section 3.1.4.4.3: SharePoint Server 2010 will return an error code value "FileNotLockedOnServerAsCoauthDisabled".

<51> Section 3.1.4.4.7: Servers running Office 2010 return a time value that is not the current time. The protocol client does not change its behavior because the protocol client uses the difference between ServerTime values, not the difference between ServerTime and the time at the client.

<52> Section 3.1.4.4.8: SharePoint Server 2010 does not support this operation.

<53> Section 3.1.4.4.8: Servers running Office 2013 automatically add a client to the editors table when it takes a coauthoring or schema lock—if the client protocol version is 2.2 or higher as described in section 2.2.5.10. Editors table subrequests are used by clients running Office 2013 that do not take locks on documents before editing them.

<54> Section 3.1.4.4.8: Only 4 key/value pairs can be associated with an editor on servers running Office 2013.

<55> Section 3.1.4.4.8: On servers running Office 2013, the editors table partition identifier is the GUID "7808F4DD-2385-49d6-B7CE-37AC-A5E4-3602", and the editors table is represented as a compressed XML fragment.

<56> Section 3.1.4.4.9: SharePoint Server 2010 does not support this operation.

<57> Section 3.1.4.4.10: SharePoint Server 2010 does not support this operation.

<58> Section 3.1.4.4.11: SharePoint Server 2010 and SharePoint Server 2013 do not support this operation.

<59> Section 3.1.4.4.12: SharePoint Server 2010 and SharePoint Server 2013 do not support this operation.

<60> Section 3.1.4.4.13: SharePoint Server 2010 does not support the AmIAIAlone operation.
Section 3.1.4.14: SharePoint Server 2010 and SharePoint Server 2013 do not support the LockStatus operation.

Section 3.1.4.15: SharePoint Server 2010 and SharePoint Server 2013 do not support the Properties operation.
8 Change Tracking

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

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

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

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

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

The changes made to this document are listed in the following table. For more information, please contact [dochelp@microsoft.com](mailto:dochelp@microsoft.com).

<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Revision class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5 Prerequisites/Preconditions</td>
<td>Added description about the whole document URL and canonical URLs.</td>
<td>Minor</td>
</tr>
<tr>
<td>2.2.5.6 GenericErrorCodeTypes</td>
<td>Added description for error ColdStoreConcurrencyViolation to specify client can retry the request.</td>
<td>Minor</td>
</tr>
</tbody>
</table>
LockTypes simple type 45
LockAndCoauthRelatedErrorCodeTypes simple type 43
LockTypes simple type 45

IM
Message processing 104
Message Schemas 148
Messages
attribute groups (section 2.2.8 51, section 2.3.3 93)
attributes 51
CellRequestErrorCodeTypes simple type 87
CellSubRequestDataOptionalAttributes attribute group 93
CellSubRequestDataGetType complex type 58
CellSubRequestType complex type 59
CellSubResponseDataOptionalAttributes attribute group 95
CellSubResponseDataGetType complex type 59
CellSubResponseType complex type 60
CoauthRequestTypes simple type 88
CoauthStatusType simple type 38
CoauthSubRequestDataOptionalAttributes attribute group 96
CoauthSubRequestDataGetType complex type 60
CoauthSubRequestType complex type 62
CoauthSubResponseDataGetType complex type 63
CoauthSubResponseType complex type 63
common data structures 55
collection types (section 2.2.4 27, section 2.3.1 55)
CollectionCheckRelatedErrorCodeTypes simple type 39
DependencyTypes simple type 39
elements 18
enumerated 16
ErrorCodes simple type 87
ExclusiveLockRequestTypes simple type (section 2.3.2.3 89, section 2.3.2.8 91, section 2.3.2.9 92, section 2.3.2.10 92)
ExclusiveLockReturnReasonTypes simple type 41
ExclusiveLockSubRequestDataOptionalAttributes attribute group 98
ExclusiveLockSubRequestDataGetType complex type 64
ExclusiveLockSubRequestType complex type (section 2.3.1.10 65, section 2.3.1.34 77, section 2.3.1.37 78, section 2.3.1.46 82, section 2.3.1.49 83, section 2.3.1.53 84)
ExclusiveLockSubResponseDataGetType complex type 65
ExclusiveLockSubResponseType complex type 66
F
Fields - vendor-extensions 15
FileVersionNumberType simple type 50
Full XML schema 148
Request Message Schema 148
Response Message Schema 157
G
GenericErrorCodeTypes simple type 41
GenericPropertiesType complex type 27
GetDocMetaInfoPropertySetType complex type 74
GetDocMetaInfoPropertyType complex type 75
GetDocMetaInfoSubRequestType complex type 74
GetDocMetaInfoSubResponseType complex type 75
Glossary 9
Groups 51
GUID simple type 43
I
Implementer - security considerations 147
Include element 19
Index of security parameters 147
Informative references 12
Initialization
- server 104
Introduction 9
L
Local events
- server 134
LockAndCoauthRelatedErrorCodeTypes simple type 43
LockTypes simple type 45

[MS-FSSHTTP] - v20190924
File Synchronization via SOAP over HTTP Protocol
Copyright © 2019 Microsoft Corporation
Release: September 24, 2019
File Synchronization via SOAP over HTTP Protocol

Copyright © 2019 Microsoft Corporation

Release: September 24, 2019

Namespaces

NewEditorsTableCategoryErrorCodeTypes simple type 49

PropertyType complex type 28

Request 17

Request element 19

Request message 17

RequestCollection element 23

RequestVersion element 23

Response 17

Response element 24

ResponseCollection element 25

ResponseVersion element 25

SchemaLockRequestTypes simple type (section 2.3.2.4 89, section 2.3.2.5 90)

SchemaLockSubRequestDataOptionalAttributes attribute group (section 2.3.3.5 99, section 2.3.3.8 101, section 2.3.3.9 102, section 2.3.3.10 102)

SchemaLockSubRequestDataResponseType complex type (section 2.3.1.13 66, section 2.3.1.23 71, section 2.3.1.33 76, section 2.3.1.36 77, section 2.3.1.45 81, section 2.3.1.52 84)

SchemaLockSubRequestResponseType complex type 68

SchemaLockSubResponseDataResponseType complex type 68

ServerTimeSubRequestResponseType complex type 69

ServerTimeSubResponseDataResponseType complex type 70

ServerTimeSubResponseType complex type (section 2.3.1.19 70, section 2.3.1.35 77, section 2.3.1.39 79, section 2.3.1.48 82, section 2.3.1.51 83, section 2.3.1.55 85)

Simple types (section 2.2.3 37, section 2.3.2 86)

SOAP Fault 18

SOAP Fault message 18

SubRequest element 26

SubRequestAttributeType simple type 47

SubRequestData element 26

SubRequestDataGenericType complex type 28

SubRequestDataOptionalAttributes attribute group 51

SubRequestElementGenericType complex type 29

SubRequestType complex type 31

SubResponse element 26

SubResponseData element 27

SubResponseDataGenericType complex type 32

SubResponseDataOptionalAttributes attribute group 54

SubResponseElementGenericType complex type 33

SubResponseType complex type 35

syntax (section 2.2 16, section 2.3 55)

transport 16

TRUEFALSE simple type 49

UserLoginType simple type 91

UserNameType simple type 91

VersioningRelatedErrorCodeTypes simple type 50

VersionNumberType simple type 49

VersionType complex type 36

WhoAmISubRequestType complex type (section 2.3.1.20 70, section 2.3.1.24 73, section 2.3.1.31 75)

WhoAmISubResponseType complex type 100

WhoAmISubResponseDataOptionalAttributes attribute group 100

WhoAmISubResponseDataType complex type (section 2.3.1.21 71, section 2.3.1.27 74, section 2.3.1.38 78, section 2.3.1.47 82, section 2.3.1.50 83, section 2.3.1.54 85)

WhoAmISubResponseType complex type (section 2.3.1.22 71, section 2.3.1.23 73, section 2.3.1.32 76)

MinorVersionNumberType simple type 46

Namespaces

NewEditorsTableCategoryErrorCodeTypes simple type 49

Normative references

Operations

AmIAlone Subrequest 132

Cell Subrequest 106

Colauth Subrequest 108

Common Message Processing Rules and Events 105

EditorsTable Subrequest 126

ExclusiveLock Subrequest 120

FileOperation Subrequest 131

GetDocMetaInfo Subrequest 128

GetVersions Subrequest 129

LockStatus Subrequest 132

Properties Subrequest 133

SchemaLock Subrequest 115

ServerTime Subrequest 126

Versioning Subrequest 130

WhoAmI Subrequest 125

Overview (Synopsis) 12

Parameters - security index 147

Preconditions 14

Prerequisites 14

Product behavior 168

PropertyType complex type 28

Protocol Details

Overview 104

References

informative 12

normative 11

Relationship to other protocols 14

Request element 19

RequestCollection element 23

RequestVersion element 23

Response element 24

ResponseCollection element 25

ResponseVersion element 25

S

176 / 178
SchemaLockRequestTypes simple type (section 2.3.2.4 89, section 2.3.2.5 90)
SchemaLockSubRequestDataOptionalAttributes attribute group (section 2.3.3.7 100, section 2.3.3.8 101, section 2.3.3.9 102, section 2.3.3.10 102)
SchemaLockSubRequestDataType complex type (section 2.3.1.13 66, section 2.3.1.23 71, section 2.3.1.33 76, section 2.3.1.36 77, section 2.3.1.45 81, section 2.3.1.52 84)
SchemaLockSubRequestType complex type 68
SchemaLockSubResponseType complex type 68
Security implementer considerations 147
parameter index 147
Sequencing rules server 104
Server abstract data model 104
AmIAlone Subrequest operation 132
Cell Subrequest operation 106
Coauth Subrequest operation 108
Common Message Processing Rules and Events operation 105
EditorsTable Subrequest operation 126
ExclusiveLock Subrequest operation 120
FileOperation Subrequest operation 131
GetDocMetaInfo Subrequest operation 128
GetVersions Subrequest operation 129
initialization 104
local events 134
LockStatus Subrequest operation 132
message processing 104
overview 104
Properties Subrequest operation 133
SchemaLock Subrequest operation 115
sequencing rules 104
ServerTime Subrequest operation 126
timer events 134
timers 104
Versioning Subrequest operation 130
WhoAmI Subrequest operation 125
ServerTimeSubRequestType complex type 69
ServerTimeSubResponseDataTypetype 70
ServerTimeSubResponseType complex type (section 2.3.1.19 70, section 2.3.1.35 77, section 2.3.1.39 79, section 2.3.1.48 82, section 2.3.1.51 83, section 2.3.1.55 85)
Simple types (section 2.2.5 37, section 2.3.2 86)
CellRequestErrorCodeTypes 87
CoauthRequestTypes 88
CoauthStatusType 38
DependencyCheckRelatedErrorCodeTypes 39
DependencyTypes 39
ErrorCodeTypes 40
ExclusiveLockRequestTypes (section 2.3.2.3 89, section 2.3.2.8 91, section 2.3.2.9 92, section 2.3.2.10 92)
ExclusiveLockReturnReasonTypes 41
FileVersionNumberType 50
GenericErrorCodeTypes 41
GUID 43
LockAndCoauthRelatedErrorCodeTypes 43
LockTypes 45
MinorVersionNumberType 46
NewEditorsTableCategoryErrorCodeTypes 49
SchemaLockRequestTypes (section 2.3.2.4 89, section 2.3.2.5 90)
SubRequestAttributeType 47
TRUEFALSE 49
UserLoginType 91
UserNameType 91
VersioningRelatedErrorCodeTypes 50
VersionNumberType 49
Standards assignments 15
SubRequest element 26
SubRequestAttributeType simple type 47
SubRequestData element 26
SubRequestDataGenericType complex type 28
SubRequestDataOptionalAttributes attribute group 51
SubRequestElementTypeGenericType complex type 29
SubRequestType complex type 31
SubResponse element 26
SubResponseData element 27
SubResponseDataGenericType complex type 32
SubResponseDataOptionalAttributes attribute group 54
SubResponseElementTypeGenericType complex type 33
SubResponseType complex type 35
Successful file open of a coauthorable document example 135
Successful file open of a non coauthorable document example 140
Successful file save of a coauthorable document example 138
Successful file save of a non coauthorable document example 143
Syntax messages - overview (section 2.2 16, section 2.3 55)

T

Timer events server 134
timers server 104
Tracking changes 173
Transport 16
TRUEFALSE simple type 49
types complex (section 2.2.4 27, section 2.3.1 46)
simple (section 2.2.5 37, section 2.3.2 86)

U

Unsuccessful file open of a coauthorable document example 145
Unsuccessful file open of a non coauthorable document example 142
UserLoginType simple type 91
UserNameType simple type 91

V

Vendor-extensible fields 15
Versioning 14
VersioningRelatedErrorCodeTypes simple type 50
**VersionNumberType simple type** 49

**VersionType complex type** 36

W

WhoAmISubRequestType complex type ([section](#) 2.3.1.20 70, [section](#) 2.3.1.24 73, [section](#) 2.3.1.31 75)

WhoAmISubResponseDataOptionalAttributes attribute group 100

WhoAmISubResponseDataResponseType complex type ([section](#) 2.3.1.21 71, [section](#) 2.3.1.27 74, [section](#) 2.3.1.38 78, [section](#) 2.3.1.47 82, [section](#) 2.3.1.50 83, [section](#) 2.3.1.54 85)

WhoAmISubResponseType complex type ([section](#) 2.3.1.22 71, [section](#) 2.3.1.25 73, [section](#) 2.3.1.32 76)

X

**XML schema** 148

**Request Message Schema** 148

**Response Message Schema** 157