

# [MS-FRS1-Diff]: File Replication Service Protocol

---

## Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation (“this documentation”) for protocols, file formats, data portability, computer languages, and standards support. Additionally, overview documents cover inter-protocol relationships and interactions.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you can make copies of it in order to develop implementations of the technologies that are described in this documentation and can distribute portions of it in your implementations that use these technologies or in your documentation as necessary to properly document the implementation. You can also distribute in your implementation, with or without modification, any schemas, IDLs, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications documentation.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that might cover your implementations of the technologies described in the Open Specifications documentation. Neither this notice nor Microsoft's delivery of this documentation grants any licenses under those patents or any other Microsoft patents. However, a given Open Specifications document might be covered by the Microsoft [Open Specifications Promise](#) or the [Microsoft Community Promise](#). If you would prefer a written license, or if the technologies described in this documentation are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting [iplg@microsoft.com](mailto: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](http://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](mailto:dochelp@microsoft.com).

## Revision Summary

| Date       | Revision History | Revision Class | Comments  |
|------------|------------------|----------------|---|
| 3/2/2007   | 1.0              | Major          | Updated and revised the technical content.                |
| 4/3/2007   | 1.1              | Minor          | Clarified the meaning of the technical content.           |
| 5/11/2007  | 2.0              | Major          | Updated technical content                                 |
| 6/1/2007   | 2.1              | Minor          | Clarified the meaning of the technical content.           |
| 7/3/2007   | 2.1.1            | Editorial      | Changed language and formatting in the technical content. |
| 8/10/2007  | 2.1.2            | Editorial      | Changed language and formatting in the technical content. |
| 9/28/2007  | 2.1.3            | Editorial      | Changed language and formatting in the technical content. |
| 10/23/2007 | 2.1.4            | Editorial      | Changed language and formatting in the technical content. |
| 1/25/2008  | 2.1.5            | Editorial      | Changed language and formatting in the technical content. |
| 3/14/2008  | 2.1.6            | Editorial      | Changed language and formatting in the technical content. |
| 6/20/2008  | 3.0              | Major          | Updated and revised the technical content.                |
| 7/25/2008  | 3.0.1            | Editorial      | Changed language and formatting in the technical content. |
| 8/29/2008  | 4.0              | Major          | Updated and revised the technical content.                |
| 10/24/2008 | 4.0.1            | Editorial      | Changed language and formatting in the technical content. |
| 12/5/2008  | 5.0              | Major          | Updated and revised the technical content.                |
| 1/16/2009  | 6.0              | Major          | Updated and revised the technical content.                |
| 2/27/2009  | 7.0              | Major          | Updated and revised the technical content.                |
| 4/10/2009  | 8.0              | Major          | Updated and revised the technical content.                |
| 5/22/2009  | 9.0              | Major          | Updated and revised the technical content.                |
| 7/2/2009   | 9.0.1            | Editorial      | Changed language and formatting in the technical content. |
| 8/14/2009  | 9.0.2            | Editorial      | Changed language and formatting in the technical content. |
| 9/25/2009  | 10.0             | Major          | Updated and revised the technical content.                |
| 11/6/2009  | 11.0             | Major          | Updated and revised the technical content.                |
| 12/18/2009 | 11.1             | Minor          | Clarified the meaning of the technical content.           |
| 1/29/2010  | 12.0             | Major          | Updated and revised the technical content.                |
| 3/12/2010  | 12.0.1           | Editorial      | Changed language and formatting in the technical content. |
| 4/23/2010  | 13.0             | Major          | Updated and revised the technical content.                |
| 6/4/2010   | 14.0             | Major          | Updated and revised the technical content.                |
| 7/16/2010  | 15.0             | Major          | Updated and revised the technical content.                |
| 8/27/2010  | 15.0             | None           | No changes to the meaning, language, or formatting of the |

| <b>Date</b>      | <b>Revision History</b> | <b>Revision Class</b> | <b>Comments</b>  |
|------------------|-------------------------|-----------------------|--|
|                  |                         |                       | technical content.   |
| 10/8/2010        | 15.0                    | None                  | No changes to the meaning, language, or formatting of the technical content. |
| 11/19/2010       | 15.0                    | None                  | No changes to the meaning, language, or formatting of the technical content. |
| 1/7/2011         | 15.0                    | None                  | No changes to the meaning, language, or formatting of the technical content. |
| 2/11/2011        | 15.0                    | None                  | No changes to the meaning, language, or formatting of the technical content. |
| 3/25/2011        | 15.0                    | None                  | No changes to the meaning, language, or formatting of the technical content. |
| 5/6/2011         | 16.0                    | Major                 | Updated and revised the technical content.                                   |
| 6/17/2011        | 17.0                    | Major                 | Updated and revised the technical content.                                   |
| 9/23/2011        | 18.0                    | Major                 | Updated and revised the technical content.                                   |
| 12/16/2011       | 19.0                    | Major                 | Updated and revised the technical content.                                   |
| 3/30/2012        | 20.0                    | Major                 | Updated and revised the technical content.                                   |
| 7/12/2012        | 21.0                    | Major                 | Updated and revised the technical content.                                   |
| 10/25/2012       | 21.0                    | None                  | No changes to the meaning, language, or formatting of the technical content. |
| 1/31/2013        | 22.0                    | Major                 | Updated and revised the technical content.                                   |
| 8/8/2013         | 23.0                    | Major                 | Updated and revised the technical content.                                   |
| 11/14/2013       | 24.0                    | Major                 | Updated and revised the technical content.                                   |
| 2/13/2014        | 24.0                    | None                  | No changes to the meaning, language, or formatting of the technical content. |
| 5/15/2014        | 24.0                    | None                  | No changes to the meaning, language, or formatting of the technical content. |
| 6/30/2015        | 25.0                    | Major                 | Significantly changed the technical content.                                 |
| 10/16/2015       | 25.0                    | None                  | No changes to the meaning, language, or formatting of the technical content. |
| 7/14/2016        | 25.0                    | None                  | No changes to the meaning, language, or formatting of the technical content. |
| 6/1/2017         | 25.0                    | None                  | No changes to the meaning, language, or formatting of the technical content. |
| <u>9/15/2017</u> | <u>26.0</u>             | <u>Major</u>          | <u>Significantly changed the technical content.</u>                          |

# Table of Contents

|            |   |           |
|------------|---|-----------|
| <b>1</b>   | <b>Introduction .....</b>                   | <b>8</b>  |
| 1.1        | Glossary .....                              | 8         |
| 1.2        | References .....                            | 13        |
| 1.2.1      | Normative References .....                  | 13        |
| 1.2.2      | Informative References .....                | 14        |
| 1.3        | Overview .....                              | 14        |
| 1.4        | Relationship to Other Protocols .....       | 16        |
| 1.5        | Prerequisites/Preconditions .....           | 17        |
| 1.6        | Applicability Statement .....               | 17        |
| 1.7        | Versioning and Capability Negotiation ..... | 17        |
| 1.7.1      | NtFrsApi .....                              | 17        |
| 1.7.2      | FRSRPC .....                                | 17        |
| 1.8        | Vendor-Extensible Fields .....              | 18        |
| 1.9        | Standards Assignments.....                  | 18        |
| <b>2</b>   | <b>Messages.....</b>                        | <b>19</b> |
| 2.1        | Transport.....                              | 19        |
| 2.2        | Common Data Types .....                     | 19        |
| 2.2.1      | WCHAR and PWCHAR .....                      | 19        |
| 2.2.2      | NtFrsApi Common Data Types .....            | 19        |
| 2.2.2.1    | NTFRSAPI_INFO.....                          | 19        |
| 2.2.3      | FRSRPC Common Data Types .....              | 21        |
| 2.2.3.1    | GVSN .....                                  | 21        |
| 2.2.3.2    | CHANGE_ORDER_COMMAND .....                  | 21        |
| 2.2.3.3    | CO_RECORD_EXTENSION_WIN2K.....              | 28        |
| 2.2.3.4    | CHANGE_ORDER_RECORD_EXTENSION.....          | 28        |
| 2.2.3.5    | COMM_PACKET and PCOMM_PACKET.....           | 29        |
| 2.2.3.6    | COMM_PACKET Elements .....                  | 34        |
| 2.2.3.6.1  | COMM_BOP .....                              | 34        |
| 2.2.3.6.2  | COMM_COMMAND .....                          | 34        |
| 2.2.3.6.3  | COMM_TO .....                               | 35        |
| 2.2.3.6.4  | COMM_FROM .....                             | 36        |
| 2.2.3.6.5  | COMM_REPLICA .....                          | 36        |
| 2.2.3.6.6  | COMM_CXTION .....                           | 37        |
| 2.2.3.6.7  | COMM_JOIN_GUID .....                        | 37        |
| 2.2.3.6.8  | COMM_LAST_JOIN_TIME .....                   | 38        |
| 2.2.3.6.9  | COMM_VVECTOR.....                           | 38        |
| 2.2.3.6.10 | COMM_JOIN_TIME.....                         | 39        |
| 2.2.3.6.11 | COMM_REPLICA_VERSION_GUID .....             | 39        |
| 2.2.3.6.12 | COMM_COMPRESSION_GUID .....                 | 39        |
| 2.2.3.6.13 | COMM_BLOCK.....                             | 40        |
| 2.2.3.6.14 | COMM_BLOCK_SIZE .....                       | 40        |
| 2.2.3.6.15 | COMM_FILE_SIZE.....                         | 41        |
| 2.2.3.6.16 | COMM_FILE_OFFSET.....                       | 41        |
| 2.2.3.6.17 | COMM_GVSN .....                             | 41        |
| 2.2.3.6.18 | COMM_CO_GUID.....                           | 42        |
| 2.2.3.6.19 | COMM_CO_SEQUENCE_NUMBER .....               | 42        |
| 2.2.3.6.20 | COMM_REMOTE_CO.....                         | 43        |
| 2.2.3.6.21 | COMM_CO_EXT_WIN2K .....                     | 43        |
| 2.2.3.6.22 | COMM_CO_EXTENSION_2 .....                   | 43        |
| 2.2.3.6.23 | COMM_EOP.....                               | 44        |
| 2.2.3.7    | DATA_EXTENSION_CHECKSUM.....                | 44        |
| 2.2.3.8    | DATA_EXTENSION_PREFIX.....                  | 44        |
| 2.2.3.9    | DATA_EXTENSION_RETRY_TIMEOUT .....          | 45        |
| 2.2.3.10   | STAGE_HEADER.....                           | 45        |

|            |   |           |
|------------|---|-----------|
| 2.3        | Directory Service Schema Elements .....                                 | 48        |
| 2.3.1      | DFS Active Directory Configuration .....                                | 50        |
| 2.3.1.1    | NTFRS Settings Object.....  | 51        |
| 2.3.1.2    | NTFRS Replica Set Object .....  | 52        |
| 2.3.1.3    | NTFRS Member Object.....  | 54        |
| 2.3.1.4    | NTDS Active Directory Service Agent (nTDSDSA) Object.....               | 54        |
| 2.3.1.5    | NTDS Active Directory Service Agent Read Only (nTDSARSARO) Object ..... | 55        |
| 2.3.1.6    | NTDS Connection Object.....   | 55        |
| 2.3.1.7    | Computer Object .....   | 56        |
| 2.3.1.8    | NTFRS Subscriptions Container .....                                     | 56        |
| 2.3.1.9    | NTFRS Subscriber Object .....   | 56        |
| 2.3.1.10   | Object Types .....  | 57        |
| 2.3.1.11   | Top class .....   | 58        |
| 2.3.2      | SYSVOL Active Directory Configuration .....                             | 58        |
| 2.3.2.1    | NTFRS Settings Object.....  | 59        |
| 2.3.2.2    | NTFRS Replica Set Object .....  | 59        |
| 2.3.2.3    | NTFRS Member Object.....  | 60        |
| 2.3.2.4    | NTDS Connection Object.....   | 60        |
| 2.3.2.5    | Computer Object .....   | 60        |
| 2.3.2.6    | NTFRS Subscriptions Container .....                                     | 60        |
| 2.3.2.7    | NTFRS Subscriber Object .....   | 60        |
| 2.4        | FRS Performance Counters .....  | 60        |
| 2.4.1      | FileReplicaConn Object .....  | 61        |
| 2.4.2      | FileReplicaSet.....   | 62        |
| <b>3</b>   | <b>Protocol Details.....</b>  | <b>67</b> |
| 3.1        | Common Details .....  | 67        |
| 3.1.1      | Abstract Data Model.....  | 67        |
| 3.1.1.1    | File System .....   | 67        |
| 3.1.1.2    | Replica Set Object.....   | 67        |
| 3.1.1.3    | Member Object (Replica Member Object) .....                             | 68        |
| 3.1.1.4    | Replica Tree .....  | 68        |
| 3.1.1.5    | IDTable .....   | 68        |
| 3.1.1.6    | Inbound Log Object (InLog) .....  | 69        |
| 3.1.1.7    | Outbound Log Object (OutLog) .....                                      | 69        |
| 3.1.1.8    | Connection Object.....  | 69        |
| 3.1.1.9    | Staging File Object.....  | 70        |
| 3.1.1.10   | Change Order Object.....  | 70        |
| 3.1.1.10.1 | Local Change Order .....  | 70        |
| 3.1.1.10.2 | Retry Change Order.....   | 71        |
| 3.1.1.10.3 | Directed Change Order.....  | 71        |
| 3.1.1.10.4 | Out-of-Order Change Order .....   | 71        |
| 3.1.1.10.5 | Skip-VV-Update Change Order .....                                       | 71        |
| 3.1.1.11   | Version Vector Object.....  | 72        |
| 3.1.1.12   | Communication Packet Object .....                                       | 72        |
| 3.1.2      | Timers .....  | 73        |
| 3.1.3      | Initialization.....   | 73        |
| 3.1.4      | Message Processing Events and Sequencing Rules .....                    | 73        |
| 3.1.5      | Timer Events.....   | 74        |
| 3.1.5.1    | Both Short and Long DS Polling Interval Timers .....                    | 74        |
| 3.1.6      | Other Local Events.....   | 77        |
| 3.2        | FRSAPI Interface.....   | 77        |
| 3.2.1      | Abstract Data Model.....  | 78        |
| 3.2.2      | Timers .....  | 78        |
| 3.2.3      | Initialization.....   | 78        |
| 3.2.4      | Message Processing Events and Sequencing Rules .....                    | 79        |
| 3.2.4.1    | NtFrsApi_Rpc_Set_DsPollingIntervalW (Opnum 4).....                      | 80        |
| 3.2.4.2    | NtFrsApi_Rpc_Get_DsPollingIntervalW (Opnum 5) .....                     | 81        |

|               |  |            |
|---------------|--|------------|
| 3.2.4.3       | NtFrsApi_Rpc_InfoW (Opnum 7) .....                           | 82         |
| 3.2.4.4       | NtFrsApi_Rpc_IsPathReplicated (Opnum 8) .....                | 83         |
| 3.2.4.5       | NtFrsApi_Rpc_WriterCommand (Opnum 9).....                    | 85         |
| 3.2.4.6       | NtFrsApi_Rpc_ForceReplication (Opnum 10).....                | 87         |
| 3.2.5         | Timer Events.....  | 88         |
| 3.2.6         | Other Local Events.....                                      | 88         |
| 3.3           | FRSRPC Interface .....                                       | 88         |
| 3.3.1         | Abstract Data Model.....                                     | 88         |
| 3.3.2         | Timers .....   | 88         |
| 3.3.2.1       | Connection Schedule Timer .....                              | 88         |
| 3.3.2.1.1     | SYSVOL Connection ScheduleTimer .....                        | 88         |
| 3.3.2.1.2     | DFS Connection Schedule .....                                | 89         |
| 3.3.3         | Initialization.....  | 89         |
| 3.3.4         | Message Processing Events and Sequencing Rules .....         | 89         |
| 3.3.4.1       | Change Orders .....  | 90         |
| 3.3.4.1.1     | File Is Added/Updated on the Upstream Partner .....          | 93         |
| 3.3.4.1.2     | File Is Removed on the Upstream Partner.....                 | 93         |
| 3.3.4.1.3     | File Is Renamed on the Upstream Partner.....                 | 93         |
| 3.3.4.1.4     | Folder Is Created/Updated on the Upstream Partner .....      | 94         |
| 3.3.4.1.5     | Folder Is Removed on the Upstream Partner .....              | 94         |
| 3.3.4.1.6     | Folder Is Renamed on the Upstream Partner.....               | 95         |
| 3.3.4.2       | FrsRpcStartPromotionParent Message (Opnum 2) .....           | 95         |
| 3.3.4.3       | FrsNOP Message (Opnum 3).....                                | 97         |
| 3.3.4.4       | FrsRpcSendCommPkt (Opnum 0) .....                            | 98         |
| 3.3.4.4.1     | Common Details.....  | 98         |
| 3.3.4.4.2     | COMM_COMMAND Is CMD_NEED_JOIN .....                          | 100        |
| 3.3.4.4.3     | COMM_COMMAND Is CMD_START_JOIN .....                         | 100        |
| 3.3.4.4.4     | COMM_COMMAND Is CMD_JOINING .....                            | 100        |
| 3.3.4.4.4.1   | Connection VVJoin .....                                      | 101        |
| 3.3.4.4.4.1.1 | Common Details for Initial Syncing a File and a Folder ..... | 101        |
| 3.3.4.4.4.1.2 | Initial Syncing a File .....                                 | 103        |
| 3.3.4.4.4.1.3 | Initial Syncing a Folder.....                                | 103        |
| 3.3.4.4.4.1.4 | CMD_VVJOIN_DONE Once Initial Sync Is Done.....               | 103        |
| 3.3.4.4.5     | COMM_COMMAND Is CMD_JOINED .....                             | 103        |
| 3.3.4.4.6     | COMM_COMMAND Is CMD_REMOTE_CO .....                          | 104        |
| 3.3.4.4.6.1   | Requesting a Staging File.....                               | 107        |
| 3.3.4.4.6.2   | Acknowledging the Change Is Done.....                        | 109        |
| 3.3.4.4.7     | COMM_COMMAND Is CMD_SEND_STAGE .....                         | 113        |
| 3.3.4.4.8     | COMM_COMMAND Is CMD_RECEIVING_STAGE .....                    | 115        |
| 3.3.4.4.9     | COMM_COMMAND Is CMD_REMOTE_CO_DONE .....                     | 116        |
| 3.3.4.4.10    | COMM_COMMAND Is CMD_ABORT_FETCH .....                        | 116        |
| 3.3.4.4.11    | COMM_COMMAND Is CMD_RETRY_FETCH.....                         | 116        |
| 3.3.4.4.12    | COMM_COMMAND Is CMD_VVJOIN_DONE .....                        | 116        |
| 3.3.4.4.13    | COMM_COMMAND Is CMD_UNJOIN_REMOTE .....                      | 116        |
| 3.3.4.5       | FrsRpcVerifyPromotionParent (Opnum 1) .....                  | 117        |
| 3.3.4.6       | Establishing a Connection Session.....                       | 117        |
| 3.3.5         | Timer Events.....  | 119        |
| 3.3.6         | Other Local Events.....                                      | 119        |
| 3.4           | PERFFRS Interface .....                                      | 119        |
| <b>4</b>      | <b>Protocol Examples .....</b>                               | <b>120</b> |
| 4.1           | Connection Establishment Sequence.....                       | 121        |
| 4.2           | Change Order Handling Sequence .....                         | 122        |
| 4.3           | COMM_PACKET .....  | 125        |
| 4.4           | SYSVOL Initial Sync.....                                     | 127        |
| 4.4.1         | Replica DC Sends Out CMD_NEED_JOIN to PDC.....               | 129        |
| 4.4.2         | PDC Sends Out CMD_START_JOIN to Replica DC .....             | 131        |
| 4.4.3         | Replica DC Sends Out CMD_JOINING .....                       | 133        |

|          |  |            |
|----------|--|------------|
| 4.4.4    | PDC Sends Out CMD_JOINED .....   | 135        |
| 4.4.5    | PDC Sends Out CMD_REMOTE_CO .....  | 137        |
| 4.4.6    | Replica DC Sends Out CMD_SEND_STAGE .....                                | 141        |
| 4.4.7    | PDC Sends Out CMD_RECEIVING_STAGE.....                                   | 145        |
| 4.4.8    | Replica DC Sends Out CMD_REMOTE_CO_DONE.....                             | 150        |
| 4.4.9    | Once All Change Orders are Sent Out, PDC Sends Out CMD_VVJOIN_DONE ..... | 154        |
| 4.4.10   | Replica DC Sends Out CMD_UNJOIN_REMOTE .....                             | 156        |
| 4.5      | Normal Sync.....   | 157        |
| 4.5.1    | Copy a File.....   | 158        |
| 4.5.1.1  | Upstream Partner Sends Out CMD_REMOTE_CO .....                           | 158        |
| 4.5.1.2  | Downstream Partner Sends Out CMD_SEND_STAGE 1 .....                      | 160        |
| 4.5.1.3  | Upstream Partner Sends Out CMD_RECEIVING_STAGE 1 .....                   | 164        |
| 4.5.1.4  | Downstream Partner Sends Out CMD_SEND_STAGE 2 .....                      | 168        |
| 4.5.1.5  | Upstream Partner Sends Out CMD_RECEIVING_STAGE 2 .....                   | 171        |
| 4.5.1.6  | Downstream Partner Sends Out CMD_REMOTE_CO_DONE .....                    | 174        |
| 4.5.2    | Rename a File .....  | 177        |
| 4.5.2.1  | Upstream Partner Sends Out CMD_REMOTE_CO .....                           | 177        |
| 4.5.2.2  | Downstream Partner Sends Out CMD_SEND_STAGE .....                        | 180        |
| 4.5.2.3  | Upstream Partner Sends Out CMD_RECEIVING_STAGE .....                     | 183        |
| 4.5.2.4  | Downstream Partner Sends Out CMD_REMOTE_CO_DONE .....                    | 188        |
| 4.5.3    | Remove a File .....  | 191        |
| 4.5.3.1  | Upstream Partner Sends Out CMD_REMOTE_CO .....                           | 191        |
| 4.5.3.2  | Downstream Partner Sends Out CMD_REMOTE_CO_DONE .....                    | 194        |
| 4.5.4    | Copy an Empty Folder .....   | 197        |
| 4.5.4.1  | Upstream Partner Sends Out CMD_REMOTE_CO .....                           | 197        |
| 4.5.4.2  | Downstream Partner Sends Out CMD_SEND_STAGE .....                        | 200        |
| 4.5.4.3  | Upstream Partner Sends Out CMD_RECEIVING_STAGE .....                     | 203        |
| 4.5.4.4  | Downstream Partner Sends Out CMD_REMOTE_CO_DONE .....                    | 208        |
| 4.5.5    | Remove a Folder .....  | 211        |
| 4.5.5.1  | Upstream Partner Sends Out CMD_REMOTE_CO .....                           | 211        |
| 4.5.5.2  | Downstream Partner Sends Out CMD_REMOTE_CO_DONE .....                    | 214        |
| 4.5.6    | Rename an Empty Folder.....  | 217        |
| 4.5.6.1  | Upstream Partner Sends Out CMD_REMOTE_CO .....                           | 217        |
| 4.5.6.2  | Downstream Partner Sends Out CMD_SEND_STAGE .....                        | 220        |
| 4.5.6.3  | Upstream Partner Sends Out CMD_RECEIVING_STAGE .....                     | 223        |
| 4.5.6.4  | Downstream Partner Sends Out CMD_REMOTE_CO_DONE .....                    | 228        |
| <b>5</b> | <b>Security.....</b>   | <b>232</b> |
| 5.1      | Security Considerations for Implementers .....                           | 232        |
| 5.2      | Index of Security Parameters .....                                       | 232        |
| <b>6</b> | <b>Appendix A: Full IDL.....</b>   | <b>233</b> |
| 6.1      | Appendix A.1: frsapi.idl.....  | 233        |
| 6.2      | Appendix A.2: frsrpc.idl .....   | 234        |
| <b>7</b> | <b>Appendix B: Product Behavior .....</b>                                | <b>236</b> |
| <b>8</b> | <b>Change Tracking.....</b>  | <b>245</b> |
| <b>9</b> | <b>Index.....</b>  | <b>246</b> |

# 1 Introduction

This is a protocol document specifying the File Replication Service (FRS) Protocol. The FRS Protocol is used to replicate files and folders among servers on the network. This protocol enables duplicate files and folders to be maintained on multiple servers. Microsoft operating systems use FRS to maintain duplicate copies of data files in system volume (SYSVOL) system folders on multiple domain controllers in a domain. Microsoft operating systems also use FRS to replicate data files among Distributed File System (DFS) shares.

This document provides detailed technical reference material for the Remote Procedure Call (RPC) interfaces, packet formats, and data structures required for interoperation using FRS.

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:

**Active Directory:** A general-purpose network directory service. Active Directory also refers to the Windows implementation of a directory service. Active Directory stores information about a variety of objects in the network. ~~Importantly, user~~User accounts, computer accounts, groups, and all related credential information used by the Windows implementation of Kerberos are stored in Active Directory. Active Directory is either deployed as Active Directory Domain Services (AD DS) or Active Directory Lightweight Directory Services (AD LDS). [MS-ADTS] describes both forms. For more information, see [MS-AUTHSOD] section 1.1.1.5.2, Lightweight Directory Access Protocol (LDAP) versions 2 and 3, Kerberos, and DNS.

**Active Directory object:** A set of directory objects that are used within Active Directory as defined in [MS-ADTS] section 3.1.1. An Active Directory object can be identified by a dsname. See also directory object.

**Active Directory replication:** The process by which the changes that are made to Active Directory objects on one domain controller (DC) are automatically synchronized with other DCs.

**Change Order:** A message that contains information about a file or folder that has changed on a replica. The change order is sent to the member's downstream partners. If the downstream partners accept the change, the partners request the associated staging file. After installing the changed file in their individual replica trees, the partners propagate the change order to their downstream partners.

**Connection Join (Join):** The process by which a connection session is established.

**Connection Session:** After FRS discovers a connection from Active Directory, FRS establishes a connection session with the remote connection partner based on the information provided by the connection object. The connection is called "joined" when a connection session is successfully established. This connection session is disconnected once the connection schedule is off (forbidding file replication on the connection).

**Dampening:** On receiving a remote change order, FRS must determine if the change order is already known to the local machine by using the version vector of the local machine. If the remote change order is known, FRS drops the change order and informs the upstream partner. This process is called "dampening" a change order.

**DFS-R:** A service that keeps DFS and SYSVOL folders in sync automatically. DFS-R is a state-based, multimaster replication system that supports replication scheduling and bandwidth throttling. This is a rewrite and new version of FRS. For more information, see [MS-FRS2].



**Directed Change Order:** A change order directed to a single downstream partner and produced when the partner is a Version Vector Join (VVJoin), such as during initial sync.

**Distributed File System (DFS):** A file system that logically groups physical shared folders located on different servers by transparently connecting them to one or more hierarchical namespaces. DFS also provides fault-tolerance and load-sharing capabilities.

**Distributed File System Replication (DFS-R):** A service that keeps DFS folders in sync automatically. DFS-R is a state-based, multi-master replication system that supports replication scheduling and bandwidth throttling. This is a rewrite and new version of the File Replication Service (FRS). For more information, see [MS-FRS2].

**domain:** A set of users and computers sharing a common namespace and management infrastructure. At least one computer member of the set must act as a domain controller (DC) and host a member list that identifies all members of the domain, as well as optionally hosting the Active Directory service. The domain controller provides authentication of members, creating a unit of trust for its members. Each domain has an identifier that is shared among its members. For more information, see [MS-AUTHSOD] section 1.1.1.5 and [MS-ADTS].

**domain controller (DC):** The service, running on a server, that implements Active Directory, or the server hosting this service. The service hosts the data store for objects and interoperates with other DCs to ensure that a local change to an object replicates correctly across all DCs. When Active Directory is operating as Active Directory Domain Services (AD DS), the DC contains full NC replicas of the configuration naming context (config NC), schema naming context (schema NC), and one of the domain NCs in its forest. If the AD DS DC is a global catalog server (GC server), it contains partial NC replicas of the remaining domain NCs in its forest. For more information, see [MS-AUTHSOD] section 1.1.1.5.2 and [MS-ADTS]. When Active Directory is operating as Active Directory Lightweight Directory Services (AD LDS), several AD LDS DCs can run on one server. When Active Directory is operating as AD DS, only one AD DS DC can run on one server. However, several AD LDS DCs can coexist with one AD DS DC on one server. The AD LDS DC contains full NC replicas of the config NC and the schema NC in its forest. The domain controller is the server side of Authentication Protocol Domain Support [MS-APDS].

**domain functional level:** A specification of functionality available in a domain. Must be less than or equal to the DC functional level of every domain controller (DC) that hosts a replica of the domain's naming context (NC). For information on defined levels, corresponding features, information on how the domain functional level is determined, and supported domain controllers, see [MS-ADTS] sections 6.1.4.2 and 6.1.4.3. When Active Directory is operating as Active Directory Lightweight Directory Services (AD LDS), domain functional level does not exist.

**Downstream Partner:** The partner that receives change orders, files, and folders.

**endpoint:** A network-specific address of a remote procedure call (RPC) server process for remote procedure calls. The actual name and type of the endpoint depends on the RPC protocol sequence that is being used. For example, for RPC over TCP (RPC Protocol Sequence `ncacn_ip_tcp`), an endpoint might be TCP port 1025. For RPC over Server Message Block (RPC Protocol Sequence `ncacn_np`), an endpoint might be the name of a named pipe. For more information, see [C706].

**endpoint mapper:** A service on a remote procedure call (RPC) server that maintains a database of dynamic endpoints and allows clients to map an interface/object UUID pair to a local dynamic endpoint. For more information, see [C706].

**Event Time:** See File Event Time.

**file:** An entity of data in the file system that a user can access and manage. A file must have a unique name in its directory. It consists of one or more streams of bytes that hold a set of related data, plus a set of attributes (also called properties) that describe the file or the data within the file. The creation time of a file is an example of a file attribute.

**file attribute:** A 32-bit bitmask containing information on a file's properties. For instance, 0x00000001 is used for the read-only attribute.

**File GUID:** An identifying property of a file or folder in a replica tree. FRS creates and manages file GUIDs, which, along with the file version number and file event time, are stored in the IDTable. Each file and folder stores its file GUID as part of its attributes; therefore, corresponding files and folders across all replica set members have the same file GUID.

**File Replication Service (FRS):** One of the services offered by a domain controller (DC), which is advertised through the Domain Controller Location protocol. The service being offered to clients is a replicated data storage volume that is associated with the default naming context (NC). The running or paused state of the FRS on a DC is available through protocols documented in [MS-ADTS] section 6.3.

**File Version Number:** A property of a file and folder in a replica tree that is incremented each time the file or folder is updated. The file version number is used to resolve concurrent updates originating from more than one member of the replica set. The version number is only incremented by the member that originated the file update. Other members that propagate the update do not change the version number.

**filter:** A setting that excludes subfolders (and their contents) or files from replication. There are two types of filters: file filters and folder filters.

**fully qualified domain name (FQDN):** An unambiguous domain name that gives an absolute location in the Domain Name System's (DNS) hierarchy tree, as defined in [RFC1035] section 3.1 and [RFC2181] section 11.

**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).

**IDTable:** A table of FRS state information that contains an entry with version and identity information for each file and folder in the replica tree. It is used to keep track of all files in the replica set and their histories.

**inbound:** The network traffic flowing from the client to the server.

**Inbound Connection:** For a given replica member, a component of the NT File Replication Service (NTFRS) member object in Active Directory that identifies upstream partners. An inbound connection exists for each upstream partner.

**inbound log:** A queue that stores pending change orders to be processed. As entries in the queue are processed, acknowledgments are sent to the inbound partners.

**Inbound Log (InLog):** A queue that stores pending change orders to be processed. As entries in the queue are processed, acknowledgments are sent to the upstream partners.

**Initial Sync:** The process a new member to the replica set before it is allowed to synchronize with its downstream partners. It is also called VVJoin.

**Install (File or Folder):** A process by which FRS applies a change order to the local file system to restore the file or folder as it is in the upstream partner. If the change order is for a deletion, the file or folder in the local file system is deleted (staging file is not needed). If the change order is for a renaming, the file or folder in the local file system is renamed (staging file is needed). If the change order is for a copying or creation, the file or folder is copied or created (staging file is needed). Installing a file or folder may fail if the file or folder is already opened

by another process. If the installation failed, FRS retries installing the file or folder at a later time.

**Interface Definition Language (IDL):** The International Standards Organization (ISO) standard language for specifying the interface for remote procedure calls. For more information, see [C706] section 4.

**join:** A combination of data from multiple sources that is based upon a common attribute or join condition.

**Local Change Order:** A change order that is created because of a change to a file or folder on the local server. The local server becomes the originator of the change order and constructs a staging file.

**MD5 hash:** A hashing algorithm, as described in [RFC1321], that was developed by RSA Data Security, Inc. An MD5 hash is used by the File Replication Service (FRS) to verify that a file on each replica member is identical.

**NetBIOS:** A particular network transport that is part of the LAN Manager protocol suite. NetBIOS uses a broadcast communication style that was applicable to early segmented local area networks. A protocol family including name resolution, datagram, and connection services. For more information, see [RFC1001] and [RFC1002].

**Normal Sync:** The synchronization among replicas after initial sync is done.

**NT file system (NTFS):** A proprietary Microsoft file system. For more information, see [MSFT-NTFS].

**NTFRS Member:** An object of class type nTFRSMember. Each NTFRS member object (class nTFRSMember) corresponds to a computer that is part of a replica set (see section NTFRS Member Object as specified in section 2.3.1.3).

**objectGUID:** The attribute on an Active Directory object whose value is a GUID that uniquely identifies the object. The GUID value of an object's objectGUID is assigned when the object was created and is immutable thereafter. The integrity of object references between NCs and of replication depends on the integrity of the objectGUID attribute. For a description of the general concept of an "object", see [MS-ADTS] section 1. For more detailed information see [MS-ADTS] section 3.1.1.1.3.

**opnum:** An operation number or numeric identifier that is used to identify a specific remote procedure call (RPC) method or a method in an interface. For more information, see [C706] section 12.5.2.12 or [MS-RPCE].

**Originator GUID:** A GUID that is associated with each replica member. All change orders produced by a given replica member carry the replica member's originator GUID, which is saved in the IDTable. The originator GUID is not the same as the member GUID, which is the objectGuid of the NTFRS member object in Active Directory. For more information, see [MS-ADTS] section 3.1.1.1.3.

**outbound:** Network traffic flowing from the server to the client.

**Outbound Connection:** For a given replica member, a component of the NTFRS member object in Active Directory that identifies downstream partners. An outbound connection exists for each downstream partner.

**Outbound Log (OutLog):** A table in the FRS database that stores pending change orders to be sent to downstream partners. The changes can originate locally or come from an upstream partner. These change orders are eventually sent to all outbound replica partners.

**Parent GUID:** The GUID of the parent folder that contains a particular file or folder in the replica tree.

**partner:** A computer connected to a local computer through either inbound or outbound connections.

**primary domain controller (PDC):** A domain controller (DC) designated to track changes made to the accounts of all computers on a domain. It is the only computer to receive these changes directly, and is specialized so as to ensure consistency and to eliminate the potential for conflicting entries in the Active Directory database. A domain has only one PDC.

**read-only domain controller (RODC):** A domain controller (DC) that does not accept originating updates. Additionally, an RODC does not perform outbound replication. An RODC cannot be the primary domain controller (PDC) for its domain.

**Remote Change Order:** A change order received from an inbound (or upstream) partner that originated elsewhere in the replica set.

**remote procedure call (RPC):** A context-dependent term commonly overloaded with three meanings. Note that much of the industry literature concerning RPC technologies uses this term interchangeably for any of the three meanings. Following are the three definitions: (\*) The runtime environment providing remote procedure call facilities. The preferred usage for this meaning is "RPC runtime". (\*) The pattern of request and response message exchange between two parties (typically, a client and a server). The preferred usage for this meaning is "RPC exchange". (\*) A single message from an exchange as defined in the previous definition. The preferred usage for this term is "RPC message". For more information about RPC, see [C706].

**Replica Member (FRS Replica):** A member of a replica set. Replica contains machine-specific information.

**replica set:** In File Replication Service (FRS), the replication of files and directories according to a predefined topology and schedule on a specific folder. The topology and schedule are collectively called a replica set. A replica set contains a set of replicas, one for each machine that participates in replication.

**replica tree:** The local replica root folder together with all files and directories underneath it, which usually is saved as a tree structure in the file system.

**Replica Tree Root:** The folder whose "children" (that is, files and folders) are replicated.

**Retry Change Order:** A change order that is in some state of completion but has been blocked for some reason and must be retried later.

**Schedule:** The frequency at which FRS replicates data under replica tree root.

**scheduled:** A status that is applied to a list item or document that specifies a time when the item or document will be published or unpublished.

**security principal name (SPN):** The name that identifies a security principal (for example, machinename\$@domainname for a machine joined to a domain or username@domainname for a user). Domainname is resolved using the Domain Name System (DNS).

**security provider:** A pluggable security module that is specified by the protocol layer above the remote procedure call (RPC) layer, and will cause the RPC layer to use this module to secure messages in a communication session with the server. The security provider is sometimes referred to as an authentication service. For more information, see [C706] and [MS-RPCE].

**staging file:** The backup of the changed file or folder. It encapsulates the data and attributes associated with a replicated file or folder. By creating the staging file, File Replication Service (FRS) ensures that file data can be supplied to partners regardless of any activity that might

prevent access to the original file. The staging files can be compressed to save disk space and network bandwidth during replication.

**system volume (SYSVOL):** A shared directory that stores the server copy of the domain's public files that must be shared for common access and replication throughout a domain.

**unique identifier (UID):** A pair consisting of a GUID and a version sequence number to identify each resource uniquely. The UID is used to track the object for its entire lifetime through any number of times that the object is modified or renamed.

**universally unique identifier (UUID):** A 128-bit value. UUIDs can be used for multiple purposes, from tagging objects with an extremely short lifetime, to reliably identifying very persistent objects in cross-process communication such as client and server interfaces, manager entry-point vectors, and RPC objects. UUIDs are highly likely to be unique. UUIDs are also known as globally unique identifiers (GUIDs) and these terms are used interchangeably in the Microsoft protocol technical documents (TDs). Interchanging the usage of these terms does not imply or require a specific algorithm or mechanism to generate the UUID. 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 UUID.

**Upstream Partner:** The partner that sends out change orders, files, and folders.

**version sequence number (VSN):** A 64-bit unsigned number. Version sequence numbers are assigned to global version sequence numbers as part of file metadata in monotonic increasing order.

**Version Vector Join (VVJoin):** The process in which a downstream partner joins with an upstream partner for the first time. Also called initial sync. This process is defined in section [MS-FRS1] section 3.3.4.6.

**Volatile Connection:** An inbound connection created for the initial sync for a system volume (SYSVOL) replica set. After the initial sync is done, the volatile connection is destroyed. Volatile connections are not represented in Active Directory.

**volume sequence number (VSN) (for file replication service):** A unique sequence number assigned to a change order to order the event sequence in a replica. It is a monotonically increasing sequence number assigned to each change that originates on a given replica member. If one change order has a smaller volume sequence number (VSN) than another change order, the change that the first change order represents occurs before the change that the second change order represents.

**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 dohelp@microsoft.com. We will assist you in finding the relevant information.

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

[MS-ADA1] Microsoft Corporation, "Active Directory Schema Attributes A-L".

[MS-ADA2] Microsoft Corporation, "Active Directory Schema Attributes M".

[MS-ADA3] Microsoft Corporation, "Active Directory Schema Attributes N-Z".

[MS-ADLS] Microsoft Corporation, "Active Directory Lightweight Directory Services Schema".

[MS-ADSC] Microsoft Corporation, "Active Directory Schema Classes".

[MS-ADTS] Microsoft Corporation, "Active Directory Technical Specification".

[MS-BKUP] Microsoft Corporation, "Microsoft NT Backup File Structure".

[MS-DFSNM] Microsoft Corporation, "Distributed File System (DFS): Namespace Management Protocol".

[MS-DFSRH] Microsoft Corporation, "DFS Replication Helper Protocol".

[MS-DRSR] Microsoft Corporation, "Directory Replication Service (DRS) Remote Protocol".

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

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

[MS-FRS2] Microsoft Corporation, "Distributed File System Replication Protocol".

[MS-FSCC] Microsoft Corporation, "File System Control Codes".

[MS-RPCE] Microsoft Corporation, "Remote Procedure Call Protocol Extensions".

[MS-RRP] Microsoft Corporation, "Windows Remote Registry Protocol".

[MS-XCA] Microsoft Corporation, "Xpress Compression Algorithm".

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992, <http://www.ietf.org/rfc/rfc1321.txt>

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

### 1.2.2 Informative References

[SPN] Microsoft Corporation, "Service-Principal-Name Attribute", <http://msdn.microsoft.com/en-us/library/ms679785.aspx>

## 1.3 Overview

The File Replication Service (FRS) Protocol is a multimaster replication protocol that is used to replicate files and folders across one or more members in an Active Directory domain. It works to keep copies of a file system tree up to date on all members of a replication group, while allowing any member of the group to change the contents at any time. A particular file system tree being replicated, along with the set of machines to which it is replicated, and the topology of connections between those machines used for replication, is known as a replica set.

The topology of a replica set is a directed graph over the machines in the set. Because the graph is directed, data only flows in one direction on any given connection. All machines in a replica set participate as a client on some connections and a server on others. As the machines in a replica set update the contents of the replicated folder, they are responsible for generating change orders that

propagate around the FRS topology. This causes the other members of the replica set to be aware of and (absent a conflict) replicate the update.

Each machine in a replication set keeps a volume sequence number (VSN) that is incremented each time it generates a change order. Each member of a replica set keeps track of the highest VSN that it knows about for each member of the replica set; together they are known as a version vector. By sending its version vector to its upstream partners in the replication topology, the upstream partners can efficiently determine what changes need to be sent and what changes are already known, and send only the appropriate set of change orders back to the downstream partner. For more information on version vectors see section 3.1.1.11.

On a given machine, FRS learns about all replica sets that it is part of, along with all its immediate partners, through a set of Active Directory objects. Replication topology is defined by two types of objects: Member objects represent a given participant, and connection objects that connect two endpoint member objects and define the direction of data flow along with the replication schedule.

FRS detects changes made to any file or folder underneath the replica tree root. Details of a given local change are captured in a change order construct. A change order represents an action that took place on the local file system, such as file write, creation, deletion, or rename. In addition, FRS maintains a single ID record for every file or folder underneath the replica tree root in the file system. The ID record provides the information FRS needs to locate the file on the file system. The ID record also stores any extra properties for the resource, such as file attributes.

A new FRS participant goes through a process called initial sync. This process creates the initial content on the new member by requesting all the data from upstream partners. New participants cannot replicate local changes until the initial sync is concluded.

At sync time, which is defined by the connection schedule, FRS establishes a connection with its upstream partner through the remote procedure call (RPC) interface exposed by every running FRS instance (one FRS instance per server). The connection is directed, so changes flow from the upstream partner to the downstream partner. FRS receives the version vector from its downstream partner in a process called Version Vector Join (or VVJoin), during which the upstream partner determines the changes that it needs to send to the downstream partner. For every such change, the upstream partner passes an appropriate change order to the downstream partner. The downstream partner inspects every change order it receives and decides, based on its local changes, to accept or reject the change. Typically, a change order is rejected if the local version supersedes the remote version of the resource. On accepting the change order, the downstream partner fetches the resource via one or more stage packets that carry the data as part of their payload in `FrsRpcSendCommPkt` method requests. Large files are partitioned into several stage packets that are serialized so that the downstream partner can reconstruct the file after receiving all the pieces.

File contents are marshaled before transfer over the wire to capture file system-specific metadata along with file data in one binary stream. The marshaled representation of a file is known as a staging file. The receiving partner has to be able to unmarshal the file at its end prior to placing it in the target location on the file system. The replicated file can also be compressed when it is marshaled to save bandwidth.

If two or more users are creating files with the same file name on different replica set members, these files will have name collisions with each other as they replicate to other members. Each of these created files is a distinct object with unique content, but only one can be kept. FRS detects that a name collision has occurred when the second file is replicated to a member after a previous file has arrived. FRS then performs the last-writer-wins reconciliation between the two distinct objects. The loser gets deleted, and the delete is propagated out to the other members. The winner keeps the name and gets installed on the member.

For folders, things are a bit different because they can have files and folders underneath them. In this case, FRS again detects the name collision when the second folder is replicated to a member, and it performs last-writer-wins reconciliation; except, in this case, the winner gets a new non-conflicting file

name (referred to as a morphed name), and the loser gets to keep the original folder name. The rename is replicated out so all copies of the renamed folder object get the same new name.

FRS supports four types of replica sets:

- FRS\_RSTYPE\_ENTERPRISE\_SYSVOL (1)
- FRS\_RSTYPE\_DOMAIN\_SYSVOL (2)
- FRS\_RSTYPE\_DFS (3)
- FRS\_RSTYPE\_OTHER (4)

FRS\_RSTYPE\_ENTERPRISE\_SYSVOL and FRS\_RSTYPE\_DOMAIN\_SYSVOL are used for SYSVOL replication.

FRS\_RSTYPE\_DFS is used for DFS replication. FRS\_RSTYPE\_OTHER is used only for testing.

The SYSVOL replica set is an FRS Replica set that has all the DCs in the domain as its members. It is created by default when a new domain is created. Every DC that is added to the domain is automatically joined as a member of this replica set. The SYSVOL replica set is mainly responsible for replicating policy data between the domain controllers.

FRS exposes two sets of RPC interfaces:

- Communication Interface—Exposes functions to implement the FRS replication protocol.
- Programming Interface—Exposes functions to implement administrative and monitoring tasks.

Each file or folder is assigned a GUID when it is first added to a replica set. All replicas in the replica set use the same GUID to refer to the file or folder. <1>

## 1.4 Relationship to Other Protocols

The FRS Protocol relies directly on RPC, as specified in [C706] and [MS-RPCE] (for message transport), and Remote Registry Protocol, as specified in [MS-RRP] (for remote performance monitoring). It relies indirectly on the Directory Replication Service (DRS) Remote Protocol to replicate its Active Directory objects, as specified in [MS-DRSR].

FRS is a deprecated protocol for DFS replica sets on Windows Server 2008 R2 operating system. The Distributed File System: Replication (DFS-R) Protocol (as specified in [MS-FRS2]) is used to replicate DFS/SYSVOL replica sets when the DFS-Replication Protocol is available. A DFS replica set is configured for replication over either this protocol or DFS-R, but not both. Functional levels are an extension of the mixed/native mode concept introduced in Windows 2000 operating system to activate new Active Directory features after all the domain controllers in the domain or forest are running the Windows Server 2003 operating system or Windows Server 2008 operating system. When a computer running Windows Server 2003 or Windows Server 2008 is installed and promoted to a domain controller, new Active Directory features are activated by the Windows Server 2003 operating system or Windows Server 2008 operating system over its Windows 2000 counterparts, or by the Windows Server 2008 operating system over its Windows Server 2003 counterparts. Additional Active Directory features are available when all domain controllers in a domain or forest are running Windows Server 2003, and the administrator activates the corresponding functional level in the domain or forest.

To activate the new domain features, all domain controllers in the domain have to be running Windows Server 2003 or Windows Server 2008 and the administrator raises the domain functional level to Windows Server 2003 or Windows Server 2008. The functional levels are separate from the revisions of the operating system in that users can upgrade to a new operating system (for example, Windows Server 2008) but not necessarily turn on the Windows Server 2008 functional level in the domains. The domain controllers for that domain will run the Windows Server 2008 operating system but display only the earlier domain functionality (for example, Windows Server 2003). However, a



domain controller in a Windows Server 2008 functional level domain has to be running the Windows Server 2008 operating system. Network clients can authenticate or access resources in the domain or forest without being affected by the Windows Server 2003 domain or forest functional levels. These levels only affect the way that domain controllers interact with each other.

For SYSVOL replica sets:

- If the domain functional level is at Windows Server 2008 level, the DFS-R Protocol is used to replicate SYSVOL replica sets, as specified in [MS-FRS2] and [MS-ADTS] (section 6.1.4.3). See section 2.3 for further details.
- If the domain functional level is Windows Server 2003 or earlier, the FRS Protocol is used to replicate SYSVOL replica sets, as specified in [MS-ADTS] (section 6.1.4.3).

For DFS replica sets, administrators can decide to create either NTFRS Active Directory objects or DFS-R Active Directory objects. A server might have some replica trees replicated using FRS and at the same time have other replica trees replicated using DFS-R. But, a replica tree cannot be replicated by both FRS and DFS-R at the same time.

- If FRS Active Directory objects exist (see section 2.3), FRS is used to replicate replica sets.
- If DFS-R objects exist, they are used to replicate replica sets, as described in [MS-DFSRH].

## 1.5 Prerequisites/Preconditions

FRS relies on the existence of a functional directory service. Machines using FRS optionally can be members of a domain and have appropriate credentials to access their partners. In particular, FRS relies on NTFRS member objects being present in the domain (see section 2.3).<2>

## 1.6 Applicability Statement

The FRS Protocol is appropriate for file and folder replication within a domain.

## 1.7 Versioning and Capability Negotiation

Versioning and capability negotiation in an RPC-based protocol is done by using the version number on the interface itself.

### 1.7.1 NtFrsApi

The version for this RPC interface is 1.1.

### 1.7.2 FRSRPC

The version for this RPC interface is 1.1.

FrsRpcSendCommPkt (see section 3.3.4.4) inside this interface has additional versioning negotiation.

FrsRpcSendCommPkt sends COMM\_PACKET (see section 2.2.3.5) packets between replication partners. Every COMM\_PACKET packet contains the following two fields:

- Major (major version number of FRS; always 0)
- Minor (minor version number of FRS; see section 2.2.3.5 for details on minor version values)

On receiving an FrsRpcSendCommPkt call from partners, FRS determines the minor version number from the packet received. This minor version number indicates the capabilities of this specific partner.

FRS cannot use any capabilities that are not available to this partner when sending packets to it through FrsRpcSendCommPkt RPC calls.<3>

## 1.8 Vendor-Extensible Fields

None.

## 1.9 Standards Assignments

FRS has no standards assignments. It uses the following UUIDs to identify its interfaces.

| Parameter                   | Value                                | Reference                    |
|-----------------------------|--------------------------------------|------------------------------|
| UUID for interface NtFrsApi | D049B186-814F-11D1-9A3C-00C04FC9B232 | As specified in section 3.2. |
| UUID for interface frsrpc   | F5CC59B4-4264-101A-8C59-08002B2F8426 | As specified in section 3.3. |

The security principal name (SPN) (for more information, see [SPN]) for FRS has the following value: NtFrs-88f5d2bd-b646-11d2-a6d3-00c04fc9b232/FQDN. The fully qualified domain name (FQDN) is the fully qualified domain name of the computer. See section 2.3.1.7 for the location of the computer's FQDN in Active Directory.

## 2 Messages

### 2.1 Transport

FRS is RPC-based, and both FRS interfaces MUST use the following protocol sequence:

NCACN\_IP\_TCP

By default, FRS replication over RPC occurs dynamically over an available port by using RPC endpoint mapper (also known as RPCSS) which is on port 135 (see [C706] part 4). This behavior can be overridden by a static port assignment so that FRS replication traffic passes through a specific port.

The protocol MUST use the underlying RPC protocol that is being used as a transport for FRS to retrieve the identity of the caller that made the method call.

FRS MUST have registered the following two security providers: <4>

- RPC\_C\_AUTHN\_GSS\_KERBEROS
- RPC\_C\_AUTHN\_GSS\_NEGOTIATE

### 2.2 Common Data Types

The following sections use RPC base types and definitions (as specified in [C706], [MS-RPCE], and [MS-RPCE] section 2.2.5.3.4.3) and system base types and definitions, as specified in [MS-DTYP].

This section defines a number of fields containing flags that are combined by using a logical OR operation. Except where otherwise specified, all undefined flags MUST be set to zero, and ignored on receipt.

#### 2.2.1 WCHAR and PWCHAR

```
typedef wchar_t WCHAR, *PWCHAR;
```

#### 2.2.2 NtFrsApi Common Data Types

The NtFrsApi interface uses the common data types specified in section 2.2. No additional common data types are required by this interface.

##### 2.2.2.1 NTFRSAPI\_INFO

```
typedef struct NTFRSAPI_INFO {
    ULONG Major;
    ULONG Minor;
    ULONG NtFrsMajor;
    ULONG NtFrsMinor;
    ULONG SizeInChars;
    ULONG Flags;
    ULONG TypeOfInfo;
    ULONG TotalChars;
    ULONG CharsToSkip;
    ULONG OffsetToLines;
    ULONG OffsetToFree;
    ULONG Lines[1];
} NTFRSAPI_INFO,
*PNTFRSAPI_INFO;
```

**Major:** A 32-bit, unsigned integer specifying the major version of the ntfrsapi.dll. This field MUST be zero (0).

**Minor:** A 32-bit, unsigned integer specifying the minor version of the ntfrsapi.dll.

**NtFrsMajor:** A 32-bit, unsigned integer specifying the major version of the NtFrs Service. This field MUST be zero (0).

**NtFrsMinor:** A 32-bit, unsigned integer specifying the minor version of the NtFrs Service.

**SizeInChars:** A 32-bit, unsigned integer specifying the size of this structure.

**Flags:** A 32-bit, unsigned integer specifying the returned Blob. It MUST be one of the following values.

| Value                                     | Meaning                                     |
|---|---|
| NTFRSAPI_INFO_FLAGS_VERSION<br>0x00000001 | Returned Version info in the Blob is valid. |
| NTFRSAPI_INFO_FLAGS_FULL<br>0x00000002    | Returned Blob is full.                      |

**TypeOfInfo:** A 32-bit unsigned integer specifying the Information type. It MUST be one of the following values:

| Value  | Meaning                             |
|--|-------------------------------------|
| NTFRSAPI_INFO_TYPE_VERSION<br>0x00000000     | Information on NtFrs Version.       |
| NTFRSAPI_INFO_TYPE_SETS<br>0x00000001        | Information on replica sets.        |
| NTFRSAPI_INFO_TYPE_DS<br>0x00000002          | Information on Directory Services.  |
| NTFRSAPI_INFO_TYPE_MEMORY<br>0x00000003      | Information on memory usage.        |
| NTFRSAPI_INFO_TYPE_IDTABLE<br>0x00000004     | Information on ID tables.           |
| NTFRSAPI_INFO_TYPE_OUTLOG<br>0x00000005      | Information on Outlog tables.       |
| NTFRSAPI_INFO_TYPE_INLOG<br>0x00000006       | Information on Inlog tables.        |
| NTFRSAPI_INFO_TYPE_THREADS<br>0x00000007     | Information on thread usage.        |
| NTFRSAPI_INFO_TYPE_STAGE<br>0x00000008       | Information on staging area.        |
| NTFRSAPI_INFO_TYPE_CONFIGTABLE<br>0x00000009 | Information on Configuration table. |

**TotalChars:** A 32-bit unsigned integer specifying the server stored context for this call. This context can be used in subsequent calls.

**CharsToSkip:** A 32-bit unsigned integer specifying number of characters to skip over in the next call.

**OffsetToLines:** A 32-bit unsigned integer specifying the starting offset of returned data.

**OffsetToFree:** A 32-bit unsigned integer specifying the offset to next free byte in this structure.

**Lines:** A 8-bit character specifying the starting value of the variable length data buffer returned.

### 2.2.3 FRSRPC Common Data Types

The FRSRPC interface uses the common data types defined in section 2.2. It also uses the data types defined within this section.

#### 2.2.3.1 GVSN

The GVSN structure, which contains a GUID and a VSN associated with a file that might require replication, MUST be formatted as follows:

```
typedef struct _GVSN {
    ULONGLONG VSN;
    GUID GUID;
} GVSN,
*PGVSN;
```

**VSN:** A 64-bit, unsigned integer containing the VSN for the originator GUID.

**GUID:** A field of type GUID that MUST contain the originator GUID.

#### 2.2.3.2 CHANGE\_ORDER\_COMMAND

The CHANGE\_ORDER\_COMMAND is referenced below in sections 2.2.3.6.20, 2.2.3.6.21, 2.2.3.6.22, and 2.2.3.10. It MUST be formatted as follows.

```
typedef struct _CHANGE_ORDER_COMMAND {
    ULONG SequenceNumber;
    ULONG Flags;
    ULONG IFlags;
    ULONG State;
    ULONG ContentCmd;
    ULONG LocationCmd;
    ULONG FileAttributes;
    ULONG FileVersionNumber;
    ULONG PartnerAckSeqNumber;
    ULONG Notused;
    ULONGLONG FileSize;
    ULONGLONG FileOffset;
    ULONGLONG FrsVsn;
    ULONGLONG FileUsn;
    ULONGLONG JrnlUsn;
    ULONGLONG JrnlFirstUsn;
    ULONG OriginalReplicaNum;
    ULONG NewReplicaNum;
    GUID ChangeOrderGuid;
    GUID OriginatorGuid;
    GUID FileGuid;
    GUID OldParentGuid;
    GUID NewParentGuid;
    GUID CxtionGuid;
```

```

ULONGLONG AckVersion;
ULONGLONG Spare2U11;
GUID Spare1Guid;
GUID Spare2Guid;
PWCHAR Spare1Wcs;
PWCHAR Spare2Wcs;
ULONG Extension;
PVOID Spare2Bin;
LARGE_INTEGER EventTime;
USHORT FileNameLength;
WCHAR FileName[MAX_PATH+1];
UCHAR Padding[4];
} CHANGE_ORDER_COMMAND;
*PCHANGE_ORDER_COMMAND;

```

**SequenceNumber:** A 32-bit, unsigned integer that specifies the sequence number in the change order command. It **MUST** be initialized to 0. The sequence number **MUST** be incremented with each change order that goes into the outbound log. The sequence number **MUST** be unique per replica set per machine.

**Flags:** A 32-bit, unsigned integer that **MUST** contain one or more change order flags, represented as the bitwise OR of zero or more of the following values (see section 3.3.4.4).

| Value                                    | Meaning  |
|--|--|
| CO_FLAG_ABORT_CO<br>0x00000001           | The change order is being aborted.   |
| CO_FLAG_VV_ACTIVATED<br>0x00000002       | A version vector activate request has been made (see section 3.3.4.4.6.2).   |
| CO_FLAG_CONTENT_CMD<br>0x00000004        | A valid content command.   |
| CO_FLAG_LOCATION_CMD<br>0x00000008       | A valid location command.  |
| CO_FLAG_ONLIST<br>0x00000010             | This change order is on a change order process list.   |
| CO_FLAG_LOCALCO<br>0x00000020            | This change order is locally generated.  |
| CO_FLAG_RETRY<br>0x00000040              | This change order <b>MUST</b> be retried (see section 3.1.1.10.2).   |
| CO_FLAG_INSTALL_INCOMPLETE<br>0x00000080 | The installation of the change order on the downstream partner is incomplete.  |
| CO_FLAG_OUT_OF_ORDER<br>0x00000200       | Do not check/update version vector.  |
| CO_FLAG_NEW_FILE<br>0x00000400           | The file or folder inside the change order is new. An IDTable record <b>MUST</b> be created for this file or folder. If the change order fails, then delete the IDTable record just created.         |
| CO_FLAG_CONTROL<br>0x00001000            | This is a control change order sent by a remote partner when an inbound connection starts up, so that the inbound log is scanned and any pending change orders for this connection are queued again. |
| CO_FLAG_DIRECTED_CO                      | This change order is directed to a single connection.  |

| Value                                   | Meaning  |
|---|--|
| 0x00002000                              |  |
| CO_FLAG_VVJOIN_TO_ORIG<br>0x00040000    | This change order is from a replica member identified by the originator GUID.                    |
| CO_FLAG_SKIP_ORIG_REC_CHK<br>0x00100000 | Skip the originator reconcile check.   |
| CO_FLAG_MOVEIN_GEN<br>0x00200000        | This change order was generated as part of a subfolder MOVEIN.                                   |
| CO_FLAG_MORPH_GEN_LEADER<br>0x00400000  | This is a MORPH_GEN leader, and it needs to refabricate the MORPH_GEN follower if it is retried. |
| CO_FLAG_JUST_OID_RESET<br>0x00800000    | Reset object ID (OID) back to its FRS-defined value.   |
| CO_FLAG_COMPRESSED_STAGE<br>0x01000000  | The staging file for this change order is compressed.  |
| CO_FLAG_SKIP_VV_UPDATE<br>0x02000000    | This change order can not update the version vector.   |

**IFlags:** A 32-bit, unsigned integer that contains additional flag values for the change order. The value of this field MUST be a bitwise OR of one or more of the values defined in the following table.

| Value                                   | Meaning  |
|---|--|
| CO_IFLAG_NONE<br>0x00000000             | This value is sent by the downstream replication partner if the change order is dampened.  |
| CO_IFLAG_VVRETIRE_EXEC<br>0x00000001    | Indicates that an FRS operation, such as a file rename or copy, is complete. This IFlag value is sent by a downstream replication partner if the change order is not dampened. |
| CO_IFLAG_CO_ABORT<br>0x00000002         | This value is sent by the downstream replication partner if change order processing needs to be aborted.   |
| CO_IFLAG_DIR_ENUM_PENDING<br>0x00000004 | This value is sent by the downstream replication partner if change order needs to enumerate its children as part of a sub-directory MoveIn.                                    |

**State:** A 32-bit, unsigned integer that indicates the change order process state.

| Value      | Meaning  |
|------------|--|
| 0x00000000 | This change order is entered in the log.                     |
| 0x00000001 | Allocating staging file space for a local change order.      |
| 0x00000002 | This local change order staging file copy has started.       |
| 0x00000003 | This local change order staging file is complete.            |
| 0x00000004 | Waiting to retry local change order staging file generation. |
| 0x00000005 | Allocated staging file space for a remote change order.      |

| Value      | Meaning   |
|------------|---|
| 0x00000006 | This remote change order staging file fetch has started.                      |
| 0x00000007 | This remote change order staging file fetch is complete.                      |
| 0x00000008 | Waiting to retry the fetch of the staging file for the remote change order.   |
| 0x00000009 | File install requested.   |
| 0x0000000A | File install has started.   |
| 0x0000000B | File install is complete.   |
| 0x0000000C | File install is waiting to try again.   |
| 0x0000000D | File install is retrying.   |
| 0x0000000E | File install rename is retrying.  |
| 0x0000000F | File install delete is retrying.  |
| 0x00000013 | This change order is being recycled to perform a folder enumeration.          |
| 0x00000014 | Request outbound propagation.   |
| 0x00000015 | Request was accepted and is now in the outbound log.                          |
| 0x00000016 | Database state update has started.  |
| 0x00000017 | Database state update has completed, and FRS is now freeing the change order. |
| 0x00000018 | This change order is being aborted.   |

**ContentCmd:** A 32-bit, unsigned integer that indicates the reasons for the change. The value of this field MUST be a bitwise OR of zero or more of the values defined in the following table.

| Value                                      | Meaning               |
|--|-----------------------|
| REASON_DATA_OVERWRITE<br>0x00000001        | Overwrite a file.     |
| REASON_DATA_EXTEND<br>0x00000002           | Extend a file.        |
| REASON_DATA_TRUNCATION<br>0x00000004       | Truncate the data.    |
| REASON_NAMED_DATA_OVERWRITE<br>0x00000010  | Overwrite named data. |
| REASON_NAMED_DATA_EXTEND<br>0x00000020     | Extend named data.    |
| REASON_NAMED_DATA_TRUNCATION<br>0x00000040 | Truncate named data.  |
| REASON_FILE_CREATE<br>0x00000100           | Create a file.        |
| REASON_FILE_DELETE                         | Delete a file.        |



| Value                                     | Meaning                            |
|---|------------------------------------|
| 0x00000200                                |                                    |
| REASON_EA_CHANGE<br>0x00000400            | Change the extended attribute.     |
| REASON_SECURITY_CHANGE<br>0x00000800      | Change the security type.          |
| REASON_RENAME_OLD_NAME<br>0x00001000      | The old name for rename operation. |
| REASON_RENAME_NEW_NAME<br>0x00002000      | The new name for rename operation. |
| REASON_BASIC_INFO_CHANGE<br>0x00008000    | Change the basic information.      |
| REASON_COMPRESSION_CHANGE<br>0x00020000   | Change the compression.            |
| REASON_ENCRYPTION_CHANGE<br>0x00040000    | Change the encryption.             |
| REASON_OBJECT_ID_CHANGE<br>0x00080000     | Change the object identifier.      |
| REASON_REPARSE_POINT_CHANGE<br>0x00100000 | Change the file reparse point.     |
| REASON_STREAM_CHANGE<br>0x00200000        | Change the file stream.            |

**LocationCmd:** File or folder location command. The location command specifies the movement of a file or folder in the replica tree. The location command also specifies whether the change order describes a change for a file or a folder. The location command MUST have the following structure.

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |  |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|--|
|   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |   |   |   |   |   | 2 |   |   |   |   |   |   |   |   |   |   |   |   | 3 |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |   |   |  |
| D | C | C | C | C | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F | F |   |  |

D = DirOrFile (1 bit)

C = File Location Command (4 bits)

F = Filler bits, not used (27 bits). MUST be 0. MUST be ignored on receipt.

**DirOrFile:** Bit that indicates whether the change is for a folder or for a file. The value MUST be one of the following:

| Value | Meaning                     |
|-------|-----------------------------|
| 0     | The change is for a file.   |
| 1     | The change is for a folder. |

**Command:** File location command MUST be one of the following values:

| Value                      | Meaning  |
|----------------------------|--|
| CO_LOCATION_CREATE<br>0x0  | Create a file or folder.   |
| CO_LOCATION_DELETE<br>0x1  | Delete a file or folder.   |
| CO_LOCATION_MOVEIN<br>0x2  | Rename a file or folder from a path outside the replica tree to a path inside the replica tree on the same volume. |
| CO_LOCATION_MOVEIN2<br>0x3 | Rename a file or folder into the replica set after a CO_LOCATION_MOVEOUT.  |
| CO_LOCATION_MOVEOUT<br>0x4 | Rename a file or folder from a path inside the replica tree to a path outside the replica tree on the same volume. |
| CO_LOCATION_MOVERS<br>0x5  | Rename a file or folder from one replica tree to another replica tree on the same volume.                          |
| CO_LOCATION_MOVEDIR<br>0x6 | Rename a file or folder from one folder to another folder in the same replica tree.                                |
| CO_LOCATION_NO_CMD<br>0x7  | No command.  |

**Filler:** These bits are not used and MUST be set to 0 and ignored by the receiver.

**FileAttributes:** A 32-bit, unsigned integer that indicates file attributes, as specified in [MS-FSCC] section 2.6.

**FileVersionNumber:** A 32-bit, unsigned integer that indicates the file version number.

**PartnerAckSeqNumber:** A 32-bit, unsigned integer that indicates the replication partner acknowledgment sequence number.

**Notused:** Four padding bytes to align the following field in memory. Not used. MUST be 0. MUST be ignored on receipt.

**FileSize:** A 64-bit, unsigned integer that indicates the file size in bytes. This is the original file size before staging and compression. It is used as a hint about how much staging space will be used on the downstream partner. The actual compressed staging file size is passed from upstream partner to downstream partner in the COMM\_FILE\_SIZE field in a COMM\_PACKET packet whose COMM\_COMMAND value is set to CMD\_RECEIVING\_STAGE (see section 3.3.4.4.7 and 3.3.4.4.8).

**FileOffset:** A 64-bit, unsigned integer that MUST be 0. Not used. MUST be ignored on receipt. (COMM\_FILE\_OFFSET inside COMM\_PACKET indicates the file offset in downloading the staging file. See section 2.2.3.6.16. This FileOffset field inside CHANGE\_ORDER\_COMMAND is not used.)

**FrsVsn:** A 64-bit, unsigned integer that indicates the originating server's volume sequence number.

**FileUsn:** A 64-bit, unsigned integer that indicates internal implementation-specific data on the current replica member. This value is meaningful only on the current replica member and has no meaning on any other replica member. Once the change order is sent to another replica member, the receiving partner MUST ignore this field.<5>

**JrnUsn:** A 64-bit, unsigned integer that indicates internal implementation-specific data on the current replica member. This value is meaningful only on the current replica member and has no

meaning on any other replica member. Once the change order is sent to another replica member, the receiving partner MUST ignore this field.<6>

**JrnlFirstUsn:** A 64-bit, unsigned integer that indicates internal implementation-specific data on the current replica member. This value is meaningful only on the current replica member and has no meaning on any other replica member. Once the change order is sent to another replica member, the receiving partner MUST ignore this field.<7>

**OriginalReplicaNum:** A 32-bit, unsigned integer that indicates internal implementation specific data on the current replica member. This value is meaningful only on the current replica member and has no meaning on any other replica member. Once the change order is sent to another replica member, the receiving partner MUST ignore this field.<8>

**NewReplicaNum:** A 32-bit, unsigned integer that indicates internal implementation specific data on the current replica member. This value is meaningful only on the current replica member and has no meaning on any other replica member. Once the change order is sent to another replica member, the receiving partner MUST ignore this field.<9>

**ChangeOrderGuid:** The identifying GUID of the change order.

**OriginatorGuid:** The GUID of the originating server.

**FileGuid:** The identifying GUID of the file.

**OldParentGuid:** The identifying GUID of the file's original parent folder.

**NewParentGuid:** The identifying GUID of the file's current parent folder.

**CxtionGuid:** The identifying GUID of the remote change order connection.

**AckVersion:** A 64-bit, unsigned integer that indicates the partner's acknowledgment version number. The downstream partner MUST preserve this value when receiving a change order from the upstream partner. The vendor can set it to any value based on the vendor's implementation or set it to 0 and ignore it.

**Spare2UI1:** An 8-byte field that is not used and MUST be set to 8 bytes of 0x00 and MUST be ignored by the receiver.

**Spare1Guid:** A 16-byte field that is not used and MUST be set to 16 bytes of 0x00 and MUST be ignored by the receiver.

**Spare2Guid:** A 16-byte field that is not used and MUST be set to 16 bytes of 0x00 and MUST be ignored by the receiver.

**Spare1Wcs:** A 4-byte field that is not used and MUST be set to 4 bytes of 0x00 and MUST be ignored by the receiver.

**Spare2Wcs:** A 4-byte field that is not used and MUST be set to 4 bytes of 0x00 and MUST be ignored by the receiver.

**Extension:** A 4-byte field that MUST be ignored.

**Spare2Bin:** A 4-byte field that is not used and MUST be set to 4 bytes of 0x00 and MUST be ignored by the receiver.

**EventTime:** USN journal event time that MUST be in the format of FILETIME, as specified in [MS-DTYP].

**FileNameLength:** A 16-bit, unsigned integer that indicates the length in bytes of **FileName** field. **FileNameLength** does not include the terminating 0 inside **FileName**.

**FileName:** MUST be a Unicode string that contains the NULL-terminated file name. It is not a whole path or a relative path, only the file name.

**Padding:** A 4-byte field (beyond the 522nd byte of **FileName**) that is not used and MUST be set to 4 bytes of 0x00 and MUST be ignored by the receiver.

### 2.2.3.3 CO\_RECORD\_EXTENSION\_WIN2K

The CO\_RECORD\_EXTENSION\_WIN2K structure MUST be formatted as follows:

```
typedef struct _CO_RECORD_EXTENSION_WIN2K {
    ULONG FieldSize;
    USHORT Major;
    USHORT OffsetCount;
    ULONG Offset;
    ULONG OffsetLast;
    DATA_EXTENSION_CHECKSUM DataChecksum;
} CO_RECORD_EXTENSION_WIN2K,
*PCO_RECORD_EXTENSION_WIN2K;
```

**FieldSize:** A 32-bit, unsigned integer that indicates the size of structure CO\_RECORD\_EXTENSION\_WIN2K. MUST be 0x00000028.

**Major:** Version of the CO\_RECORD\_EXTENSION\_WIN2K structure. MUST be 0.

**OffsetCount:** MUST be 0x0001.

**Offset:** MUST be 0x00000010.

**OffsetLast:** MUST be 0.

**DataChecksum:** DATA\_EXTENSION\_CHECKSUM structure, as specified in section 2.2.3.7.

### 2.2.3.4 CHANGE\_ORDER\_RECORD\_EXTENSION

The CHANGE\_ORDER\_RECORD\_EXTENSION structure MUST be formatted as follows.

```
typedef struct _CHANGE_ORDER_RECORD_EXTENSION {
    ULONG FieldSize;
    USHORT Major;
    USHORT OffsetCount;
    ULONG Offset[2];
    ULONG OffsetLast;
    ULONG Not used;
    DATA_EXTENSION_CHECKSUM DataChecksum;
    DATA_EXTENSION_RETRY_TIMEOUT DataRetryTimeout;
} CHANGE_ORDER_RECORD_EXTENSION,
*PCHANGE_ORDER_RECORD_EXTENSION;
```

**FieldSize:** Size, in bytes, of the CHANGE\_ORDER\_RECORD\_EXTENSION structure. MUST be 0x00000048, which is the size of CHANGE\_ORDER\_RECORD\_EXTENSION in bytes.

**Major:** A 16-bit, unsigned integer that specifies the version of the CHANGE\_ORDER\_RECORD\_EXTENSION structure. MUST be one of the following two values. <10>

| Value                                       | Meaning                                     |
|---|---|
| CO_RECORD_EXTENSION_VERSION_WIN2K<br>0x0000 | Version 0 of CHANGE_ORDER_RECORD_EXTENSION. |

| Value                                   | Meaning                                     |
|---|---|
| CO_RECORD_EXTENSION_VERSION_1<br>0x0001 | Version 1 of CHANGE_ORDER_RECORD_EXTENSION. |

**OffsetCount:** MUST be 0x0002, which is the number of element in array Offset.

**Offset:** Offset[0] is the offset of the DataChecksum member from the beginning of the CHANGE\_ORDER\_RECORD\_EXTENSION structure. MUST be 0x00000018.

**OffsetLast:** MUST be 0.

**Not used:** Four padding bytes to align the following structure in memory. Not used. MUST be 0. MUST be ignored on receipt.

**DataChecksum:** DATA\_EXTENSION\_CHECKSUM structure, as specified in section 2.2.3.7.

**DataRetryTimeout:** DATA\_EXTENSION\_RETRY\_TIMEOUT structure, as specified in section 2.2.3.9.

### 2.2.3.5 COMM\_PACKET and PCOMM\_PACKET

The COMM\_PACKET is the primary message used to accomplish file replication through FRS. It MUST be transmitted as the payload of the FrsRpcSendCommPkt method. By successively invoking FrsRpcSendCommPkt, a computer participates in a synchronous request/response-based conversation with another computer that results in replication of data files. On receiving an FrsRpcSendCommPkt call from another computer, FRS responds based on the nature of the request inside the COMM\_PACKET data received (see section 3.3.4.4).

A PCOMM\_PACKET is a pointer to a COMM\_PACKET.

The COMM\_PACKET contains information about the partner, the replication command, and data specific to that replication command, plus hash fields to verify the data's integrity. This information is contained in a series of elements within each COMM\_PACKET. There MUST be no padding between each element unless it is called out in the specification.

Twelve types of COMM\_PACKETS, each representing a different replication command, are sent over the wire. The replication command for the packet is indicated by the data field in the COMM\_COMMAND element.

Every packet, regardless of replication command type, MUST begin with the following eight elements. There MUST be no padding between each element unless it is called out in the specification.

COMM\_BOP: Beginning-of-packet marker, as specified in section 2.2.3.6.1.

COMM\_COMMAND: The replication command, as specified in section 2.2.3.6.2.

COMM\_TO: GUID and name of the member to receive this packet, as specified in section 2.2.3.6.3.

COMM\_FROM: GUID and name of the member that sent this packet, as specified in section 2.2.3.6.4.

COMM\_REPLICA: GUID and name that identifies the replica set to which this packet applies, as specified in section 2.2.3.6.5.

COMM\_CXTION: GUID and name of the connection over which this packet is transmitted, as specified in section 2.2.3.6.6.

COMM\_JOIN\_GUID: GUID that identifies the last successful Join over the connection identified in the COMM\_CXTION element, as specified in section 2.2.3.6.7.

COMM\_LAST\_JOIN\_TIME: Time that the Join identified by the COMM\_JOIN\_GUID element completed, as specified in section 2.2.3.6.8.

In addition, every packet MUST end with the following element, which MUST appear in the packet immediately after the preceding element, with no padding between it and the preceding element.

COMM\_EOP: End-of-Packet (EOP) marker, as specified in section 2.2.3.6.23.

FRS MUST respond to zero or more replication command-specific elements that appear between the last element in the beginning sequence and the ending element. There MUST be no padding bytes before or after any elements in the packet, so the following text gives the bit-level layout of the fields. All of the COMM\_PACKET elements are defined in the following section.

The types of replication command packets are listed in the following table along with a brief description and the list of elements that are contained in the packet. Only the mandatory elements and the elements listed for that COMM\_COMMAND value in the following table MUST be present for a packet with a particular COMM\_COMMAND value. The mandatory elements are omitted from this table, but MUST be present in every replication command packet. FRS MUST respond to these requests, as specified in section 3.3.4.4.

| COMM_COMMAND value          | Extra sequence of elements in packet (no padding between elements)   | Meaning  |
|-----------------------------|--|--|
| CMD_REMOTE_CO (0x218)       | COMM_REMOTE_CO<br>COMM_CO_EXTENSION_2  | To send a remote change order to a downstream partner.   |
| CMD_RECEIVING_STAGE (0x238) | COMM_BLOCK<br>COMM_BLOCK_SIZE<br>COMM_FILE_SIZE<br>COMM_FILE_OFFSET<br>COMM_CO_GUID<br>COMM_CO_SEQUENCE_NUMBER   | To transmit staging file to a downstream partner.  |
| CMD_REMOTE_CO_DONE (0x250)  | COMM_BLOCK_SIZE<br>COMM_FILE_SIZE<br>COMM_FILE_OFFSET<br>COMM_GVSN<br>COMM_CO_GUID<br>COMM_CO_SEQUENCE_NUMBER<br>COMM_REMOTE_CO<br>COMM_CO_EXTENSION_2 | To inform the upstream partner that processing the remote change order is complete. This packet also MUST include the change order command that has completed.       |
| CMD_ABORT_FETCH (0x246)     | COMM_BLOCK_SIZE<br>COMM_FILE_SIZE<br>COMM_FILE_OFFSET<br>COMM_CO_GUID<br>COMM_CO_SEQUENCE_NUMBER   | To inform the downstream partner that the staged file data cannot be sent. This command tells the partner to abort the fetch operation on the staging file.          |
| CMD_RETRY_FETCH (0x244)     | COMM_BLOCK_SIZE<br>COMM_FILE_OFFSET<br>COMM_CO_GUID<br>COMM_CO_SEQUENCE_NUMBER   | To inform the downstream partner that the request for the staging file data cannot be fulfilled at this time and that the request SHOULD be retried at a later time. |
| CMD_NEED_JOIN (0x121)       |  | To inform the upstream partner that a Join operation is needed.  |

| COMM_COMMAND value        | Extra sequence of elements in packet (no padding between elements)  | Meaning  |
|---------------------------|---|--|
| CMD_START_JOIN (0x122)    |   | To inform the downstream partner that a Join operation is starting.  |
| CMD_JOINING (0x130)       | COMM_VVECTOR<br>COMM_JOIN_TIME<br>COMM_REPLICA_VERSION_GUID<br>COMM_COMPRESSION_GUID  | To send a version vector to an upstream partner.   |
| CMD_JOINED (0x128)        |   | To inform the downstream partner that Join is successful.  |
| CMD_UNJOIN_REMOTE (0x148) |   | On completion of a VVJ on a connection, this command causes the upstream partner to disconnect the connection so that it can be torn down. |
| CMD_VVJOIN_DONE (0x136)   |   | To inform the downstream partner that all change orders are sent out during Initial Sync.  |
| CMD_SEND_STAGE (0x228)    | COMM_BLOCK_SIZE<br>COMM_FILE_SIZE<br>COMM_FILE_OFFSET<br>COMM_CO_GUID<br>COMM_CO_SEQUENCE_NUMBER<br>COMM_REMOTE_CO<br>COMM_CO_EXTENSION_2 | To request staging data from an upstream partner.  |

The structure of the COMM\_PACKET MUST be as follows:

```
typedef struct {
    unsigned long Major;
    unsigned long Minor;
    unsigned long CsId;
    unsigned long MemLen;
    [range(0,262144)] unsigned long PktLen;
    unsigned long UpkLen;
    [size_is(PktLen)] unsigned char* Pkt;
    [ignore] void* DataName;
    [ignore] void* DataHandle;
} COMM_PACKET,
*PCOMM_PACKET;
```

**Major:** A 32-bit, unsigned integer indicating the major version number of the FRS. This field MUST be zero (0).

**Minor:** A 32-bit unsigned integer that specifies the minor version number of FRS. The minor version number of FRS MUST be one of the following.

| Value                            | Meaning                    |
|----------------------------------|----------------------------|
| NTFRS_COMM_MINOR_0<br>0x00000000 | This is the first version. |
| NTFRS_COMM_MINOR_1               | Supports MD5.              |

| Value                            | Meaning  |
|----------------------------------|--|
| 0x00000001                       |  |
| NTFRS_COMM_MINOR_2<br>0x00000002 | Supports trigger schedule (see section 3.3.2.1.1).   |
| NTFRS_COMM_MINOR_3<br>0x00000003 | Supports change order record extension (see section 2.2.3.6.21).   |
| NTFRS_COMM_MINOR_4<br>0x00000004 | Forces the replica number fields in a change order to be an unsigned long instead of an unsigned long PTR for 32–64 bit interoperability. Supports the use of the COMM_COMPRESSION_GUID. Supports COMM_CO_EXTENSION_2. |
| NTFRS_COMM_MINOR_5<br>0x00000005 | Supports change order extension COMM_CO_EXTENSION_2 when replication partner is running FRS minor version 4 or later.  |
| NTFRS_COMM_MINOR_6<br>0x00000006 | No new capabilities.   |
| NTFRS_COMM_MINOR_7<br>0x00000007 | No new capabilities.   |
| NTFRS_COMM_MINOR_8<br>0x00000008 | No new capabilities.   |
| NTFRS_COMM_MINOR_9<br>0x00000009 | No new capabilities.   |

Version 0.7 or earlier is deprecated and SHOULD be used only for backward compatibility.

Subsequent versions MUST subsume all capabilities of lower versions.

**CsId:** A 32-bit, unsigned integer that MUST be set to 1.

**MemLen:** A 32-bit, unsigned integer containing the size, in bytes, of **Pkt**. Used to determine the packet length.

**PktLen:** A 32-bit, unsigned integer containing the length, in bytes, of the packet's valid data portion. Not all bytes inside **Pkt** hold valid data. The valid data inside **Pkt** is from offset 0 to PktLen.

**UpkLen:** MUST be 0.

**Pkt:** A pointer to a buffer that MUST consist of a sequence of COMM\_PACKET element structures placed back to back in contiguous memory. These elements MUST be in a particular order; however, some elements can be omitted depending on the command that is being transmitted. The table earlier in this section specifies the elements that MUST exist in the **Pkt** buffer, for each supported COMM\_COMMAND value.

The following table shows the order of the elements in the **Pkt** buffer. The first column shows when the element is included in the table: A – Always; V – included when a version vector is transmitted; S – included when staging data is transmitted; C – included when a change order Command is transmitted. See the first table in section 2.2.3.5 to see what element types MUST appear for each communication packet. For more information on version vectors see section 3.1.1.11.

See sections 2.2.3.6 through 2.2.3.6.23 for the definitions of the packet structures associated with each element type.



| <b>Usage</b> | <b>Element type and value</b>       | <b>Element contents</b>  |
|--------------|-------------------------------------|--|
| A            | COMM_BOP<br>0x0001                  | Beginning of the packet. This element MUST be the first element in the packet.   |
| A            | COMM_COMMAND<br>0x0002              | Type of command sent in this packet. See the "Type of communication packets" table in section 2.2.3.5.   |
| A            | COMM_TO<br>0x0003                   | GUID and name of the member to receive this packet. This MUST refer to either local machine or partner.  |
| A            | COMM_FROM<br>0x0004                 | GUID and name of member that sent this packet. This MUST refer to either local machine or partner.   |
| A            | COMM_REPLICA<br>0x0005              | GUID and name that identifies the replica set to which this packet applies.  |
| A            | COMM_CXTION<br>0x0008               | GUID and name of the connection over which this packet is transmitted.   |
| A            | COMM_JOIN_GUID<br>0x0006            | GUID that identifies the last successful Join over the connection identified in the COMM_CXTION element.   |
| A            | COMM_LAST_JOIN_TIME<br>0x0012       | Time (see section 2.2.3.6.8) that the Join identified by the COMM_JOIN_GUID element completed. Packet element contains the time of the last Join by this connection.   |
| V            | COMM_VVECTOR<br>0x0007              | A COMM_VVECTOR element MUST be inserted into Pkt for each GVSN structure in the version vector. There MUST be one COMM_VVECTOR element for each partner.   |
| V            | COMM_JOIN_TIME<br>0x0011            | This element MUST be initialized to the FILETIME (as specified in [MS-DTYP]) when the connection is joined.  |
| V            | COMM_REPLICA_VERSION_GUID<br>0x0014 | GUID of the machine that generated the version vector.   |
| V            | COMM_COMPRESSION_GUID<br>0x0018     | A COMM_COMPRESSION_GUID element MUST be inserted for each compression algorithm that the sending member supports (see section 2.2.3.6.12). There can be more than one COMM_COMPRESSION_GUID element. There are no other GUID values used by FRS. |
| S            | COMM_BLOCK<br>0x0009                | Block of data from the staging file that is being transmitted.   |
| S            | COMM_BLOCK_SIZE<br>0x000A           | Size in bytes of the block of data in the COMM_BLOCK element.  |
| S            | COMM_FILE_SIZE<br>0x000B            | Size in bytes of the staging file being transmitted.   |
| S            | COMM_FILE_OFFSET<br>0x000C          | Offset from the beginning of the staging file to the beginning of the block in the COMM_BLOCK element.   |
| S            | COMM_GVSN<br>0x000E                 | GVSN structure for the file being transmitted.   |
| S            | COMM_CO_GUID                        | GUID that identifies the change order that caused the file to be   |

| Usage | Element type and value                                       | Element contents   |
|-------|--|--|
|       | 0x000F   | transmitted.   |
| S     | COMM_CO_SEQUENCE_NUMBER<br>0x0010                            | Sequence number starting at 1 to record the order of change order commands. It MUST be used for acknowledgment.  |
| C     | COMM_REMOTE_CO<br>0x000D                                     | Includes a remote change order command.  |
| C     | COMM_CO_EXT_WIN2K<br>0x0016<br>COMM_CO_EXTENSION_2<br>0x0017 | COMM_CO_EXT_WIN2K or a COMM_CO_EXTENSION_2 element will be in Pkt. If either partner is NTFRS_COMM_MINOR_3 or lower, the COMM_CO_EXT_WIN2K element will be used. Otherwise, the COMM_CO_EXTENSION_2 element will be used. It contains change order extensions. |
| A     | COMM_EOP (0x13)  | End-of-packet (EOP) element.   |

**DataName:** MUST be 0.

**DataHandle:** MUST be 0.

## 2.2.3.6 COMM\_PACKET Elements

### 2.2.3.6.1 COMM\_BOP

The structure of the COMM\_BOP packet element is as follows:

```
typedef struct _COMM_BOP {
    USHORT CommType;
    ULONG Length;
    ULONG Data;
} COMM_BOP,
*PCOMM_BOP;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0001.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x00000004.

**Data:** A 32-bit, unsigned integer that contains the data within this COMM\_PACKET element. MUST be set to 0x00000000.

### 2.2.3.6.2 COMM\_COMMAND

The COMM\_COMMAND element contains the communication packet type. The structure of the COMM\_COMMAND packet element is as follows:

```
typedef struct _COMM_COMMAND {
    USHORT CommType;
    ULONG Length;
    ULONG Data;
} COMM_COMMAND,
*PCOMM_COMMAND;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0002.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x00000004.

**Data:** A 32-bit, unsigned integer that indicates the communication packet type. The value of this field MUST be one of the following.

| Value                             | Meaning  |
|-----------------------------------|--|
| CMD_REMOTE_CO<br>0x00000218       | To send a remote change order to a downstream partner.   |
| CMD_RECEIVING_STAGE<br>0x00000238 | To transmit a staging file to a downstream partner.  |
| CMD_REMOTE_CO_DONE<br>0x00000250  | To inform the upstream partner that processing the remote change order is complete. This packet also includes the change order command that has completed.   |
| CMD_ABORT_FETCH<br>0x00000246     | To inform the downstream partner that the staging file data cannot be sent. This command tells the partner to abort the fetch operation on the staging file.   |
| CMD_RETRY_FETCH<br>0x00000244     | To inform the downstream partner that the request for the staging file data cannot be fulfilled at this time and that the request has to be retried at a later time.   |
| CMD_NEED_JOIN<br>0x00000121       | To inform the upstream partner that a Join operation is needed.  |
| CMD_START_JOIN<br>0x00000122      | To inform the downstream partner that a Join operation is starting.  |
| CMD_JOINING<br>0x00000130         | To send a version vector to an upstream partner.   |
| CMD_JOINED<br>0x00000128          | To inform the downstream partner that the Join is successful.  |
| CMD_UNJOIN_REMOTE<br>0x00000148   | To inform an upstream partner on completion of a version vector join (vvjoin) over a volatile connection. This command requests that the upstream partner disconnect the connection so that it can be torn down. |
| CMD_VVJOIN_DONE<br>0x00000136     | To inform the downstream partner that all change orders were sent out during initial sync.   |
| CMD_SEND_STAGE<br>0x00000228      | To request staging data from an upstream partner.  |

### 2.2.3.6.3 COMM\_TO

The structure of the COMM\_TO packet element is as follows:

```
typedef struct _COMM_TO {  
    USHORT CommType;  
    ULONG Length;  
    ULONG LengthGuid;
```

```

    GUID Guid;
    ULONG LengthName;
    wchar_t Name[];
} COMM_TO,
*PCOMM_TO;

```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0003.

**Length:** A 32-bit, unsigned integer that indicates the length, in bytes, of the COMM\_TO packet element data following the Length field.

**LengthGuid:** A 32-bit, unsigned integer indicates the length, in bytes, of the GUID field. MUST be 0x00000010.

**Guid:** A field of type GUID that indicates the GUID of the receiving partner.

**LengthName:** A 32-bit, unsigned integer that indicates the length, in bytes, of the **Name** field, including the NULL terminator.

**Name:** A Unicode UTF-16 string that identifies the receiving partner. MUST be a variable-length array that contains a NULL-terminated string (0x0000) of 16-bit (Unicode) characters.

#### 2.2.3.6.4 COMM\_FROM

The structure of the COMM\_FROM packet element is as follows:

```

typedef struct _COMM_FROM {
    USHORT CommType;
    ULONG Length;
    ULONG LengthGuid;
    GUID Guid;
    ULONG LengthName;
    wchar_t Name[];
} COMM_FROM,
*PCOMM_FROM;

```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0004.

**Length:** A 32-bit, unsigned integer that indicates the length, in bytes, of the data following the Length field.

**LengthGuid:** A 32-bit, unsigned integer that indicates the length, in bytes, of the GUID field. MUST be 0x00000010.

**Guid:** A field of type GUID that indicates the GUID of the sending partner.

**LengthName:** A 32-bit, unsigned integer that indicates the length, in bytes, of the **Name** field, including the NULL terminator.

**Name:** A Unicode UTF-16 string that identifies the sending partner. MUST be a variable-length array that contains a NULL (0x0000) terminated string of 16-bit (Unicode) characters.

#### 2.2.3.6.5 COMM\_REPLICA

The structure of the COMM\_REPLICA packet element is as follows.

```

typedef struct _COMM_REPLICA {
    USHORT CommType;

```

```

    ULONG Length;
    ULONG LengthGuid;
    GUID Guid;
    ULONG LengthName;
    wchar_t Name[];
} COMM_REPLICA,
*PCOMM_REPLICA;

```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0005.

**Length:** A 32-bit, unsigned integer that indicates the length, in bytes, of the data following the Length field.

**LengthGuid:** A 32-bit, unsigned integer that indicates the length, in bytes, of the GUID field. MUST be 0x00000010.

**Guid:** A field of type GUID that indicates the GUID of the receiving Partner.

**LengthName:** A 32-bit, unsigned integer that indicates the length, in bytes, of the **Name** field, including the NULL terminator.

**Name:** A Unicode UTF-16 string that identifies the affected replica set. MUST BE a variable-length array that contains a NULL (0x0000) terminated string of 16-bit (Unicode) characters. There are no other constraints on the format.

#### 2.2.3.6.6 COMM\_CXTION

The structure of the COMM\_CXTION packet element is as follows:

```

typedef struct _COMM_CXTION {
    USHORT CommType;
    ULONG Length;
    ULONG LengthGuid;
    GUID Guid;
    ULONG LengthName;
    wchar_t Name[];
} COMM_CXTION,
*PCOMM_CXTION;

```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0008.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field.

**LengthGuid:** A 32-bit, unsigned integer that indicates the length in bytes of the GUID field. MUST be 0x00000010.

**Guid:** A field of type GUID that contains the GUID of the connection.

**LengthName:** A 32-bit, unsigned integer that indicates the length in bytes of the **Name** field, including the NULL terminator.

**Name:** A Unicode UTF-16 string that identifies the connection. MUST be a variable-length array that contains a NULL (0x0000) terminated string of 16-bit (Unicode) characters. There are no other constraints on the format.

#### 2.2.3.6.7 COMM\_JOIN\_GUID

The structure of the COMM\_JOIN\_GUID packet element is as follows:

```
typedef struct _COMM_JOIN_GUID {
    USHORT CommType;
    ULONG Length;
    ULONG DataLength;
    GUID Data;
} COMM_JOIN_GUID,
*PCOMM_JOIN_GUID;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0006.

**Length:** A 32-bit, unsigned integer that indicates the length, in bytes, of the data following the Length field. MUST be 0x00000014.

**DataLength:** A 32-bit, unsigned integer that indicates the length, in bytes, of the GUID field. MUST be 0x00000010.

**Data:** A field of type GUID that specifies a GUID object that denotes the last Join on this connection. Each Join MUST be identified by a unique GUID.

#### 2.2.3.6.8 COMM\_LAST\_JOIN\_TIME

The structure of the COMM\_LAST\_JOIN\_TIME packet element is as follows:

```
typedef struct _COMM_LAST_JOIN_TIME {
    USHORT CommType;
    ULONG Length;
    ULONGLONG Data;
} COMM_LAST_JOIN_TIME,
*PCOMM_LAST_JOIN_TIME;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0012.

**Length:** A 32-bit, unsigned integer that indicates the length, in bytes, of the data following the Length field. MUST be 0x00000008.

**Data:** A 64-bit, unsigned integer that MUST indicate the Coordinated Universal Time (UTC) time of the last successful connection join. If the value of Data is 0x0000000000000001, the last join time is invalid. If this field contains a valid last join time, it MUST be in FILETIME format, as specified in [MS-DTYP].

#### 2.2.3.6.9 COMM\_VVECTOR

The structure of the COMM\_VVECTOR packet element is as follows:

```
typedef struct _COMM_VVECTOR {
    USHORT CommType;
    ULONG Length;
    ULONG DataLength;
    GVSN Data;
} COMM_VVECTOR,
*PCOMM_VVECTOR;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0007.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x0000001C.

**DataLength:** A 32-bit, unsigned integer that indicates the length in bytes of the data in the **Data** field. MUST be 0x00000018.

**Data:** The GVSN structure, as specified in section 2.2.3.1, being transmitted.

### 2.2.3.6.10 COMM\_JOIN\_TIME

The structure of the COMM\_JOIN\_TIME packet element is as follows:

```
typedef struct _COMM_JOIN_TIME {
    USHORT CommType;
    ULONG Length;
    ULONG DataLength;
    FILETIME Data;
} COMM_JOIN_TIME,
*PCOMM_JOIN_TIME;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0011.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x0000000C.

**DataLength:** A 32-bit, unsigned integer that indicates the length in bytes of the data in the **Data** field. MUST be 0x00000008.

**Data:** A FILETIME (as specified in [MS-DTYP]) structure that specifies the UTC time when the partner builds this communication packet.

### 2.2.3.6.11 COMM\_REPLICA\_VERSION\_GUID

The structure of the COMM\_REPLICA\_VERSION\_GUID packet element is as follows:

```
typedef struct _COMM_REPLICA_VERSION_GUID {
    USHORT CommType;
    ULONG Length;
    ULONG DataLength;
    GUID Data;
} COMM_REPLICA_VERSION_GUID,
*PCOMM_REPLICA_VERSION_GUID;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0014.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x00000014.

**DataLength:** A 32-bit, unsigned integer that indicates the length in bytes of the GUID field. MUST be 0x00000010.

**Data:** A field of type GUID that specifies a GUID object that represents the machine that provided the version vector on the last Join.

### 2.2.3.6.12 COMM\_COMPRESSION\_GUID

The structure of the COMM\_COMPRESSION\_GUID packet element is as follows:

```

typedef struct _COMM_COMPRESSION_GUID {
    USHORT CommType;
    ULONG Length;
    GUID Data;
} COMM_COMPRESSION_GUID,
*PCOMM_COMPRESSION_GUID;

```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0018.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x00000010.

**Data:** A field of type GUID that contains a GUID that represents a compression algorithm. This field MUST contain the GUID of a compression algorithm supported by the FRS implementation. The two machines exchanging FRS messages MUST agree on the compression algorithm represented by any GUID sent in this field. A GUID value of 00000000-0000-0000-0000-000000000000 indicates uncompressed data. A GUID value of 64d2f7d2-2695-436d-8830-8d3c58701e15 indicates LZNT1 compression, as specified in [MS-XCA] section 2.5.<11>

### 2.2.3.6.13 COMM\_BLOCK

The structure of the COMM\_BLOCK packet element is as follows:

```

typedef struct _COMM_BLOCK {
    USHORT CommType;
    ULONG Length;
    ULONG DataLength;
    UCHAR Data[];
} COMM_BLOCK,
*PCOMM_BLOCK;

```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0009.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be set to the sum of 4 and the length of the Data field.

**DataLength:** A 32-bit, unsigned integer that MUST indicate the length in bytes of the **Data** field.

**Data:** A binary large object (BLOB) that contains the block of staging file data that is being transmitted. <12>

### 2.2.3.6.14 COMM\_BLOCK\_SIZE

The structure of the COMM\_BLOCK\_SIZE packet element is as follows:

```

typedef struct _COMM_BLOCK_SIZE {
    USHORT CommType;
    ULONG Length;
    ULONGLONG Data;
} COMM_BLOCK_SIZE,
*PCOMM_BLOCK_SIZE;

```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x000A.



**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x00000008.

**Data:** A 64-bit, unsigned integer representing the size in bytes of the data in the COMM\_BLOCK element. If Data is 0, COMM\_BLOCK MUST NOT be included in the packet. When sending compressed data in the COMM\_BLOCK element, this field MUST contain the size of the compressed data.

#### 2.2.3.6.15 COMM\_FILE\_SIZE

The structure of the COMM\_FILE\_SIZE packet element is as follows.

```
typedef struct _COMM_FILE_SIZE {
    USHORT CommType;
    ULONG Length;
    ULONGLONG Data;
} COMM_FILE_SIZE,
*PCOMM_FILE_SIZE;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x000B.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x00000008.

**Data:** A 64-bit, unsigned integer that specifies the total size in bytes of the staged file.

#### 2.2.3.6.16 COMM\_FILE\_OFFSET

The structure of the COMM\_FILE\_OFFSET packet element is as follows.

```
typedef struct _COMM_FILE_OFFSET {
    USHORT CommType;
    ULONG Length;
    ULONGLONG Data;
} COMM_FILE_OFFSET,
*PCOMM_FILE_OFFSET;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x000C.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x00000008.

**Data:** A 64-bit, unsigned integer that specifies the offset of the data block in the staged file. Large staged files require multiple packets to transfer. The offset here is relative to the first byte of the staged file.

#### 2.2.3.6.17 COMM\_GVSN

The structure of the COMM\_GVSN packet element is as follows.

```
typedef struct _COMM_GVSN {
    USHORT CommType;
    ULONG Length;
    ULONG DataLength;
    GVSN Data;
} COMM_GVSN,
```

\*PCOMM\_GVSN;

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x000E.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x0000001C.

**DataLength:** A 32-bit, unsigned integer that indicates the length in bytes of the **Data** field. MUST be 0x00000018.

**Data:** MUST be a structure of type GVSN, as specified in section 2.2.3.1, for the file being transmitted.

### 2.2.3.6.18 COMM\_CO\_GUID

The structure of the COMM\_CO\_GUID packet element is as follows.

```
typedef struct _COMM_CO_GUID {
    USHORT CommType;
    ULONG Length;
    ULONG DataLength;
    GUID Data;
} COMM_CO_GUID,
*PCOMM_CO_GUID;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x000F.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x00000014.

**DataLength:** A 32-bit, unsigned integer that indicates the length in bytes of the GUID field. MUST be 0x00000010.

**Data:** MUST be a field of type GUID that contains the GUID that denotes the change order that is being processed.

### 2.2.3.6.19 COMM\_CO\_SEQUENCE\_NUMBER

The structure of the COMM\_CO\_SEQUENCE\_NUMBER packet element is as follows.

```
typedef struct _COMM_CO_SEQUENCE_NUMBER {
    USHORT CommType;
    ULONG Length;
    ULONG Data;
} COMM_CO_SEQUENCE_NUMBER,
*PCOMM_CO_SEQUENCE_NUMBER;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0010.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x00000004.

**Data:** A 32-bit, unsigned integer that MUST indicate the change order sequence number (see section 2.2.3.2). The sequence number MUST be incremented with each change order that goes into the outbound log. The sequence number MUST be a unique per replica set per machine.

### 2.2.3.6.20 COMM\_REMOTE\_CO

The structure of the COMM\_REMOTE\_CO packet element is as follows.

```
typedef struct _COMM_REMOTE_CO {
    USHORT CommType;
    ULONG Length;
    ULONG DataLength;
    CHANGE_ORDER_COMMAND Data;
} COMM_REMOTE_CO,
*PCOMM_REMOTE_CO;
```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x000D.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x0000031C.

**DataLength:** A 32-bit, unsigned integer that indicates the length in bytes of the **Data** field. MUST be 0x00000318.

**Data:** The CHANGE\_ORDER\_COMMAND message being transmitted.

The CHANGE\_ORDER\_COMMAND structure defines a file replication service (FRS) change order. See the definition of the CHANGE\_ORDER\_COMMAND in section 2.2.3.2.

### 2.2.3.6.21 COMM\_CO\_EXT\_WIN2K

The COMM\_CO\_EXT\_WIN2K element contains a CO\_RECORD\_EXTENSION\_WIN2K structure, which is an extension to the CHANGE\_ORDER\_COMMAND structure that is compatible with down-level clients (NTFRS\_COMM\_MINOR\_3). The structure of the COMM\_CO\_EXT\_WIN2K packet element is as follows.

```
typedef struct _COMM_CO_EXT_WIN2K {
    USHORT CommType;
    ULONG Length;
    ULONG DataLength;
    CO_RECORD_EXTENSION_WIN2K Data;
} COMM_CO_EXT_WIN2K,
*PCOMM_CO_EXT_WIN2K;
```

**CommType:** A 16-bit, unsigned integer that specifies the type of this COMM\_PACKET element. MUST be 0x0016.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x0000002C.

**DataLength:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the DataLength field. MUST be 0x00000028.

**Data:** The CO\_RECORD\_EXTENSION\_WIN2K structure being transmitted (see section 2.2.3.3).

### 2.2.3.6.22 COMM\_CO\_EXTENSION\_2

The COMM\_CO\_EXTENSION\_2 element MUST contain a CHANGE\_ORDER\_RECORD\_EXTENSION structure, which is an extension to the CHANGE\_ORDER\_COMMAND structure. The structure of the COMM\_CO\_EXTENSION\_2 packet element is as follows.

```
typedef struct _COMM_CO_EXTENSION_2 {
    USHORT CommType;
```

```

    ULONG Length;
    CHANGE_ORDER_RECORD_EXTENSION Data;
} COMM_CO_EXTENSION_2,
*PCOMM_CO_EXTENSION_2;

```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST set to 0x0017.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be set to the **FieldSize** member of the CHANGE\_ORDER\_RECORD\_EXTENSION structure, which is 0x00000048 (72 bytes).

**Data:** The data that is being transmitted in the form of a CHANGE\_ORDER\_RECORD\_EXTENSION (see section 2.2.3.4).

### 2.2.3.6.23 COMM\_EOP

The structure of the COMM\_EOP packet element is as follows.

```

typedef struct _COMM_EOP {
    USHORT CommType;
    ULONG Length;
    ULONG Data;
} COMM_EOP,
*PCOMM_EOP;

```

**CommType:** A 16-bit, unsigned integer that indicates the type of this COMM\_PACKET element. MUST be set to 0x0013.

**Length:** A 32-bit, unsigned integer that indicates the length in bytes of the data following the Length field. MUST be 0x00000004.

**Data:** A 32-bit, unsigned integer that indicates the end of the packet. MUST be set to 0xFFFFFFFF.

### 2.2.3.7 DATA\_EXTENSION\_CHECKSUM

The DATA\_EXTENSION\_CHECKSUM structure specifies an MD5 checksum. The structure of DATA\_EXTENSION\_CHECKSUM is as follows.

```

typedef struct _DATA_EXTENSION_CHECKSUM {
    DATA_EXTENSION_PREFIX Prefix;
    UCHAR Data[16];
} DATA_EXTENSION_CHECKSUM,
*PDATA_EXTENSION_CHECKSUM;

```

**Prefix:** Contains a DATA\_EXTENSION\_PREFIX structure, as specified in section 2.2.3.8 element. The **Size** member of the DATA\_EXTENSION\_PREFIX structure MUST be set to 0x18. The **Type** field of the DATA\_EXTENSION\_PREFIX structure MUST be set to 1 to indicate the use of MD5 hashing.

**Data:** MUST be a 128-bit MD5 digest of staging file and attributes, as specified in [RFC1321]. See section 2.2.3.10 for how the MD5 digest is constructed on a staging file and attributes.

### 2.2.3.8 DATA\_EXTENSION\_PREFIX

The DATA\_EXTENSION\_PREFIX structure contains a data component size and type. The structure of DATA\_EXTENSION\_PREFIX is as follows.

```
typedef struct _DATA_EXTENSION_PREFIX {
    ULONG Size;
    LONG Type;
} DATA_EXTENSION_PREFIX,
*PDATA_EXTENSION_PREFIX;
```

**Size:** A 32-bit, unsigned integer that specifies the size of the data component, including this prefix.

**Type:** A 32-bit, unsigned integer that specifies the type of the data component. It MUST be one of the following values.

| Value      | Meaning   |
|------------|---|
| 0x00000000 | Terminates a data extension record. MUST NOT be used. |
| 0x00000001 | Data checksum record using an MD5 hash.               |
| 0x00000002 | Data retry time-out record.                           |

### 2.2.3.9 DATA\_EXTENSION\_RETRY\_TIMEOUT

The DATA\_EXTENSION\_RETRY\_TIMEOUT structure specifies retries to a change order and the time of the first try. The structure of DATA\_EXTENSION\_RETRY\_TIMEOUT is as follows.

```
typedef struct _DATA_EXTENSION_RETRY_TIMEOUT {
    DATA_EXTENSION_PREFIX Prefix;
    DWORD Count;
    ULONG Notused;
    LONGLONG FirstTryTime;
} DATA_EXTENSION_RETRY_TIMEOUT,
*PDATA_EXTENSION_RETRY_TIMEOUT;
```

**Prefix:** MUST be the same as DATA\_EXTENSION\_PREFIX. Size MUST be set to 0x18. Type MUST be set to 2.

**Count:** A 32-bit, unsigned integer that specifies the number of retries of this change order to date.

**Notused:** Four padding bytes to align the following structure in memory. Not used. MUST be 0. MUST be ignored on receipt.

**FirstTryTime:** The UTC time of the first try of this change order. MUST be in FILETIME format.

### 2.2.3.10 STAGE\_HEADER

To prepare a file for replication to another computer, the computer that sends the file will construct a staging file. A staging file is an archive consisting of the original file contents (that is, its data) as well as metadata related to it such as security information.

The STAGE\_HEADER MUST appear at the beginning of each staging file. Reparse data for the file MUST appear immediately following the STAGE\_HEADER if reparse data is associated with the file. After the reparse data (if present) MUST come the file data. The file data MUST be in the Microsoft backup format, as specified in [MS-BKUP]. An MD5 hash MUST be calculated over all stream data, so the receiving system can verify the integrity of the transmitted file. If compression is enabled, the stream data MUST be compressed as 64-kilobyte blocks (the file data is read in 64-KB blocks to be compressed on each block).<13>

The overall format of a staging file is as follows.

|   |
|---|
| <b>STAGE_HEADER</b>                             |
| Optional Reparse Data (only for reparse points) |
|   |
| File Data (might be compressed)                 |
|   |

The format of the STAGE\_HEADER MUST be as follows.

```
typedef struct _STAGE_HEADER {
    ULONG Major;
    ULONG Minor;
    ULONG DataHigh;
    ULONG DataLow;
    USHORT Compression;
    UCHAR NotUsed[6];
    FILE_NETWORK_OPEN_INFORMATION Attributes;
    CHANGE_ORDER_COMMAND ChangeOrderCommand;
    FILE_OBJECTID_BUFFER FileObjId;
    CO_RECORD_EXTENSION_WIN2K CocExt;
    GUID CompressionGuid;
    ULONG EncryptedDataHigh;
    ULONG EncryptedDataLow;
    LARGE_INTEGER EncryptedDataSize;
    BOOL ReparseDataPresent;
    ULONG ReparsePointDataHigh;
    ULONG ReparsePointDataLow;
    ULONG Padding2;
} STAGE_HEADER,
*PSTAGE_HEADER;
```

**Major:** A 32-bit, unsigned integer indicating the major version of the structure. This field MUST be 0.

**Minor:** A 32-bit, unsigned integer indicating the minor version of the staging file. This field exists in all versions of the staging file.

This field MUST contain one of the following values.

| Value                            | Meaning   |
|----------------------------------|---|
| NFRS_STAGE_MINOR_0<br>0x00000000 | Minor version is 0. This value MUST NOT be used.                      |
| NFRS_STAGE_MINOR_1<br>0x00000001 | Minor version is 1. ChangeOrder Record extension added to stage file. |
| NFRS_STAGE_MINOR_2<br>0x00000002 | Minor version is 2. Compression GUID added to stage file.             |
| NFRS_STAGE_MINOR_3<br>0x00000003 | Minor version is 3. Reparse Point data added to stage file.           |

NFRS\_STAGE\_MINOR\_1 is the earliest version. NFRS\_STAGE\_MINOR\_0 was never used in any products, and the server MUST NOT consider it to be a valid value.

<14>

The following field specifications in this section define the usage of the Minor values in the context of particular fields.

- CocExt
- CompressionGuid
- EncryptedDataHigh
- EncryptedDataLow
- EncryptedDataSize
- ReparseDataPresent
- ReparsePointDataHigh
- ReparsePointDataLow

**DataHigh:** A 32-bit, unsigned integer that specifies the higher four bytes of the offset from the beginning of the staging file to the file data. This field exists in all versions of the staging file.

**DataLow:** A 32-bit, unsigned integer that specifies the lower four bytes of the offset from the beginning of the staging file to the file data. This field exists in all versions of the staging file.

**Compression:** A 16-bit, unsigned integer that MUST be set to 0 and MUST be ignored on receipt.

**NotUsed:** Not used. MUST be 0. MUST be ignored on receipt.

**Attributes:** A structure of type FILE\_NETWORK\_OPEN\_INFORMATION that describes the file, as specified in [MS-FSCC] section 2.4.27, for the definition of this structure. This field exists in all versions of the staging file.

**ChangeOrderCommand:** A structure of type CHANGE\_ORDER\_COMMAND that contains a copy of the change order that was used to generate this staging file. The **ChangeOrderCommand.Extension** member is always set to NULL. This field exists in all versions of the staging file.

**FileObjId:** A structure of type FILE\_OBJECTID\_BUFFER. For the definition of this structure, see [MS-FSCC] section 2.1.3. This field exists in all versions of the staging file. Only the **ObjectId** field of this structure is used; all other fields in this structure MUST be 0 and MUST be ignored.

**CocExt:** A CO\_RECORD\_EXTENSION\_WIN2K structure. This field exists in staging file minor versions greater than or equal to NTFRS\_STAGE\_MINOR\_1.

**CompressionGuid:** A GUID that is initialized to the GUID for the compression algorithm used. FRS defines these GUIDs (see section 2.2.3.6.12). If compression is not enabled, the GUID MUST be set to 0. This field exists in staging file minor versions greater than or equal to NTFRS\_STAGE\_MINOR\_2.

**EncryptedDataHigh:** A 32-bit, unsigned integer. Not used. MUST be 0. MUST be ignored on receipt. This field exists in staging file minor versions greater than or equal to NTFRS\_STAGE\_MINOR\_2.

**EncryptedDataLow:** A 32-bit, unsigned integer. Not used. MUST be 0. MUST be ignored on receipt. This field exists in staging file minor versions greater than or equal to NTFRS\_STAGE\_MINOR\_2.

**EncryptedDataSize:** A LARGE\_INTEGER. Not used. MUST be 0. MUST be ignored on receipt. This field exists in staging file minor versions greater than or equal to NTFRS\_STAGE\_MINOR\_2.

**ReparseDataPresent:** A 32-bit, unsigned integer that is set to a nonzero value if this replicated file is a reparse point. If the file is not a reparse point, ReparseDataPresent MUST be set to zero. If

the replicated file is a reparse point, ReparseDataPresent MUST be set to a nonzero value, and the staging file will contain reparse data in addition to the actual file data. This field exists in staging file minor versions equal to NTFRS\_STAGE\_MINOR\_3.

**ReparsePointDataHigh:** A 32-bit, unsigned integer that specifies the higher four bytes of the offset from the beginning of the staging file to the reparse data. This field exists in staging file minor versions equal to NTFRS\_STAGE\_MINOR\_3.

**ReparsePointDataLow:** A 32-bit, unsigned integer that specifies the lower four bytes of the offset from the beginning of the staging file to the reparse data. This field exists in staging file minor versions equal to NTFRS\_STAGE\_MINOR\_3.

**Padding2:** Four bytes. Not Used. MUST be 0. MUST be ignored on receipt.

The staging file structure MUST contain the STAGE\_HEADER initialized as specified above. If the file is a reparse point (ReparseDataPresent is set to nonzero in the STAGE\_HEADER), the reparse data MUST exist at the offset specified by ReparsePointDataLow and ReparsePointDataHigh. After the reparse data (if present) the file data itself MUST be present. The file data can be found by the offset specified by DataLow and DataHigh.

If the replicated file is not a reparse point, the data offset MUST be set to 0x0400. If the replicated file is a reparse point, the data offset MUST be set to 0x0400 + ReparseDataLength + 0x18.

The file data MUST be processed in chunks of 64KB blocks and, if file compression is enabled (that is, the **CompressionGuid** member is set to a nonzero value), each 64KB chunk of file data is compressed using the algorithm specified by the CompressionGuid of STAGE\_HEADER.

The MD5 digest is generated on all file data before compression. This includes the WIN32\_STREAM\_ID (as specified in [MS-BKUP]) structures. However, the first stream from the file contains a SECURITY\_DESCRIPTOR structure, as specified in [MS-DTYP] section 2.4.6. For the purpose of generating the MD5 digest, certain bits must be cleared. These bits are all found in the second 16-bit, unsigned integer in the SECURITY\_DESCRIPTOR. To clear the proper bits, set bitwise AND 0xF3D4 to the second 16-bit, unsigned integer in the SECURITY\_DESCRIPTOR.

Once the MD5 digest is calculated, the bits MUST be restored to their previous state before being compressed (if necessary). The bits MUST be written to the staging file.

## 2.3 Directory Service Schema Elements

The protocol accesses the following Directory Service schema classes and attributes listed in the following tables.

For the syntactic specifications of the following <Class> or <Class><Attribute> pairs, refer either to:

Active Directory Domain Services (AD DS) ([MS-ADA1], [MS-ADA2], [MS-ADA3], and [MS-ADSC]).

Or to:

Active Directory Lightweight Directory Services (AD LDS) ([MS-ADLS]).

| Class           | Attribute          | Specified in              |
|-----------------|--------------------|---------------------------|
| nTFRSSettings   | fRSExtensions      | [MS-ADA1] (section 2.245) |
| nTFRSReplicaSet | fRSDirectoryFilter | [MS-ADA1] (section 2.243) |
|                 | fRSDSPoll          | [MS-ADA1] (section 2.244) |
|                 | fRSExtensions      | [MS-ADA1] (section 2.245) |
|                 | fRSFileFilter      | [MS-ADA1] (section 2.247) |
|                 | fRSFlags           | [MS-ADA1] (section 2.248) |



| Class              | Attribute   | Specified in   |
|--------------------|---|--|
|                    | fRSLevelLimit<br>fRSPartnerAuthLevel<br>fRSPrimaryMember<br>fRSReplicaSetGUID<br>fRSReplicaSetType<br>fRSRootSecurity<br>fRSServiceCommand<br>fRSVersionGUID<br>msFRS-Hub-Member<br>msFRS-Topology-Pref<br>schedule<br>objectGUID<br>name | [MS-ADA1] (section 2.249)<br>[MS-ADA1] (section 2.252)<br>[MS-ADA1] (section 2.253)<br>[MS-ADA1] (section 2.254)<br>[MS-ADA1] (section 2.255)<br>[MS-ADA1] (section 2.257)<br>[MS-ADA1] (section 2.258)<br>[MS-ADA1] (section 2.265)<br>[MS-ADA2] (section 2.495)<br>[MS-ADA2] (section 2.496)<br>[MS-ADA3] (section 2.225)<br>[MS-ADA3] (section 2.44)<br>[MS-ADA3] (section 2.1) |
| nTFRSMember        | frsComputerReference<br>fRSControlDataCreation<br>fRSControlInboundBacklog<br>fRSControlOutboundBacklog<br>fRSExtensions<br>fRSFlags<br>fRSPartnerAuthLevel<br>fRSRootSecurity<br>fRSServiceCommand<br>fRSUpdateTimeout                   | [MS-ADA1] (section 2.238)<br>[MS-ADA1] (section 2.240)<br>[MS-ADA1] (section 2.241)<br>[MS-ADA1] (section 2.242)<br>[MS-ADA1] (section 2.245)<br>[MS-ADA1] (section 2.248)<br>[MS-ADA1] (section 2.252)<br>[MS-ADA1] (section 2.257)<br>[MS-ADA1] (section 2.258)<br>[MS-ADA1] (section 2.263)   |
| nTDSDSA            | fRSRootPath   | [MS-ADA1] (section 2.256)  |
| nTDSDSARO          | fRSRootPath   | [MS-ADA1] (section 2.256)  |
| nTDSConnection     | enabledConnection<br>fromServer<br>options<br>schedule<br>nTSecurityDescriptor<br>objectGUID  | [MS-ADA1] (section 2.221)<br>[MS-ADA1] (section 2.237)<br>[MS-ADA3] (section 2.59)<br>[MS-ADA3] (section 2.225)<br>[MS-ADA3] (section 2.37)<br>[MS-ADA3] (section 2.44)  |
| Computer           | dnsHostName<br>serverReferenceBL  | [MS-ADA1] (section 2.185)<br>[MS-ADA3] (section 2.243)   |
| nTFRSSubscriptions | fRSExtensions<br>fRSVersion<br>fRSWorkingPath   | [MS-ADA1] (section 2.245)<br>[MS-ADA1] (section 2.264)<br>[MS-ADA1] (section 2.266)  |
| nTFRSSubscriber    | fRSExtensions<br>fRSFaultCondition<br>fRSFlags<br>fRSMemberReference<br>fRSRootPath<br>fRSServiceCommand<br>fRSServiceCommandStatus<br>fRSStagingPath   | [MS-ADA1] (section 2.245)<br>[MS-ADA1] (section 2.246)<br>[MS-ADA1] (section 2.248)<br>[MS-ADA1] (section 2.250)<br>[MS-ADA1] (section 2.256)<br>[MS-ADA1] (section 2.258)<br>[MS-ADA1] (section 2.259)<br>[MS-ADA1] (section 2.260)   |

| Class | Attribute   | Specified in   |
|-------|---|--|
|       | frsTimeLastCommand<br>frsTimeLastConfigChange<br>frsUpdateTimeout<br>schedule | [MS-ADA1] (section 2.261)<br>[MS-ADA1] (section 2.262)<br>[MS-ADA1] (section 2.263)<br>[MS-ADA3] (section 2.225) |
| TOP   | frsComputerReferenceBL<br>frsMemberReferenceBL                                | [MS-ADA1] (section 2.239)<br>[MS-ADA1] (section 2.251)   |

For FRS to function properly, certain critical directory objects (as well as their attributes and parent containers) MUST exist in the directory. These objects, which define a replica set's topology, schedule, and filters, MUST exist in the directory prior to starting up the FRS protocol for the first time. An implementation can perform these tasks at any time prior to first use of the FRS protocol. Directory replication (as specified in [MS-DRSR]) MUST be used to replicate these objects to all domain controllers in a domain; missing or corrupted objects cause FRS replication to fail. <15>

The following terms that appear in the sections below are defined as specified in [MS-ADTS] section 1.1: distinguished name (DN), canonical name (CN), relative distinguished name (RDN), attribute, object, naming context.

FRS reads the following attribute for every object.

**distinguishedName:** The distinguished name for the object. This is used to identify an object. <16>

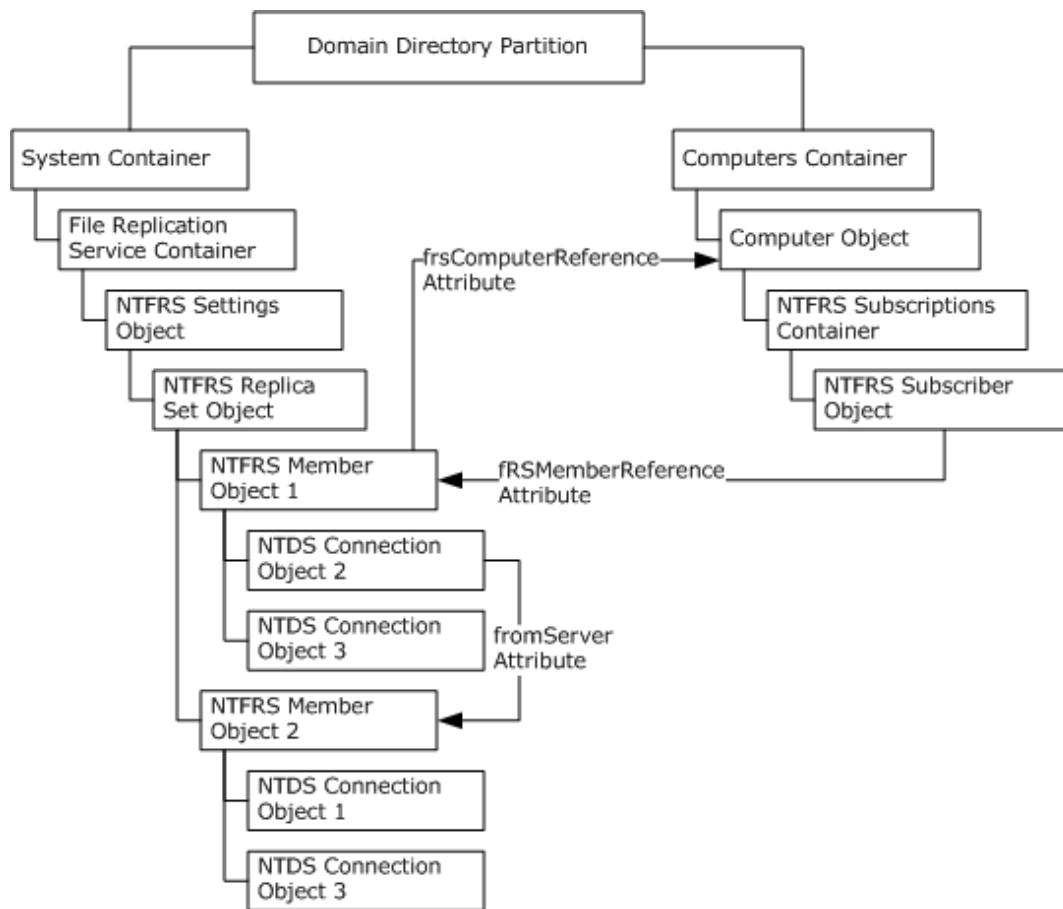
**objectGUID:** The unique identifier (UID) for the object. This value is a 16-byte GUID that is set when the object is created and cannot be changed. <17>

**uSNChanged:** USN (Update Sequence Number) value assigned by the local directory for the latest change, including creation. It is 8 bytes in size. <18>

Section 2.3.1 describes DFS Active Directory configuration. Section 2.3.2 describes SYSVOL Active Directory configuration.

### 2.3.1 DFS Active Directory Configuration

The following figure specifies the hierarchy of FRS-related Active Directory containers and objects for DFS replica sets. The figure also specifies how several of these objects are linked together by using reference attributes.



**Figure 1: Objects linked together by using reference attributes**

### 2.3.1.1 NTFRS Settings Object

An NTFRS settings object (class nTFRSSettings) is used to organize NTFRS replica set objects and other NTFRS settings objects into a hierarchy of FRS replica sets, each subtree of which is administered by a particular user or group. An implementation SHOULD allow permissions to be set on nTFRSSettings objects as a means of controlling access to the master list of replica sets.

The top-level NTFRS settings object MUST be stored in the domain-naming context at the following RDN, within each domain's Active Directory domain DNS object (as specified in [MS-ADSC] section 2.44).

"CN=File Replication Service,CN=System"

Exactly one top-level NTFRS settings object MUST exist for each domain in the forest. Each of these can contain additional NTFRS settings objects. Active Directory places no limit on the number of levels allowed. However, Active Directory can have limits on the string length of the fRSMemberReference attribute of the NTFRS Subscriber object (section 2.3.1.9), which indirectly determines the levels allowed.

The NTFRS settings object MUST use the schema definition provided by the nTFRSSettings class definition, as specified in [MS-ADSC] section 2.210.

The attribute of this object defined for FRS is:

- fRSExtensions: Attribute not used by FRS.

### 2.3.1.2 NTFRS Replica Set Object

Each NTFRS replica set object (class nTFRSReplicaSet) represents a replica set. A replica set consists of a number of replicas, each of which lives on a different computer.

Each NTFRS replica set object MUST be stored in the domain naming context at the following RDN, within the NTFRS settings object that is its parent:

"CN=ReplicaSetName"

Where ReplicaSetName is the name of the replica set.

The attributes of this object that FRS uses are:

**frsReplicaSetType:** MUST be set to 0x00000003 for DFS replica sets.

**frsFileFilter:** A Unicode string that can specify a comma-separated list of wildcard file name filters for the replica set (\* matches 0 or more characters; ? matches exactly 1 character). Any file whose name matches any of the filters SHOULD be excluded from replication. The value of this attribute SHOULD contain, at minimum: "\*.tmp,\*.bak,~\*". The frsFileFilter attribute is case-insensitive, and a one-space character (" ") can be present after each comma. FRS MUST NOT impose a limit on the maximum length of an extension or the list of extensions. However, the underlying file system or Active Directory can limit the lengths. The server MUST replicate all files if the frsFileFilter is set to the value ",", <19>

**frsDirectoryFilter:** A Unicode string that MAY specify the names of folders whose contents SHOULD be excluded from replication (\* matches 0 or more characters; ? matches exactly 1 character). The frsDirectoryFilter attribute is case-insensitive. FRS MUST NOT impose a limit on the maximum length of an extension or the list of extensions. However, the underlying file system or Active Directory MAY limit the lengths. <20>

**frsFlags:** Integer. When FRS has downloaded a file from a remote partner and is trying to use the new content to over-write an existing local file with the same name that has been opened by other processes, frsFlags specifies whether FRS MUST wait until it can open the file with write access, or FRS MUST rename the opened file out of the way in order to allow the installation of a new updated version of the file.

FRS MUST NOT rename the opened file if FRS does not have permission to delete that file. If the target file is currently open with a sharing mode that denies delete access to other opens, FRS MUST NOT install the updated version until the file is closed. FRS MUST apply this flag only when processing an incoming file, and MUST NOT apply this flag when processing an incoming directory.

| Value                          | Meaning   |
|--------------------------------|---|
| <Not Set><21>                  | Same as 0.  |
| The least significant bit is 0 | FRS MUST wait until it can open the file with write access before overwriting an opened local file. |
| The least significant bit is 1 | FRS MUST rename the opened file out of the way before overwriting an opened local file.             |

**frsPrimaryMember:** The distinguishedName of the NTFRS member object, as specified in 2.3.1.3, that acts as the primary member of this replica set. The primary member of a replica is usually the first member of the replica set, which acts as the seed to populate the whole replica set. A replica set MUST have exactly one primary member. (See Section 3.3.3).

**Schedule:** If set, MUST apply to all the NTDS connection objects in the replica set that do not have a schedule attribute.

The schedule attribute defines an 84-byte array that specifies the replication schedule for one week. Each bit in the byte array represents a 15-minute replication period. The possible values for each bit on the array MUST be as follows.

| Value | Meaning  |
|-------|--|
| 0     | Do not replicate during this 15-minute period. |
| 1     | Replicate during this 15-minute period.        |

The first bit in the array corresponds to the 15-minute interval between 0:00 A.M. and 0:14 A.M. on Sunday.

Exactly one NTFRS replica set object MUST exist for each replica set within a domain. A domain MUST define zero or more replica sets.

The NTFRS replica set object MUST use the schema definition provided by the nTFRSReplicaSet class definition, as specified in [MS-ADSC] section 2.209.

**objectGUID:** A unique identifier of the replica set.

**Name:** Uniquely identifies the replica set and specifies the replica set name.

- fRSDSPoll: Attribute not used by FRS.
- fRSExtensions: Attribute not used by FRS.
- fRSLevelLimit: Attribute not used by FRS.
- fRSPartnerAuthLevel: Attribute not used by FRS.
- fRSReplicaSetGUID: Attribute not used by FRS.
- fRSRootSecurity: Attribute not used by FRS.
- fRSServiceCommand: Attribute not used by FRS.
- fRSVersionGUIDmsFRS-Hub-Member: Attribute not used by FRS.
- msFRS-Topology-Pref: This attribute stores the preferred NTFRS topology setting. When an FRS member is added to or deleted from the replica set, adjustments are made to the connections between the FRS members in the replica set according to the value of this attribute. <22> This attribute MUST be set to one of the following values:

| Value | Topology Type               | Description  |
|-------|-----------------------------|--|
| 1     | FRS_RSTOPOLOGYPREF_RING     | FRS sorts members based on their site, such that members on the same site are neighbors. Then, for any two neighbors N1 and N2, it establishes a connection from N1 to N2, and from N2 to N1.                                    |
| 2     | FRS_RSTOPOLOGYPREF_HUBSPOKE | FRS establishes a connection between each member M in the replica set, such that M is not a hub node, and the hub node H that is identified by the msFRS-Hub-Member attribute. Connections are also established between H and M. |
| 3     | FRS_RSTOPOLOGYPREF_FULLMESH | Connections are established between all pairs of members in the replica set.   |
| 4     | FRS_RSTOPOLOGYPREF_CUSTOM   | Connections are established interactively by the user, through a user interface.   |

### 2.3.1.3 NTFRS Member Object

Each NTFRS member object (class nFRSMember) corresponds to a computer that is part of a replica set.

Each NTFRS member object MUST be stored in the domain naming context at the following RDN, within the NTFRS replica set object that is its parent:

"CN=memberName"

where memberName is a unique string to identify this member. It can be a GUID or host name or any strings that is unique.

The attributes of this object that FRS uses are:

**frsComputerReference:** MUST be the Unicode DN of the computer object to which this NTFRS member object applies. This attribute provides a link to the computer object from the NTFRS member object.

- fRSControlDataCreation: Attribute not used by FRS.
- fRSControlInboundBacklog: Attribute not used by FRS.
- fRSControlOutboundBacklog: Attribute not used by FRS.
- fRSExtensions: Attribute not used by FRS.
- fRSFlags: Attribute not used by FRS.
- fRSPartnerAuthLevel: Attribute not used by FRS.
- fRSRootSecurity: Attribute not used by FRS.
- fRSServiceCommand: Attribute not used by FRS.
- fRSUpdateTimeout: Attribute not used by FRS.

The NTFRS member object can contain one or more NTDS connection objects (section 2.3.1.6) that define the upstream partners that a member replicates from. NTDS connection objects refer to other NTFRS member objects in the same replica set object using the fromServer attribute.

Exactly one NTFRS member object MUST exist for each computer that participates in each replica set within a domain.

The schema definition for the NTFRS member object is provided by the nFRSMember class definition, as specified in [MS-ADSC] section 2.208.

### 2.3.1.4 NTDS Active Directory Service Agent (nTDSDSA) Object

This object represents the AD DS and Active Directory service agent (DSA) process on the server, as specified in [MS-ADSC] section 2.204. This class is not used by FRS.

The attributes of this object defined for FRS are:

fRSRootPath: Attribute not used by FRS.

### 2.3.1.5 NTDS Active Directory Service Agent Read Only (nTDSARSAR) Object

A subclass of the DSA, which is distinguished by its reduced privilege level, as specified in [MS-ADSC] section 2.205. This class is not used by FRS.

The attributes of this object defined for FRS are:

FRSRootPath: Attribute not used by FRS.

### 2.3.1.6 NTDS Connection Object

Each NTDS connection object specifies a one-way replication connection from an Upstream Partner to the member object (section 2.3.1.3) containing this NTDS Connection Object. Each NTDS connection object for a DFS replica set MUST be stored in the domain naming context at the following RDN, within the NTFRS member object that is its parent:

"CN=connectionName"

Where connectionName is a Unicode string containing a unique name for this connection object.<23>

The attributes of this object Defined for FRS are:

**enabledConnection:** A Boolean value that MUST specify the state of the connection. If the connection is available for use, the value of the attribute MUST be TRUE. If the connection is not available for use, the value of the attribute MUST be FALSE.

**fromServer:** MUST be a Unicode string containing the DN of the responding member's nTFRSMember object.<24>

**Schedule:** Can specify the replication times for this connection. The format of this attribute MUST be as specified in section 2.3.1.2.

**options:** Options defines the priority of the connection.

The following MUST be used to retrieve the value of options.

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
|   |   |   |   |   |   |   |   |   |   | 1 |   |   |   |   |   |   |   |   |   | 2 |   |   |   |   |   |   |   |   |   |   |   |   | 3 |   |   |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0 | 1 |   |   |   |   |
| X | T | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | P | P | P | S |

**X** = Do Not Care

**P** = Priority (3 bits for connection priority class)

**S** = FRS\_IGNORE\_SCHEDULE (1 – Ignore; 0 – Do Not Ignore)

**T** = NTDSCONN\_OPT\_TWOWAY\_SYNC (1 – Two way; 0 – One way)

FRS\_IGNORE\_SCHEDULE flag MUST be used to indicate if the schedule has to be ignored or followed for the initial VVJoin.

NTDSCONN\_OPT\_TWOWAY\_SYNC (force sync in opposite direction at end of sync) flag MUST be used in the following way: If the NTDSCONN\_OPT\_TWOWAY\_SYNC flag is set on the connection, FRS MUST merge (bitwise OR) the schedule on this connection with the schedule on the corresponding connection, between these two servers, that is in the opposite direction and use the resultant schedule on the connection in the opposite direction.

The use of Priority is as specified in section 3.1.5.1.<25>

Exactly one NTDS connection object MUST exist for each computer that replicates to each member of a replica set. An NTDS connection object MUST be stored in the directory under the NTFRS member object to which it is inbound. The fromServer attribute of that object MUST point to the NTFRS member object to which it is outbound.

The NTDS connection object MUST use the schema definition provided by the nTDSConnection class definition, as specified in [MS-ADSC] section 2.203.

### 2.3.1.7 Computer Object

Each computer object represents a computer in the domain. The relationship between the NTFRS member object and the computer object is indicated by the frsComputerReference attribute. This object is not specific to FRS and is not created by any FRS management tools.

The attributes of this object defined for FRS are:

**dnsHostName:** MUST be a Unicode string containing the computer's FQDN.

Exactly one computer object MUST exist for each computer within the domain.

The computer object MUST use the schema definition provided by the computer class definition, as specified in [MS-ADSC] section 2.21.

### 2.3.1.8 NTFRS Subscriptions Container

The NTFRS subscriptions object (class nTFRSSubscriptions) is primarily used as a container to group NTFRS subscriber objects that apply to a particular computer.

The NTFRS subscriptions object for a computer is stored under the Computer object (2.3.1.7).

The attributes of this object defined for FRS are:

- frSExtensions: Attribute not used by FRS.
- frSVersion: Attribute not used by FRS.
- frSWorkingPath: Attribute not required for protocol. The information stored is implementation-specific and has no bearing on the operations of the protocol.<26>

Exactly one NTFRS subscriptions container MUST exist for each computer within the domain that is a member of at least one replica set subscription.

The NTFRS subscriptions object MUST use the schema definition provided by the nTFRSSubscriptions class definition, as specified in [MS-ADSC] section 2.212.

### 2.3.1.9 NTFRS Subscriber Object

Every NTFRS subscriber object (class nTFRSSubscriber) under a computer's computer object corresponds to a replica set that the computer is a member of.

The NTFRS subscriber object for a DFS replica set MUST be stored within a computer NTFRS subscriptions object:

"CN=ReplicaSetName"

Where ReplicaSetName is the name of a replica set in which this computer participates.

The attributes of this object defined for FRS are:



**fRSMemberReference:** MUST be a Unicode string that specifies the DN of the nTFRSMember object of the replica set it corresponds to.

**fRSRootPath:** MUST be a Unicode string that specifies the path to the replica tree root to replicate.

**fRSStagingPath:** MUST be a Unicode string that specifies the folder under which the staging files are stored.

Exactly one NTFRS subscriber object MUST exist for each replica set to which a computer within the domain subscribes.

The NTFRS subscriber object MUST use the schema definition provided by the nTFRSSubscriber class definition, as specified in [MS-ADSC] section 2.211.

**fRSExtensions:** Attribute not used by FRS.

**fRSFaultCondition:** Attribute not used by FRS.

**fRSFlags:** Attribute not used by FRS.

**fRSServiceCommand:** Attribute not used by FRS.

**fRSServiceCommandStatus:** Attribute not used by FRS.

**fRSTimeLastCommand:** Attribute not used by FRS.

**fRSTimeLastConfigChange:** Attribute not used by FRS.

**fRSUpdateTimeout:** Attribute not used by FRS.

**schedule:** Attribute not used by FRS.

### 2.3.1.10 Object Types

| Attribute name            | Type   | Description   |
|---------------------------|--------|---|
| Frs-Computer-Reference    | DN     | DN of computer object.  |
| Frs-Computer-Reference-BL | DN     | DS created back link.   |
| FRS-Directory-Filter      | String | Comma-separated wildcard string for excluded dirs (* matches 0 or more characters; ? matches exactly 1 character).  |
| FRS-File-Filter           | String | Comma separated wildcard string for excluded files (* matches 0 or more characters; ? matches exactly 1 character). |
| FRS-Member-Reference      | DN     | DN of FRS member object.  |
| FRS-Member-Reference-BL   | DN     | DS created back link to subscriber object.  |
| FRS-Primary-Member        | DN     | DN to primary NTFRS-Member for initial Replica Tree loading.  |
| FRS-Replica-Set-Type      | DWORD  | Code for SYSVOL, DFS, other.  |
| FRS-Root-Path             | String | Path to the Replica Tree Root.  |
| FRS-Staging-Path          | String | File staging area.  |

| Attribute name   | Type   | Description                       |
|------------------|--------|-----------------------------------|
| FRS-Working-Path | String | Path to FRS database.             |
| Schedule         | Blob   | On-Off replication schedule.      |
| Server-Reference | DN     | DN of server object (for SYSVOL). |

### 2.3.1.11 Top class

The top level class from which all classes are derived, as specified in [MS-ADSC] section 2.265.

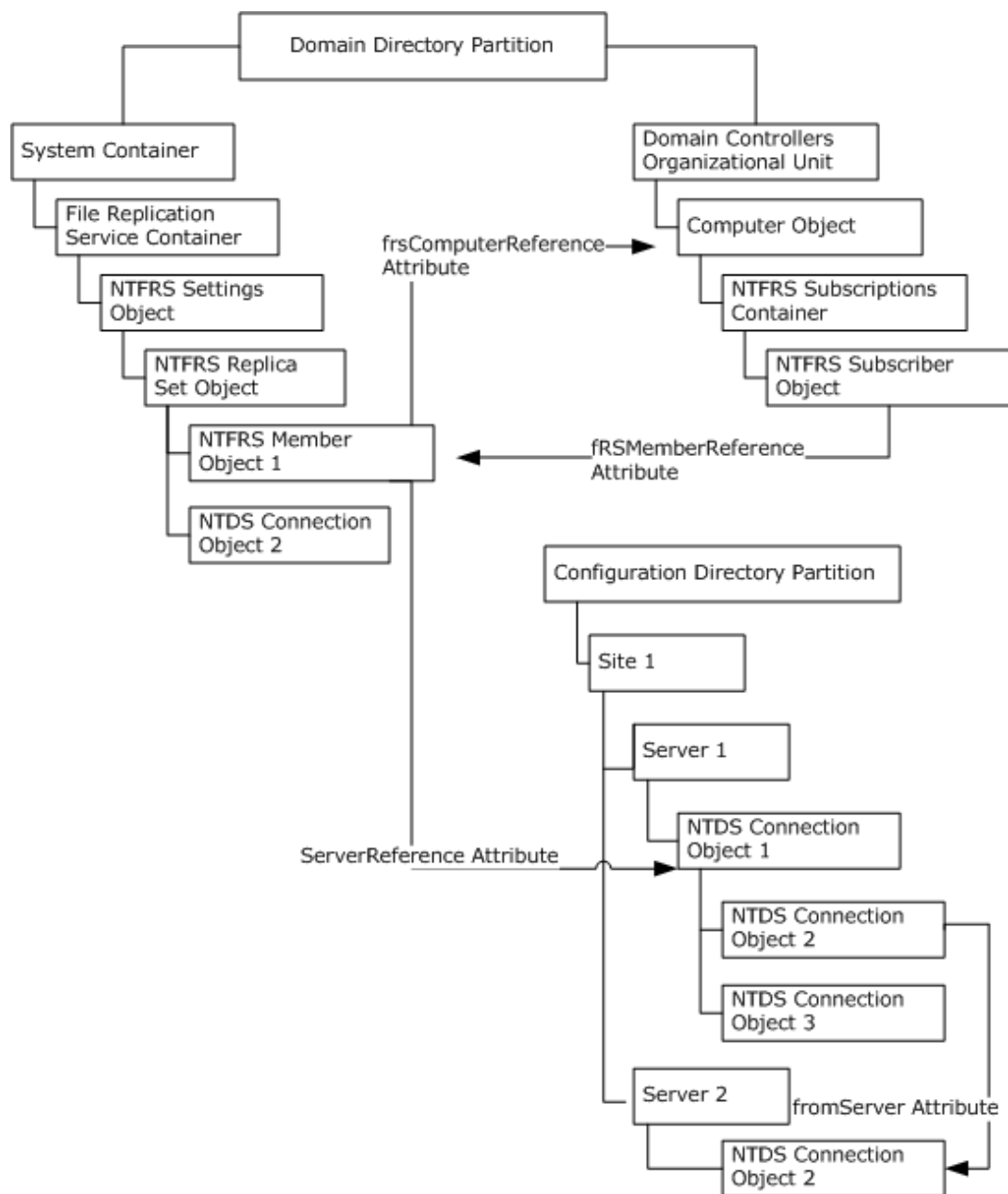
The attributes of this object defined for FRS are:

**frsComputerReferenceBL:** Attribute value is populated by DS and is a back link to the computer object.

**frsMemberReferenceBL:** Attribute value is populated by DS and is a back link to the member object.

### 2.3.2 SYSVOL Active Directory Configuration

SYSVOL replica sets have a different hierarchy of objects in Active Directory than DFS replica sets. See [MS-ADTS] for more information. This hierarchy is specified in the following figure.



**Figure 2: SYSVOL replica set hierarchy of objects**

### 2.3.2.1 NTFRS Settings Object

Same as DFS replica sets.

### 2.3.2.2 NTFRS Replica Set Object

Same as DFS replica sets with the following exception:

A domain **MUST** have only one NTFRS replica set object of type SYSVOL. The `frsReplicaSetType` attribute of a SYSVOL type object **MUST** be set to 2.<27>

### 2.3.2.3 NTFRS Member Object

In SYSVOL replica sets, the serverReference attribute of the NTFRS member object points to the NTDS Settings objects (class nTDSDSA) that contain the NTDS connection objects that this member replicates from.

### 2.3.2.4 NTDS Connection Object

For SYSVOL replica sets, FRS uses both Volatile Connection objects (section 3.3.4.2) and connection objects that are generated by the Knowledge Consistency Checker (KCC) that are located in the NTDS settings object in the configuration naming context. This is in contrast to DFS Share replication where NTFRS MUST use connection objects mentioned in section 2.3.1.6. These connection objects are also used during Active Directory replication.

For SYSVOL replica sets, the NTDS connection object is inbound to the NTFRS member object that corresponds to the NTDS settings object that the NTDS connection object is located under. It is outbound from the NTFRS member object that corresponds to the NTDS settings object that its fromServer attribute points to. <28>

### 2.3.2.5 Computer Object

Same as DFS replica sets, with the addition of the following attribute that FRS MUST use for SYSVOL replica sets:

**serverReferenceBL:** The distinguished name of the same computer under Configuration container's Sites object, in the form "CN=ServerName,CN=Servers,CN=SiteName,CN=Sites,CN=Configuration,DC=domainnamepart". In the preceding distinguished name relative to the domain's Configuration container, SiteName represents the site in which the computer is located, and ServerName represents the host part (the portion up to the first ".") of the computer's name. Active Directory MUST define this attribute as a multi-string value, however FRS MUST access only the first string stored in this attribute.

### 2.3.2.6 NTFRS Subscriptions Container

Same as DFS replica sets.

### 2.3.2.7 NTFRS Subscriber Object

Same as DFS replica sets, with the following exception.

The NTFRS subscriber object for a SYSVOL replica set is found in the domain naming context at the following RDN, within the computer's NTFRS subscriptions object:

"CN=Domain System Volume (SYSVOL share)"

Exactly one SYSVOL replica set subscriber object MUST be defined for each domain controller for every domain controller to have a copy of the domain SYSVOL.

## 2.4 FRS Performance Counters

FRS maintains a set of performance counters. A vendor SHOULD implement the capability to respond to requests for FRS performance counters through the Windows Remote Registry Protocol, as specified in [MS-RRP]. <29>

The way these statistics are collected from the FRS server process is implementation-specific. <30>

## 2.4.1 FileReplicaConn Object

The FileReplicaConn performance object displays performance statistics of the Replicaconn object that defines replica connections to Distributed File System (DFS) roots.

All counters are of type PERF\_COUNTER\_COUNTER (32-bit counter) unless otherwise noted in the counter's description. PERF\_COUNTER\_BULK\_COUNT is a 64-bit counter type. In the following table, the Counter Name field contains counter names that are exposed through the performance interface by OpenPerformanceData, as specified in [MS-RRP].

| Counter name                     | Description  |
|----------------------------------|--|
| Authentications                  | Number of authentications performed.   |
| Authentications in Error         | Number of authentications performed in error.  |
| Bindings                         | Number of bindings completed.  |
| Bindings in Error                | Number of bindings completed in error.   |
| Communication Timeouts           | Number of communications timed out.  |
| Fetch Blocks Received            | Number of fetch blocks received.   |
| Fetch Blocks Received in Bytes   | Number of fetch blocks (in bytes) received. This counter is of type PERF_COUNTER_BULK_COUNT. |
| Fetch Blocks Sent                | Number of fetch blocks sent.   |
| Fetch Blocks Sent in Bytes       | Number of fetch blocks (in bytes) sent. This counter is of type PERF_COUNTER_BULK_COUNT.     |
| Fetch Requests Received          | Number of fetch requests received.   |
| Fetch Requests Sent              | Number of fetch requests sent.   |
| Inbound Change Orders Dampened   | Number of inbound Change Orders dampened.  |
| Join Notifications Received      | Number of Join notifications received.   |
| Join Notifications Sent          | Number of Join notifications sent.   |
| Joins                            | Number of Joins.   |
| Local Change Orders Sent         | Number of local Change Orders sent.  |
| Local Change Orders Sent at Join | Number of local Change Orders sent at Join.  |
| Outbound Change Orders Dampened  | Number of outbound Change Orders dampened.   |
| Packets Sent                     | Number of packets sent.  |
| Packets Sent in Bytes            | Number of packets (in bytes) sent. This counter is of type PERF_COUNTER_BULK_COUNT.          |
| Packets Sent in Error            | Number of packets sent in error.   |
| Remote Change Orders Received    | Number of remote Change Orders received.   |
| Remote Change Orders Sent        | Number of remote Change Orders sent.   |

| Counter name | Description        |
|--------------|--------------------|
| Unjoins      | Number of unjoins. |

## 2.4.2 FileReplicaSet

The FileReplicaSet performance object displays performance statistics of the Replicaset object that defines a replica set (that is, one or more replicas, shared volumes, or folders that store duplicates of the contents of an original share). The object reports statistics for the computer that constitute the original share. For example, in a bidirectional ring topology of three computers A, B, and C, where computer A and computer C both replicate from computer B, the FileReplicaSet counters provide data for computer B.

All counters are of type PERF\_COUNTER\_COUNTER, unless otherwise noted in the counter description.

| Counter name                 | Description  |
|------------------------------|--|
| Authentications              | Number of authentications performed.   |
| Authentications in Error     | Number of authentications performed in error.  |
| Bindings                     | Number of bindings completed.  |
| Bindings in Error            | Number of bindings completed in error.   |
| Bytes of Files Installed     | Number of bytes of files installed. This counter is of type PERF_COUNTER_BULK_COUNT.   |
| Bytes of Staging Fetched     | Number of bytes of staging fetched. Staging is the temporary storage of files prior to replication. This counter is of type PERF_COUNTER_BULK_COUNT. |
| Bytes of Staging Generated   | Number of bytes of staging generated. This counter is of type PERF_COUNTER_BULK_COUNT.   |
| Bytes of Staging Regenerated | Number of bytes of staging regenerated. This counter is of type PERF_COUNTER_BULK_COUNT.   |
| Change Orders Aborted        | Number of Change Orders aborted.   |
| Change Orders Evaporated     | Number of Change Orders evaporated.  |
| Change Orders Issued         | Number of Change Orders issued.  |
| Change Orders Morphed        | Number of Change Orders morphed.   |
| Change Orders Propagated     | Number of Change Orders propagated.  |
| Change Orders Received       | Number of Change Orders received. This is an important counter to monitor. In an idle state, this counter has to be zero.                            |
| Change Orders Retired        | Number of Change Orders retired.   |
| Change Orders                | Number of Change Orders retried.   |

| <b>Counter name</b>               | <b>Description</b>  |
|-----------------------------------|---|
| Retried                           |   |
| Change Orders Retried at Fetch    | Number of Change Orders retried at fetch.   |
| Change Orders Retried at Generate | Number of Change Orders retried at generate.  |
| Change Orders Retried at Install  | Number of Change Orders retried at install.   |
| Change Orders Retried at Rename   | Number of Change Orders retried at rename.  |
| Change Orders Sent                | Number of Change Orders sent. This is an important counter to monitor. In an idle state, this counter has to be zero. |
| Communication Time Outs           | Number of communications timed out.   |
| DS Bindings                       | Number of directory service bindings.   |
| DS Bindings in Error              | Number of directory service bindings in error.  |
| DS Objects                        | Number of directory service objects.  |
| DS Objects in Error               | Number of directory service objects in error.   |
| DS Polls                          | Number of directory service polls.  |
| DS Polls with Changes             | Number of directory service polls with changes.   |
| DS Polls Without Changes          | Number of directory service polls without changes.  |
| DS Searches                       | Number of directory service searches.   |
| DS Searches in Error              | Number of directory service searches in error.  |
| Fetch Blocks Received             | Number of fetch blocks received.  |
| Fetch Blocks Received in Bytes    | Number of fetch blocks (in bytes) received. This counter is of type PERF_COUNTER_BULK_COUNT.                          |
| Fetch Blocks Sent                 | Number of fetch blocks sent.  |
| Fetch Blocks Sent in Bytes        | Number of fetch blocks (in bytes) sent. This counter is of type PERF_COUNTER_BULK_COUNT.                              |
| Fetch Requests Received           | Number of fetch requests received.  |
| Fetch Requests Sent               | Number of fetch requests sent.  |
| Files Installed                   | Number of files installed. This is an important counter to monitor. In an idle state, this counter has to be zero.    |
| Files Installed with Error        | Number of files installed with error.   |
| Inbound Change                    | Number of inbound Change Orders dampened.   |

| <b>Counter name</b>                     | <b>Description</b>   |
|---|--|
| Orders Dampened                         |  |
| Join Notifications Received             | Number of Join notifications received.   |
| Join Notifications Sent                 | Number of Join notifications sent.   |
| Joins                                   | Number of Joins.   |
| KB of Staging Space Free                | Number of kilobytes (KB) of staging space that were free. This is an important counter to monitor. In an idle state, this counter has to be 660KB. This counter is of type PERF_COUNTER_LARGE_RAWCOUNT.  |
| KB of Staging Space In Use              | Number of kilobytes (KB) of staging space that were in use. This is an important counter to monitor. If all available staging space is in use, replication stops. This counter is of type PERF_COUNTER_LARGE_RAWCOUNT.   |
| Local Change Orders Aborted             | Number of local Change Orders aborted.   |
| Local Change Orders Issued              | Number of local Change Orders issued.  |
| Local Change Orders Morphed             | Number of local Change Orders morphed.   |
| Local Change Orders Propagated          | Number of local Change Orders propagated.  |
| Local Change Orders Retired             | Number of local Change Orders retired.   |
| Local Change Orders Retried             | Number of local Change Orders retried.   |
| Local Change Orders Retried at Fetch    | Number of local Change Orders retried at fetch.  |
| Local Change Orders Retried at Generate | Number of local Change Orders retried at generate.   |
| Local Change Orders Retried at Install  | Number of local Change Orders retried at install.  |
| Local Change Orders Retried at Rename   | Number of local Change Orders retried at rename.   |
| Local Change Orders Sent                | Number of local Change Orders sent.  |
| Local Change Orders Sent At Join        | Number of local Change Orders sent at Join.  |
| Outbound Change Orders Dampened         | Number of outbound Change Orders dampened.   |
| Packets Received                        | Number of packets (that is, change notifications, file data, or other command packets) received. This is an important counter to monitor. In an idle state, there has to be no packets received unless a computer is having trouble Joining with other computers in the Replica Set. |
| Packets Received in                     | Number of packets (in bytes) received. This counter is of type   |



| <b>Counter name</b>                      | <b>Description</b>  |
|--|---|
| Bytes                                    | PERF_COUNTER_BULK_COUNT.  |
| Packets Received in Error                | Number of packets received in error.  |
| Packets Sent                             | Number of packets sent. This is an important counter to monitor.                    |
| Packets Sent in Bytes                    | Number of packets (in bytes) sent. This counter is of type PERF_COUNTER_BULK_COUNT. |
| Packets Sent in Error                    | Number of packets sent in error.  |
| Remote Change Orders Aborted             | Number of remote Change Orders aborted.   |
| Remote Change Orders Issued              | Number of remote Change Orders issued.  |
| Remote Change Orders Morphed             | Number of remote Change Orders morphed.   |
| Remote Change Orders Propagated          | Number of remote Change Orders propagated.  |
| Remote Change Orders Received            | Number of remote Change Orders received.  |
| Remote Change Orders Retired             | Number of remote Change Orders retired.   |
| Remote Change Orders Retried             | Number of remote Change Orders retried.   |
| Remote Change Orders Retried at Fetch    | Number of remote Change Orders retried at fetch.                                    |
| Remote Change Orders Retried at Generate | Number of remote Change Orders retried at generate.                                 |
| Remote Change Orders Retried at Install  | Number of remote Change Orders retried at install.                                  |
| Remote Change Orders Retried at Rename   | Number of remote Change Orders retried at rename.                                   |
| Remote Change Orders Sent                | Number of remote Change Orders sent.  |
| Replica Sets Created                     | Number of Replica Sets created.   |
| Replica Sets Deleted                     | Number of Replica Sets deleted.   |
| Replica Sets Removed                     | Number of Replica Sets removed.   |
| Replica Sets Started                     | Number of Replica Sets started.   |
| Staging Files Fetched                    | Number of staging files fetched.  |

| <b>Counter name</b>                | <b>Description</b>  |
|------------------------------------|---|
| Staging Files Generated            | Number of staging files generated.  |
| Staging Files Generated with Error | Number of staging files generated with error.   |
| Staging Files Regenerated          | Number of staging files regenerated.  |
| Threads Exited                     | Number of threads exited.   |
| Threads Started                    | Number of threads started.  |
| Unjoins                            | Number of unjoins.  |
| USN Reads                          | Number of update sequence number (USN) reads.   |
| USN Records Accepted               | Number of USN records accepted. This is an important counter to monitor. Replication is triggered by entries to the USN journal. A high value on this counter, such as one every five seconds, indicates heavy replication traffic and can result in replication latency. |
| USN Records Examined               | Number of USN records examined.   |
| USN Records Rejected               | Number of USN records rejected.   |

## 3 Protocol Details

The following sections specify details of the File Replication Service (FRS) Protocol, including abstract data models, interface method syntax, and message processing rules.

### 3.1 Common Details

All constants in section 3 that begin with `ERROR_` have their values defined in [MS-ERREF] section 2.2.

#### 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 organization is provided to explain how the protocol behaves. This document does not mandate that implementations adhere to this model, as long as their external behavior is consistent with what is described in this document.

A replica set contains a set of replica members. On a replica member, FRS maintains the following objects:

- **Replica Tree**
- **Inbound Connection**
- **Outbound Connection**
  - **Partner Version Vector**
  - **Outbound Connection Outbound Log**
- **IDTable**
- **Member Version Vector**
- **Inbound Log**
- **Member Outbound Log**
- **OutlogChangeHistoryTime**: The maximum time to retain the Change orders in Outbound log
- **FRSStaticPort**: A port used by FRS for replication over remote procedure calls (RPCs)
- **Staging File Generator**
- **Staging File Fetcher**

The following sections describe the elements of the data model.

##### 3.1.1.1 File System

The file system is the store where replicated files reside and are changed. The store maintains file data and organizes the data in a way that is specific to the semantics of the store.

##### 3.1.1.2 Replica Set Object

The Replica Set object represents a set of computers that replicate a specified folder tree and a common set of data among them. There will be one Replica Set object for every replica set. FRS

maintains a list of Replica Set objects. FRS replicates files and folders inside each replica set independently.

The Replica Set object maintains the following attributes:

**frsFileFilter:** A comma-separated list of file names (can include wildcards) that do not get replicated.

**frsDirectoryFilter:** A comma-separated list of folder names (can include wildcards) that do not get replicated.

**objectGUID:** A unique identifier of the replica set.

**FRSReplicaSetType:** Either SYSVOL or DFS replicas.

**String:** Identifies the replica set. Sometimes cannot be unique.

### 3.1.1.3 Member Object (Replica Member Object)

The Member object corresponds to a computer that is a part of the replica set. The Member object contains one or more Connection objects that define the upstream partners that a member replicates from. FRS determines the set of replica sets of which the local machine is a member.

For each replica set that the local machine is a member of, the Member object of the local machine maintains the following set of attributes:

**Connection Objects:** Define the connections that this member receives change orders from or sends change orders to.

Member GUID: Uniquely identifies the member.

Replica Set GUID: Identifies the replica set to which this system belongs.

Replica Tree Root GUID: Object ID for the replica tree root in the IDTable.

Replica Tree Root Path: Path to the replica tree root on the local server.

Version Vector: Current version numbers for all the files in the replica set.

A single machine cannot be a member of a given replica set more than once. When a member object is deleted from the replica set, the following configuration information is deleted from Active Directory:

1. The nTFRSSubscriber object ([MS-ADSC] section 2.211)
2. The nTFRSMember object ([MS-ADSC] section 2.208)
3. All NTDS Connection objects which have one of their endpoints as the member.

### 3.1.1.4 Replica Tree

A replica tree is a local folder (including all the files and folders underneath it) that usually is saved as a tree structure in the file system. The content under a Replica Tree is kept in sync with other replica members.

A replica tree cannot be a subtree of another replica tree.

### 3.1.1.5 IDTable

The IDTable object stores persistent data associated with each file and folder in the replica set. There is one IDTable entry for each file and folder in the replica set. This data is used to uniquely identify

each file and folder, and is used to determine if the file or folder needs to be replicated. The information in the IDTable object is saved across restarts.

The IDTable maintains the following attributes:

Event Time: 64-bit integer containing the time of the last change to the file or folder.

File Name: Name of the file.

Object ID: GUID assigned to each file or folder. The Object ID uniquely identifies the file or folder within the replica set and across all members of the replica set.

Originator GUID: GUID assigned to the system on which the first change order originated. The originator GUID uniquely identifies the system on which the file or folder changed.

Originator Version Number: Version number of the last change for this file on the replica member identified by the originator GUID.

Parent Object ID: Object ID of the folder that contains this object.

Replica Set Name: String that identifies the replica set to which the file belongs.

Version Number: Identifies the version of the file or folder. The file version number is a monotonically increasing unsigned long that gets incremented each time the file or folder has changed and is replicated. The file version number is updated each time the file is closed after being modified. The server uniquely identifies the files on the local file system. If NTFS is being used, the server uses the reference number as the unique key to identify the files. If this protocol is being implemented on a different file system, the vendor is free to choose a way to uniquely identify a file in a file system.<31>

### **3.1.1.6 Inbound Log Object (InLog)**

The Inbound Log object stores pending change orders to be processed. The Inbound Log contains change orders arriving from all upstream partners. The change orders are logged in the order they arrive.

As entries are processed, acknowledgments are sent to the upstream partners. The Inbound Log tracks the progress of a change order and determines where to continue after a system crash or a retry of a failed operation. The information in the Inbound Log object is saved across restarts.

### **3.1.1.7 Outbound Log Object (OutLog)**

The outbound log object stores pending change orders to be sent to downstream partners. (The outbound log is different from the outbound connection outbound log, which stores pending change orders to be sent only to the specific downstream partner in the outbound connection.) By default, change orders remain in the outbound log for the amount of time specified by the **OutlogChangeHistoryTime** ADM element, even if all replication Partners have received the change. The minimum limit is 1 minute. The maximum limit is 0x7FFFFFFF minutes. The outbound log is saved across restarts.<32>

### **3.1.1.8 Connection Object**

The Connection object represents a directional connection between two members. It tracks the state of each inbound and outbound connection for the replica set and tracks the delivery state of each outbound change order. The local member object replicates with other member objects in the direction specified in the connection objects based on the connection object's schedule. Connection objects are populated when FRS service polls directory service (DS) for FRS configuration changes.

Each Member object maintains a list of Connection objects, each of which contains the following data about a connection.

**Direction:** Inbound or outbound.

**Session GUID:** Each Join on a connection is uniquely identified by a session GUID. The session GUID is used to retry change orders that were accepted by a connection that has since unjoined from its partner. The change orders for a previous session are retried because they are out-of-order with respect to the change orders for the current session. In FRS, the order of change orders is maintained per session by coordinating the partners at connection establishment time.

**Join Status:** Joined or not joined. Partner version vectors if joined.

**Schedule:** See section 2.3.1.2.

**Partner GUID:** GUID that identifies the member of the replica set that this connection connects to.

**LastJoinTime:** MUST be the Join time of the current successful connection session.

For outbound connections, the Member object also has the following objects:

**Outbound Connection Outbound Log:** Change orders to be sent to the downstream partner. The leading (or next change order to be sent out) and trailing (or last acknowledged change order) index for the downstream partner is in the outbound connection outbound log. The outbound connection object keeps track of its last Join time.

**Partner Version Vector:** Downstream partner's version vector. <33>

### 3.1.1.9 Staging File Object

FRS uses staging files to transfer files between Partners during replication. There is one staging file per file replicated. The staging file for a file to be replicated is created by first writing a STAGE\_HEADER to disk. If the replicated file is a reparse point, the STAGE\_HEADER is followed by the reparse data. Finally, the file data is written to the staging file. The file data in the staging file can be compressed to save disk space and network bandwidth.

FRS MUST use the format specified in [MS-BKUP] section 2.1 to read and write the replicated file. When the replicated file is installed into the new file system, backup semantics are again used to reconstitute the replicated file from the contents of the staging file. The staging file is sent to downstream partners on request (CMD\_SEND\_STAGE). See section 3.3.4.4.7.

The staging file object contains the following parts:

- Staging Header.
- Reparse Data: Reparse point data. This data only exists if the file is a reparse point. <34>
- File Data: Contents of the file. This data can be compressed.

For the staging header, reparse data, and file data, see section 2.2.3.10.

### 3.1.1.10 Change Order Object

A change order is a message that contains information about a file or folder that has changed on a replica member. Change orders are stored in the inbound and outbound queues on each replica member. For Change Order objects, see section 2.2.3.2.

There are five types of change orders. A change order can fit into more than one category type.

#### 3.1.1.10.1 Local Change Order

A local change order is a change order that is created because of a change to a file or folder on the local server. The local server becomes the originator of the change order and constructs a staging file.

Local change orders are identified by the CO\_FLAG\_LOCALCO flag inside the CHANGE\_ORDER\_COMMAND structure Flags field.

A change order is referred to as a Remote Change Order if it is not a Local Change Order.

When the downstream partner receives a change order with the CO\_FLAG\_LOCALCO flag, the change that this change order represents originated on the upstream partner.

#### **3.1.1.10.2 Retry Change Order**

A Retry Change Order is a change order that is a retry of a previous change order that was not replicated successfully due to a temporary condition. Both local and remote change orders become Retry Change Orders when temporary failure is hit.

Retry Change Orders are identified by the CO\_FLAG\_RETRY flag in the change order Flags field. A Retry Change Order is also an Out-of-Order change order.

If the downstream partner cannot apply the change order because the change order corresponding to its parent folder has not arrived or been installed, then the Change Order becomes a Retry Change Order. In that case, the server SHOULD process the order further only if the following conditions are met:

- DATA\_EXTENSION\_RETRY\_TIMEOUT.Count in CMD\_REMOTE\_CO is less than or equal to an implementation-specific value.<35>
- (CurrentTime - DATA\_EXTENSION\_RETRY\_TIMEOUT.FirstTryTime) in CMD\_REMOTE\_CO is less than or equal to an implementation specific value.<36>

#### **3.1.1.10.3 Directed Change Order**

A directed change order is a change order that is directed at a single downstream partner and produced when a Partner is doing a VVJoin, such as during initial synchronization (see section 3.1.1.11).

Directed change orders are used by the upstream partner to specify a specific downstream partner to which to send a change order.

Directed change orders are identified by the CO\_FLAG\_DIRECTED\_CO or the CO\_FLAG\_VVJOIN\_TO\_ORIG flag in the change order Flags field.

#### **3.1.1.10.4 Out-of-Order Change Order**

A change order is marked out-of-order to indicate that there was at least one more recent change order that was propagated before this change order. Every member that receives a change order marked out-of-order accepts it even though the member's version vector indicates that a more recent change has already been received. When a change order is retried by a member after a later change order for a different file is received, the retried change order is marked out-of-order and continues to be propagated to downstream partners. If a newer change order is received for the same file as the retried change order, the retried change order is out of date and is NOT propagated.

Out-of-order change orders are identified by the CO\_FLAG\_OUT\_OF\_ORDER flag inside the CHANGE\_ORDER\_COMMAND structure Flags field.

#### **3.1.1.10.5 Skip-VV-Update Change Order**

A change order is marked Skip-VV-Update when an earlier change order is pending to send to a downstream partner. This indicates to the downstream partner that this change does not guarantee that it has received all the changes up to this point. It is also NOT used to update the version vector. This flag is written to the IDTable so that any future VVJoins pick it up.

Skip-VV-Update change orders are identified by the CO\_FLAG\_SKIP\_VV\_UPDATE flag inside the CHANGE\_ORDER\_COMMAND structure Flags field.

### **3.1.1.11 Version Vector Object**

The version vector is a vector of tuples. Each tuple consists of an originator GUID and a Volume Sequence Number (VSN). There is one tuple for each member of the replica set. Each replica tree has its own version vector.

The originator GUID is a unique identifier for each member of the Replica Set. The VSN is the VSN of the last change order received from the member identified by the originator GUID except when the change orders are received out of order. If change orders are received out of order, the VSN in the version vector is the VSN of the most recent change order received that was not out of order.

Each member has its own originator GUID in its version vector. The VSN associated with a member's own originator GUID MUST be the VSN of the last change order that the member generated.

When a member initializes itself, it sets its own VSN to the current time (FILETIME). Using the current time ensures that all VSNs generated from the time of initialization forward will be greater than any VSN previously generated on the system. This is critical to ensure that replication continues to work properly across restarts of FRS on a system. Each time a change order is generated, the VSN is incremented, and the new VSN is assigned to the new change order.

The version vector is persistent. If FRS stops running, the stored version vector can be used to determine what files need to be replicated onto the Partner to bring it back up to date. This is accomplished by a VVJoin process. In a VVJoin, the downstream partner establishes a connection with an upstream partner. Once the connection is established, the downstream partner's version vector is sent to the Upstream Partner. The Upstream Partner compares the downstream partner's version vector to its own version vector. Any entries in the downstream partner's version vector that are older than the Upstream Partner's version vector entries cause the Upstream Partner to scan its IDTable to determine what files and folders are replicated to bring the downstream partner up to date.

Once the upstream partner determines what files and folders are replicated to the downstream partner, change orders are generated and sent to the downstream partner. Once the downstream partner receives a change order from the Upstream Partner, it determines if a staging file is needed for this change order. If a staging file is needed, the downstream partner requests the staging file from the Upstream Partner. Once the downstream partner receives all parts of the staging file, or if a staging file is not needed, the downstream partner applies the change order on the file system and updates IDTable and version vector objects based on the change order. The downstream partner then forwards the same change order to its downstream partners.

A VVJoin takes place when a schedule initially comes online.

On receipt of a change order, the originator GUID and the VSN of the change order is checked against the version vector. This check (called version dampening) allows a member to recognize change orders for files or folders that have already been applied in the local replica tree.

### **3.1.1.12 Communication Packet Object**

FRS uses communication packet objects to send data between replica members using FrsSendCommPkt during initial sync and normal sync.

**Major Version:** Major version number of the COMM\_PACKET structure.



**Minor Version:** Minor version number of the COMM\_PACKET structure.

**Data Packet Length:** Size in bytes of the data packet.

**Data Packet Buffer:** Actual data being transmitted is packaged into elements and placed in the data packet buffer.

### 3.1.2 Timers

FRS periodically polls the directory service (DS) to obtain new or updated information about the replica set topology, schedules, and file and folder filters based on the following two timers:

- Short DS Polling Interval Timer
- Long DS Polling Interval Timer

Each time the FRS service is started, FRS polls the computer object in Active Directory at short intervals for a period of time. If no changes in Active Directory are detected during that time period, FRS polls in long intervals until it detects configuration or subscriber list changes in Active Directory. It then goes back to polling at short intervals, again, for a period of time.<37>

The configuration change can be any of the following:

- New member joining the replica set.
- Member leaving the replica set.
- Change in the list of upstream partners.
- Change in the list of downstream partners.
- Change in the physical locations of the replica tree or staging folder.
- Change in the replication schedule on individual connections.
- Change in the replica set wide replication schedule.<38>

### 3.1.3 Initialization

When FRS is started on a new machine for the first time, the IDTable MUST be empty. If IDTable is not empty, FRS MUST assume the records in the IDTable are valid and this is not the first start of FRS on a new machine for the first time. FRS MUST poll Active Directory periodically to discover new Replica Sets that the local machine is a member of. When a new replica set is discovered, FRS MUST scan the replica tree root to fill the IDTable.

FRS must initialize the **OutlogChangeHistoryTime** ADM element to an implementation specific value.<39>

The **FRSStaticPort** ADM element MUST be initialized to 0. If the server uses static port for the replication, the server MUST update this value to an implementation-specific port number.<40>

### 3.1.4 Message Processing Events and Sequencing Rules

No message processing events and sequencing rules.

## 3.1.5 Timer Events

### 3.1.5.1 Both Short and Long DS Polling Interval Timers

On expiration, FRS MUST do the following:

1. Locate the computer object for the server it is running on and enumerate the FRS Subscriber objects.
2. For a given subscriber, follow the `frsMemberReference` attribute to the FRS Member object.
3. Read the FRS replica set object (always directly above the FRS Member object) to determine if it is a SYSVOL or a DFS replica set.
4. Enumerate the connection objects. This process is different for SYSVOL and DFS.

For DFS replica sets, FRS MUST:

- Find all the NTDS connection objects that represent an inbound connection to this member. (All NTDS connection objects that represent an inbound connection are directly under the NTFRS member object.)
- Find all the NTDS connection objects that represent an outbound connection to this member. (All NTDS connection objects that represent an outbound connection have this member name in the `fromServer` attribute.) See section 2.3.1.6.
- Form a list of FRS member objects that have an inbound or outbound connection with this member. This list MUST be a combined list for all the replica sets that this server is a member of.
- Form a list of the computers corresponding to the members in the member list. All members have an `frsComputerReference` attribute pointing to the computer object.
- Find the FQDN of each of the computers in the computer list. This is the name that FRS uses to bind to its upstream and downstream partners.

For SYSVOL replica sets, FRS MUST:

- Find all the NTDS Connection objects that represent an inbound connection to this member. The `serverReference` attribute on the FRS member object points to the NTDS settings object. (All the NTDS connection objects that represent an inbound connection are directly under the NTDS settings object.)
- Find all NTDS connection objects that represent an outbound connection to this member. (All NTDS connection objects that represent an outbound connection have this member name in the `fromServer` attribute.)
- Form a list of member objects that have an inbound connection or outbound connection with this member. Forming the list of members is a two-step process for SYSVOL replica sets. FRS MUST:
  1. Form the list of NTDS settings objects.
  2. Find the NTFRS member objects that point to each of these NTDS settings objects. This list MUST be a combined list for all the replica sets that this server is a member of.
- Find the FQDN of each of the computers in the computer list. This is the name that FRS uses to bind to its upstream and downstream partners.

Once FRS detects a configuration change and realizes that it became a new member in a given replica set, FRS MUST:

1. Add the replica tree root folder into the IDTable.
2. Generate a new GUID to uniquely identify the local replica set member. This GUID MUST be used in version vectors to identify the local replica set member.
3. Initialize the replica set object with the values discovered from Active Directory.
4. Initialize all the connections of the local replica set member.
5. Initialize inbound log and inbound log to be empty.
6. Put the new replica in Seeding state, which triggers the initial sync process. (If the member is configured as a primary member, this process does not apply.)
7. Start monitoring the replica tree root in the local file system to detect changes that need to be replicated out to downstream partners.

Once FRS detects a configuration change and determines that it is no longer a member in a given replica set, the following MUST be performed by FRS:

1. All the connections having one endpoint as the machine which is going to be removed are first set to Unjoined state, and then deleted.<41>
2. Outbound log processing is shut down.
3. Entries corresponding to this replica in the filter tables and parent file IDTable are deleted.
4. Journal processing for the replica is stopped.
5. Staging space is reclaimed.
6. All the pre-install files and pre-install directories for the replica are deleted.

Following are some of the types of changes that trigger replication:

- File or folder was created.
- File or folder was deleted.
- File or folder was renamed.
- Main file data stream was overwritten.
- Main file data stream was extended.
- Main file data stream was truncated.
- Basic information was changed. Changes related to the archive flag or the last access time do not cause files to be replicated.
- Object ID was changed.
- Alternate data stream name was changed.
- Alternate data stream was overwritten.
- Alternate data stream was extended.
- Alternate data stream was truncated.

- Extended file attribute was changed.
- File permission, auditing, or ownership was changed.
- File compression attribute changed.
- File encryption changed. This type of change triggers replication only when the file is changed from encrypted to not encrypted. <42>
- Reparse point deleted or reparse point changes related to SIS or HSM.

When a change is detected, FRS MUST replicate the entire file or folder in the staging file. FRS relies on the update sequence number (USN) journal to log records of files that have changed on a replica member. As a result, FRS does not lose track of a changed file even if a replica member shuts down abruptly. After the replica member comes back online, FRS replicates new or updated files that originated from other replica members, as well as replicating locally created or updated files that occurred before the shutdown. This replication takes place according to the replication schedule.

FRS uses the following process to determine if a file has changed:

1. FRS monitors the USN journal for changes. When FRS detects a closed record for a file (a handle being closed can trigger this), FRS gathers relevant information about the recently closed file, including the file's attributes and MD5 checksum, from the file IDTable.
2. FRS computes an MD5 checksum for the recently closed file. The MD5 checksum is calculated based on the file's data, including its security descriptors. File attributes are not included in this calculation. If the MD5 checksum and attributes of the recently closed file are identical to the information about the file stored in the file IDTable, the file is suppressed and not replicated. If either the MD5 checksum or the attributes are different, FRS begins the change order process.

If a file is changing frequently, the changes are suppressed using the aging cache. Changes to a file are held in the aging cache for a period of time (three seconds by default) before FRS examines the file to determine if replication is necessary. This allows multiple changes to be collapsed into one change order and prevents excessive updates from getting sent to downstream partners.

When doing initial sync for the replica set member, FRS performs the following steps in order to complete a VVJoin with each of its upstream partners:

1. FRS forms a list of connections to upstream partners in order of the connection priority stored in the AD connection object (see section 2.3.1.6).
2. For every connection in a given priority class (see the following table), FRS sends a NEED\_JOIN request to the upstream partner. For more details on connection joining, see section 3.3.4.6.
3. A downstream partner can only perform a VVJoin with a single upstream partner at a time. Once an upstream partner responds to an FRS request, the connections to all other upstream partners are paused to prevent VVJoins from proceeding on those connections. On successful completion of the VVJoin on a connection, the connection state never goes back to initial sync. If the VVJoin failed (that is, if the partner goes offline), the connection is returned back to the list and the logic goes back to connection selection, and then performs a VVJoin again on the newly selected connection. For more details on how files and folders are replicated out to partners during initial sync, see section 3.3.4.4.1.
4. The replica set leaves the Seeding state and becomes Online only after it has successfully completed a VVJoin with each of its upstream partners.

FRS performs initial VVJoin with one partner at a time. The sequence depends on the priority class of the upstream partner. *Next class* in the table below means the class with a larger priority value (plus 1). The priority classes and their meanings follow.

| Priority class | Behavior  |
|----------------|---|
| 0              | FRS MUST go to partners in the next class even if it has not successfully completed VVJoin with partners in the current class. The class has the lowest priority.                             |
| 1-2            | FRS MUST NOT go to partners in the next class until it has successfully completed VVJoin with every partner in the current class. 1 > 2 in priority. These classes have the highest priority. |
| 3-4            | FRS MUST NOT go to partners in the next class until it has successfully completed VVJoin with at least one partner in the current class. 3 > 4 in priority.                                   |
| 5-7            | FRS MUST go to partners in the next class even if it has not successfully completed VVJoin with partners in the current class. 5 > 6 > 7 in priority.   |

Using the priority class assigned to a connection, a sorted list of upstream partners is formed using the following priority order (highest to lowest): 1, 2, 3, 4, 5, 6, 7, 0.

FRS walks through this list and performs a VVJoin with each partner. The behavior differs based on the priority class.

Along with the priority, the FRS\_IGNORE\_SCHEDULE value SHOULD be used to indicate if the schedule is to be ignored or followed for the initial VVJoin.

The Options attribute on the NTDS Connection object, used to set the VVJoin priority, indicate that FRS MUST force a sync in the opposite direction at the end of a sync, and indicate whether the schedule is to be ignored. The format of the Options attribute is as specified in section 2.3.1.6.

For the SYSVOL replica set, the FRS service MUST detect the domain functional level (as specified in [MS-ADTS] section 6.1.4.3) during Active Directory Polling. FRS might not be used as the SYSVOL replication engine for some domain functional levels. <43><44>

### 3.1.6 Other Local Events

No other local events are processed.

## 3.2 FRSAPI Interface

The NtFrsApi interface provides administrative and monitoring methods. These methods are useful for tools that can observe and control FRS replication. The version of this interface is 1.1.

To receive incoming remote calls for this interface, the server MUST use the port specified in the **FRSStaticPort** ADM element, if it is not set to 0. Otherwise, the server MUST implement the service at a dynamically assigned endpoint. The server must use the UUID D049B186-814F-11D1-9A3C-00C04FC9B232.

The FRSAPI interface provides three types of methods:

1. Methods used for monitoring only:
  - NtFrsApi\_Rpc\_Get\_DsPollingIntervalW
  - NtFrsApi\_Rpc\_InfoW
  - NtFrsApi\_Rpc\_IsPathReplicated
  - NtFrsApi\_Rpc\_WriterCommand
2. Methods used for configuring only:

- NtFrsApi\_Rpc\_Set\_DsPollingIntervalW

3. Methods used for ForceReplication to overrule the connection schedule:

- NtFrsApi\_Rpc\_ForceReplication

Opnum values 0 through 3 and 6 MUST NOT be used over the wire by this protocol.

### 3.2.1 Abstract Data Model

This section describes a sample model of possible data organization that can be implemented to support this protocol. The purpose of this description is simply to help explain how this aspect of the protocol works. This specification does not prescribe that implementations adhere to this model as long as their external behavior is consistent with what is described throughout this document.

**IsSetDsPollingIntervalAccessCheckEnabled:** Specifies whether a user or process can change the Active Directory polling interval.

**SetDsPollingIntervalAccessRequired:** Specifies the permissions with which a user or process can change the Active Directory polling interval.

**IsGetDsPollingIntervalAccessCheckEnabled:** Specifies whether a user or process can get information about the Active Directory polling interval.

**GetDsPollingIntervalAccessRequired:** Specifies the permissions with which a user or process can get information about the Active Directory polling interval.

**IsGetInfoWAccessCheckEnabled:** Specifies whether a user or process can get internal information about FRS.

**GetInfoWAccessRequired:** Specifies the permissions with which a user or process can get internal information about FRS.

**IsWriterCommandCheckEnabled:** Specifies whether the FRS writer can issue commands.

**WriterCommandAccessRequired:** Specifies the permissions with which FRS writer can issue commands.

**IsPathReplicatedCheckEnabled:** Specifies whether a user or process can check the Path.

**ReplicatedPathAccessRequired:** Specifies the permissions with which a user or process can check the Path.

**IsForceReplicationCheckEnabled:** Specifies whether user or process can force replication.

**ForceReplicationAccessRequired:** Specifies the permissions with which a user or process can force replication.

### 3.2.2 Timers

No other timers required.

### 3.2.3 Initialization

**IsSetDsPollingIntervalAccessCheckEnabled** SHOULD <45> be initialized to "None" or to an implementation-specific value

**SetDsPollingIntervalAccessRequired** SHOULD <46> be initialized to "None" or to an implementation-specific value.

**IsGetDsPollingIntervalAccessCheckEnabled** SHOULD <47> be initialized to "None" or to an implementation-specific value.

**GetDsPollingIntervalAccessRequired** SHOULD <48> be initialized to "None" or to an implementation-specific value.

**IsGetInfoWAccessCheckEnabled** SHOULD <49> be initialized to "None" or to an implementation-specific value.

**GetInfoWAccessRequired** SHOULD <50> be initialized to "None" or to an implementation-specific value.

**IsWriterCommandCheckEnabled** SHOULD <51> be initialized to "None" or to an implementation-specific value.

**WriterCommandAccessRequired** SHOULD <52> be initialized to "None" or to an implementation-specific value.

**IsPathReplicatedCheckEnabled** SHOULD <53> be initialized to "None" or to an implementation-specific value.

**ReplicatedPathAccessRequired** SHOULD <54> be initialized to "None" or to an implementation-specific value.

**IsForceReplicationCheckEnabled** SHOULD <55> be initialized to "None" or to an implementation-specific value.

**ForceReplicationAccessRequired** SHOULD <56> be initialized to "None" or to an implementation-specific value.

### 3.2.4 Message Processing Events and Sequencing Rules

On receiving each FRSAPI message, an FRS server MUST verify the access check flags (specified in section 3.2.1) for the corresponding request. If the flag is set to "None", the server MUST fail the call with an implementation-specific value. If the flag is set to "Disabled", the server MUST process the request without performing an access check. If the flag is set to neither "None" nor "Disabled", the server MUST perform an access check to ensure that the calling client is authorized to perform that function.<57>

Methods in RPC Opnum Order

| Method                              | Description  |
|-------------------------------------|--|
| Opnum0NotUsedOnWire                 | Opnum: 0   |
| Opnum1NotUsedOnWire                 | Opnum: 1   |
| Opnum2NotUsedOnWire                 | Opnum: 2   |
| Opnum3NotUsedOnWire                 | Opnum: 3   |
| NtFrsApi_Rpc_Set_DsPollingIntervalW | Adjusts the interval at which Active Directory is polled for updates unless both <i>LongInterval</i> and <i>ShortInterval</i> are 0, and then MUST initiate a polling cycle.<br>Opnum: 4 |
| NtFrsApi_Rpc_Get_DsPollingIntervalW | Returns the current Active Directory polling intervals.<br>Opnum: 5  |
| Opnum6NotUsedOnWire                 | Opnum: 6   |

| Method                        | Description  |
|-------------------------------|--|
| NtFrsApi_Rpc_InfoW            | Returns internal information about FRS.<br>Opnum: 7                                  |
| NtFrsApi_Rpc_IsPathReplicated | Method not used by FRS.<br>Opnum: 8  |
| NtFrsApi_Rpc_WriterCommand    | Deactivates or reactivates the replication of the specified Replica Set.<br>Opnum: 9 |
| NtFrsApi_Rpc_ForceReplication | Triggers replication on the connection even if the schedule is off.<br>Opnum: 10     |

All methods MUST NOT throw exceptions.

### 3.2.4.1 NtFrsApi\_Rpc\_Set\_DsPollingIntervalW (Opnum 4)

The NtFrsApi\_Rpc\_Set\_DsPollingIntervalW method adjusts the interval at which Active Directory is polled (see section 3.1.5.1) for updates unless both *LongInterval* and *ShortInterval* are 0, and then MUST initiate a polling cycle.

```

unsigned long NtFrsApi_Rpc_Set_DsPollingIntervalW(
    [in] handle_t Handle,
    [in] unsigned long UseShortInterval,
    [in] unsigned long LongInterval,
    [in] unsigned long ShortInterval
);

```

**Handle:** Binding handle obtained when the Partner authenticates with the file replication service.

**UseShortInterval:** If nonzero, requests that the server initiate a polling cycle using the short interval. If 0, requests that the server initiate a polling cycle using the long interval.

| Value                      | Meaning                                |
|----------------------------|--|
| 0x00000000                 | Value in <i>LongInterval</i> is used.  |
| 0x00000001 —<br>0xFFFFFFFF | Value in <i>ShortInterval</i> is used. |

**LongInterval:** Long interval in minutes. A 0x00000000 value indicates that the *LongInterval* MUST NOT be modified by this call.<58>

**ShortInterval:** Short interval in minutes. A 0x00000000 value indicates that the *ShortInterval* MUST NOT be modified by this call. If both *ShortInterval* and *LongInterval* are set, then the minimum of both will be used as the *ShortInterval* value.<59>

**Return Values:** The method MUST return 0 on success or a nonzero error code on failure. All nonzero values MUST be treated as equivalent failures unless otherwise specified.

| Return value/code           | Description                        |
|-----------------------------|------------------------------------|
| 0x00000000<br>ERROR_SUCCESS | The method completed successfully. |



**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

**Validating the NtFrsApi\_Rpc\_Set\_DsPollingIntervalW request:** The server must validate the NtFrsApi\_Rpc\_Set\_DsPollingIntervalW request by performing the following checks.

- If **IsSetDsPollingIntervalAccessCheckEnabled** is set to "None", the server MUST fail the call with an implementation-specific value.
- If **IsSetDsPollingIntervalAccessCheckEnabled** is set to "Disabled", the server MUST process the request without performing an access check.
- If **IsSetDsPollingIntervalAccessCheckEnabled** is set to "Enabled", the server MUST check if the calling client is authorized to perform that function. If not, the server MUST return ERROR\_NOT\_AUTHENTICATED.
  - If **SetDsPollingIntervalAccessRequired** is set to "None", the server MUST fail the call with an implementation-specific value.
  - If **SetDsPollingIntervalAccessRequired** is set to "Read", the server MUST verify that the caller has read access. If the caller does not have read access, the server MUST return FRS\_ERR\_INSUFFICIENT\_PRIV.
  - If **SetDsPollingIntervalAccessRequired** is set to "Write", the server MUST verify that the caller has write access. If the caller does not have write access, the server MUST return FRS\_ERR\_INSUFFICIENT\_PRIV.

**Actions Triggered:** The local member MUST set the long and short polling intervals and kick off a new polling cycle. If both intervals are set, the new polling cycle MUST use the interval specified by UseShortInterval.

**Note** An Active Directory polling cycle MUST NOT be initiated if one is already in progress. An implementation can either return an error or wait for the in-progress polling cycle to complete.

### 3.2.4.2 NtFrsApi\_Rpc\_Get\_DsPollingIntervalW (Opnum 5)

The NtFrsApi\_Rpc\_Get\_DsPollingIntervalW method MUST return the current Active Directory polling intervals.

Provides the current, long, and short polling intervals. All polling intervals are in minutes.

```
unsigned long NtFrsApi_Rpc_Get_DsPollingIntervalW(
    [in] handle_t Handle,
    [out] unsigned long* Interval,
    [out] unsigned long* LongInterval,
    [out] unsigned long* ShortInterval
);
```

**Handle:** Binding handle obtained when the partner authenticates with the file replication service.

**Interval:** Current interval in minutes, which MUST be the same value as either LongInterval or ShortInterval (see section 3.1.2).

**LongInterval:** Long interval in minutes.

**ShortInterval:** Short interval in minutes.

**Return Values:** The method MUST return 0 on success or a nonzero error code on failure. All nonzero values MUST be treated as equivalent failures unless otherwise specified.

| Return value/code           | Description                        |
|-----------------------------|------------------------------------|
| 0x00000000<br>ERROR_SUCCESS | The method completed successfully. |

**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

**Validating the `_Rpc_Get_DsPollingIntervalW` request:** The server must validate the `NtFrsApi_Rpc_Get_DsPollingIntervalW` request by performing the following checks.

- If **`IsGetDsPollingIntervalAccessCheckEnabled`** is set to "None", the server MUST fail the call with an implementation-specific value.
- If **`IsGetDsPollingIntervalAccessCheckEnabled`** is set to "Disabled", the server MUST process the request without performing an access check.
- If **`IsGetDsPollingIntervalAccessCheckEnabled`** is set to "Enabled", the server MUST check whether the calling client is authorized to perform that function. If the client is not authorized to perform that function, the server MUST return `ERROR_NOT_AUTHENTICATED`.
- If **`GetDsPollingIntervalAccessRequired`** is set to "None", the server MUST fail the call with an implementation-specific value.
- If **`GetDsPollingIntervalAccessRequired`** is set to "Read", the server MUST verify that the caller has read access. If the caller does not have read access, the server MUST return `FRS_ERR_INSUFFICIENT_PRIV`.
- If **`GetDsPollingIntervalAccessRequired`** is set to "Write", the server MUST verify that the caller has write access. If the caller does not have write access, the server MUST return `FRS_ERR_INSUFFICIENT_PRIV`.

### 3.2.4.3 `NtFrsApi_Rpc_InfoW` (Opnum 7)

The `NtFrsApi_Rpc_InfoW` method MUST return internal information. The method is not used in server-to-server interoperation. The vendor MUST fill the BLOB with implementation-dependent data structures.

```

unsigned long NtFrsApi_Rpc_InfoW(
    [in] handle_t Handle,
    [in, range(0,65536)] unsigned long BlobSize,
    [in, out, size_is(BlobSize), unique]
    unsigned char* Blob
);

```

**Handle:** Binding handle obtained when the partner authenticates with the file replication service (FRS).

**BlobSize:** The size of the `Blob` parameter in bytes. MUST be one of the following values, to fit the returned information in the `Blob`.

| Value  | Meaning            |
|--|--------------------|
| <code>NTFRSAPI_DEFAULT_INFO_SIZE</code><br>(64 * 1024) | Default info size. |
| <code>NTFRSAPI_MINIMUM_INFO_SIZE</code>                | Minimum info size. |

| Value       | Meaning |
|-------------|---------|
| ( 1 * 1024) |         |

**Blob:** Custom binary format object that contains internal information about FRS. Vendors MUST choose their own implementation-dependent data structures for the *Blob* parameter to dump debug/tracing-related information. This protocol does not require a specific format for the information in this BLOB. This structure is not used for server-to-server interoperability, and the vendor can fill the BLOB with implementation-dependent data structures. <60><61>

**Return Values:** The method MUST return 0 on success or a nonzero error code on failure. All nonzero values MUST be treated as equivalent failures unless otherwise specified.

| Return value/code           | Description                        |
|-----------------------------|------------------------------------|
| 0x00000000<br>ERROR_SUCCESS | The method completed successfully. |

**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

**Validating the `_NtFrsApi_Rpc_InfoW` request:** The server must validate the `NtFrsApi_Rpc_InfoWrequest` request by performing the following checks.

- If **IsGetInfoWAccessCheckEnabled** is set to "None", the server MUST fail the call with an implementation specific value.
- If **IsGetInfoWAccessCheckEnabled** is set to "Disabled", the server MUST process the request without performing an access check.
- If **IsGetInfoWAccessCheckEnabled** is set to "Enabled", the server MUST check whether the calling client is authorized to perform that function. If the calling client is not authorized to perform that function, it MUST return `ERROR_NOT_AUTHENTICATED`.
- If **GetInfoWAccessRequired** is set to "None", the server MUST fail the call with an implementation specific value.
- If **GetInfoWAccessRequired** is set to "Read", the server MUST check whether the caller has read access. If the caller does not have read access, the server MUST return `FRS_ERR_INSUFFICIENT_PRIV`.
- If **GetInfoWAccessRequired** is set to "Write", the server MUST check whether the caller has write access. If the caller does not have read access, the server MUST return `FRS_ERR_INSUFFICIENT_PRIV`.
- If the server is not able to verify the parameters inside blob, then it MUST fail the call with an implementation-defined failure value `FRS_ERR_INVALID_SERVICE_PARAMETER`.

**Actions Triggered:** FRS MUST return the internal information about FRS. The information returned MUST be used for dumping FRS internal implementation-specific information for debugging purposes. It MUST NOT be used in initial sync, normal sync, or in any way that affects the FRS Protocol behavior. The vendor can implement this RPC call to dump the vendor's implementation specific info. However, the structure of this information is implementation-specific and has no bearing on the operations of the protocol. <62>

### 3.2.4.4 `NtFrsApi_Rpc_IsPathReplicated` (Opnum 8)

This method is not used by FRS.

```

unsigned long NtFrsApi_Rpc_IsPathReplicated(
    [in] handle_t Handle,
    [in, string, unique] PWCHAR Path,
    [in] unsigned long ReplicaSetTypeOfInterest,
    [out] unsigned long* Replicated,
    [out] unsigned long* Primary,
    [out] unsigned long* Root,
    [out] GUID* ReplicaSetGuid
);

```

**Handle:** Binding handle obtained when the partner authenticates with the file replication service (FRS).

**Path:** Local path being queried.<63>

**ReplicaSetTypeOfInterest:** Replica set type identified by the path. The value MUST be one of the following.

| Value                                      | Meaning  |
|--|--|
| 0x00000000                                 | Indicates any replica set.   |
| FRS_RSTYPE_ENTERPRISE_SYSVOL<br>0x00000001 | Indicates the replica set for the enterprise system volume (SYSVOL).                   |
| FRS_RSTYPE_DOMAIN_SYSVOL<br>0x00000002     | Indicates the replica set for the domain SYSVOL.                                       |
| FRS_RSTYPE_DFS<br>0x00000003               | Indicates the replica set for the distributed file system, as specified in [MS-DFSNM]. |
| FRS_RSTYPE_OTHER<br>0x00000004             | Indicates none of the previous types.  |

**Replicated:** Boolean value that indicates if the replica set is replicated by the domain controller (DC). The value MUST be one of the following.

| Value               | Meaning  |
|---------------------|--|
| FALSE<br>0x00000000 | The <i>Path</i> parameter is not part of a replica set of the type specified in <i>ReplicaSetTypeOfInterest</i> . If this value is returned, <i>Primary</i> , <i>Root</i> , and <i>ReplicaSetGuid</i> are set to NULL. |
| TRUE<br>0x00000001  | The <i>Path</i> parameter is part of a replica set of the type specified in <i>ReplicaSetTypeOfInterest</i> .  |

**Primary:** Indicates if the computer is the first computer for the replica set for initial sync. The value MUST be one of the following.

| Value      | Meaning  |
|------------|--|
| 0x00000000 | Current computer is not the primary computer for the matching replica set. |
| 0x00000001 | Current computer is the primary computer for the matching replica set.     |
| 0x00000002 | Matching replica set does not have a primary computer.                     |

**Root:** A Boolean value that indicates if the path is the replica tree root for the replica set. The value MUST be one of the following.

| Value               | Meaning  |
|---------------------|--|
| FALSE<br>0x00000000 | The <i>Path</i> parameter is not the replica tree root for the matching replica set. |
| TRUE<br>0x00000001  | The <i>Path</i> parameter is the replica tree root for the matching replica set.     |

**ReplicaSetGuid:** GUID for the matching replica set.

**Return Values:** The method MUST return 0 on success or a nonzero error code on failure. All nonzero values MUST be treated as equivalent failures unless otherwise specified.

| Return value/code           | Description                        |
|-----------------------------|------------------------------------|
| 0x00000000<br>ERROR_SUCCESS | The method completed successfully. |

**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

**Validating the NtFrsApi\_Rpc IsPathReplicated request:** The server must validate the NtFrsApi\_Rpc IsPathReplicated request by performing the following checks.

- If **IsPathReplicatedCheckEnabled** is set to "None", the server MUST fail the call with an implementation-specific value.
- If **IsPathReplicatedCheckEnabled** is set to "Disabled", the server MUST process the request without performing an access check.
- If **IsPathReplicatedCheckEnabled** is set to "Enabled", the server MUST check whether the calling client is authorized to perform that function. If the calling client is not authorized to perform that function, the server MUST return ERROR\_NOT\_AUTHENTICATED.
  - If **ReplicatedPathAccessRequired** is set to "None", the server MUST fail the call with an implementation-specific value.
  - If **ReplicatedPathAccessRequired** is set to "Read", the server MUST check whether the caller has read access. If the caller does not have read access, the server MUST return FRS\_ERR\_INSUFFICIENT\_PRIV.
  - If **ReplicatedPathAccessRequired** is set to "Write", the server MUST check whether the caller has write access. If the caller does not have write access, the server MUST return FRS\_ERR\_INSUFFICIENT\_PRIV.
- If the server is not able to verify the passed in parameters, then it MUST ignore the failure and succeed the call.

**Actions Triggered:** The NtFrsApi\_Rpc\_IsPathReplicated method MUST return a value that specifies if a Path is part of a replica set of the type specified in the ReplicaSetTypeOfInterest parameter. The method is not used in server-to-server interoperation.

### 3.2.4.5 NtFrsApi\_Rpc\_WriterCommand (Opnum 9)

The NtFrsApi\_Rpc\_WriterCommand method MUST deactivate or reactivate the replication of the specified Replica Set.

```

unsigned long NtFrsApi_Rpc_WriterCommand(
    [in] handle_t Handle,
    [in] unsigned long Command
);

```

**Handle:** RPC binding handle obtained when the partner authenticates with the file replication service (FRS), as specified in [MS-RPCE]. For RPC binding handle transport, see section 2.1.

**Command:** A 32-bit unsigned integer that indicates if the FRS can install new files. It MUST be one of the following.

| Value  | Meaning  |
|--|--|
| NTFRSAPI_WRITER_COMMAND_FREEZE<br>0x00000001 | Prevent the FRS from installing new files. FRS MUST continue sending/receiving/processing updates. However, the updates received will not be installed to the file system unless NTFRSAPI_WRITER_COMMAND_THAW is received. |
| NTFRSAPI_WRITER_COMMAND_THAW<br>0x00000002   | Allow the FRS to install new files.  |

**Return Values:** The method MUST return 0 on success or a nonzero error code on failure. All nonzero values MUST be treated as equivalent failures unless otherwise specified.

| Return value/code           | Description                        |
|-----------------------------|------------------------------------|
| 0x00000000<br>ERROR_SUCCESS | The method completed successfully. |

**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

**Validating the NtFrsApi\_Rpc WriterCommand request:** The server must validate the NtFrsApi\_Rpc WriterCommand request by performing the following checks.

- If **IsWriterCommandCheckEnabled** is set to "None", the server MUST fail the call with an implementation specific value.
- If **IsWriterCommandCheckEnabled** is set to "Disabled", the server MUST process the request without performing an access check.
- If **IsWriterCommandCheckEnabled** is set to "Enabled", the server MUST check whether the calling client is authorized to perform that function. If the calling client is not authorized to perform that function, the server MUST return ERROR\_NOT\_AUTHENTICATED.
  - If **WriterCommandAccessRequired** is set to "None", the server MUST fail the call with an implementation-specific value.
  - If **WriterCommandAccessRequired** is set to "Read", the server MUST check whether the caller has read access. If the caller does not have read access, the server MUST return FRS\_ERR\_INSUFFICIENT\_PRIV.
  - If **WriterCommandAccessRequired** is set to "Write", the server MUST check whether the caller has write access. If the caller does not have write access, the server MUST return FRS\_ERR\_INSUFFICIENT\_PRIV.
- If the server is not able to verify the passed-in parameters, then it MUST ignore the failure and succeed the call.

**Actions Triggered:** If the command is NTFRSAPI\_WRITER\_COMMAND\_FREEZE, FRS MUST stop installing new files for all replica sets. If the command is NTFRSAPI\_WRITER\_COMMAND\_THAW, FRS MUST resume installing new files. When FRS is frozen, the Partner continues to participate in replication, change orders are passed between partners, and staging files are transmitted to and from the partner. The staging files MUST remain in the staged state as long as FRS is frozen. Staging files can only be installed into the replica tree when FRS is in the thawed state.

### 3.2.4.6 NtFrsApi\_Rpc\_ForceReplication (Opnum 10)

This call MUST trigger replication on the connection even if the schedule is off.

```
unsigned long NtFrsApi_Rpc_ForceReplication(
    [in] handle_t Handle,
    [in, unique] GUID* ReplicaSetGuid,
    [in, unique] GUID* CxtionGuid,
    [in, string, unique] PWCHAR ReplicaSetName,
    [in, string, unique] PWCHAR PartnerDnsName
);
```

**Handle:** The binding handle that is obtained when the partner authenticated with the file replication service (FRS).

**ReplicaSetGuid:** A pointer to the globally unique identifier (GUID) of the replica set.

**CxtionGuid:** A pointer to the GUID of the upstream partner connection.

**ReplicaSetName:** The replica set name.

**PartnerDnsName:** The fully qualified domain name of the upstream partner.

**Return Values:** The method MUST return 0 on success or a nonzero error code on failure. All nonzero values MUST be treated as equivalent failures unless otherwise specified.

| Return value/code           | Description                        |
|-----------------------------|------------------------------------|
| 0x00000000<br>ERROR_SUCCESS | The method completed successfully. |

**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

**Validating the \_NtFrsApi\_Rpc ForceReplication request:** The server must validate the NtFrsApi\_Rpc ForceReplication request by performing the following checks.

- If **IsForceReplicationCheckEnabled** is set to "None", the server MUST fail the call with an implementation-specific value.
- If **IsForceReplicationCheckEnabled** is set to "Disabled", the server MUST process the request without performing an access check.
- If **IsForceReplicationCheckEnabled** is set to "Enabled", the server MUST check whether the calling client is authorized to perform that function. If the caller is not authorized to perform that function, the server MUST return ERROR\_NOT\_AUTHENTICATED.
  - If **ForceReplicationAccessRequired** is set to "None", the server MUST fail the call with an implementation-specific value.

- If **ForceReplicationAccessRequired** is set to "Read", the server MUST check whether the caller has read access. If the caller does not have read access, the server MUST return `FRS_ERR_INSUFFICIENT_PRIV`.
- If **ForceReplicationAccessRequired** is set to "Write", the server MUST check whether the caller has write access. If the caller does not have write access, the server MUST return `FRS_ERR_INSUFFICIENT_PRIV`.
- If the server is not able to verify the passed in parameters then it MUST ignore the failure and succeed the call.

**Actions Triggered:** The `NtFrsApi_Rpc_ForceReplication` method MUST force replication to occur outside the scheduled replication activity.

### 3.2.5 Timer Events

No other timer events.

### 3.2.6 Other Local Events

No other local events are required.

## 3.3 FRSRPC Interface

The FRSRPC interface provides methods that support the initial replication activities needed when a server is promoted to a domain controller as well as subsequent replication activities. The version for this interface is 1.1.

To receive incoming remote calls for this interface, a server MUST use the port specified in the `FRSStaticPort` element, if it is not set to 0. Otherwise, the server MUST implement the service at a dynamically assigned endpoint. The server MUST use the UUID `F5CC59B4-4264-101A-8C59-08002B2F8426`. NTFRS MUST be layered above TCP/IP and RPC.

### 3.3.1 Abstract Data Model

No other abstract data models required.

### 3.3.2 Timers

#### 3.3.2.1 Connection Schedule Timer

The replication schedule defines the periods during which replication takes place.

Both DFS and SYSVOL connection schedules are stored in an attribute associated with each NTFRS Replica Set object (see section 2.3.1.2) and with each NTDS connection object (see section 2.3.1.6). However, SYSVOL replica set connections are using a trigger schedule (see section 3.3.2.1.1) while DFS replica set connections are using an on/off schedule (see section 3.3.2.1.2).

Replication schedules MUST be stored in Coordinated Universal Time (UTC).

##### 3.3.2.1.1 SYSVOL Connection ScheduleTimer

FRS MUST replicate SYSVOL using the same intra-site connection objects and schedule (section 2.3.2.4) built by the DRSR, as specified in [MS-DRSR], for Active Directory replication.

FRS uses two types of replication schedules for SYSVOL:



- SYSVOL connection within a site. The connection is always considered to be on; any schedule MUST be ignored and the changed files replicated immediately.
- SYSVOL connection between sites. SYSVOL replication is initiated between two inter-site members at the start of the 15-minute interval, assuming the schedule is open. The connection MUST be using a trigger schedule. When a triggering schedule is used, the upstream partner ignores its schedule and responds to any request by the downstream partner. When the schedule closes, the upstream partner unjoins the connection only after the current contents of the outbound log, at the time of join, have been sent and acknowledged. The SYSVOL replication schedule can be configured so that replication takes place four, two, one, or zero times per hour:
  - When set to four times per hour, replication MUST start at 0:00, 0:15, 0:30, and 0:45. This schedule is essentially continuous replication. 0:00 means exactly on the hour.
  - When set to two times per hour, replication MUST occur at 15-minute intervals starting at 0:15 and 0:45.
  - When set to one time per hour, replication MUST start at 0:00 and ends at 0:15.
  - When set to zero times per hour, replication MUST NOT start.

### 3.3.2.1.2 DFS Connection Schedule

When replication is set to be available in the schedule, replication MUST occur continuously. Unlike SYSVOL schedules, DFS replica set schedules are on/off schedules. As long as the schedule is open, the connection stays in the Joined state and change orders MUST be sent out as they are processed by the Upstream Partner. Any queued change orders in the upstream partner's outbound log MUST be sent first. When the schedule closes, FRS unjoins the connection.

### 3.3.3 Initialization

FRS MUST check Active Directory for replica set configurations (section 2.3). FRS MUST check if it is the primary member of the replica set by checking the `frsPrimaryMember` of the NTFRS replica set object (section 2.3.1.2):

- If FRS is the primary member of the replica set, FRS MUST scan the local replica tree to add each folder and file to the IDTable. FRS MUST be ready to populate all its downstream partners based on its IDTable.
- If FRS is not the primary member of the replica set, FRS MUST NOT scan the local replica tree. Instead, FRS MUST get its files/folders from upstream partners through inbound connections. When FRS has received all files/folders, FRS is done with the initial synchronization. FRS MUST then start monitoring its local replica tree for changes to be added to its IDTable after the initial synchronization. FRS MUST be ready to populate its downstream partners based on its latest IDTable.

Based on the inbound/outbound connection schedule (section 3.3.2.1), FRS MUST establish connections (section 3.3.4.6) with its upstream partners and downstream partners. On each outbound connection, when initial synchronization is done, FRS MUST construct a change order for each unknown file or folder in its IDTable to be sent to corresponding downstream partners. See `VVJoin` in section 3.3.4.4.4.1 through `FrsSendCommPkt` when the schedule for the connection is open (section 3.3.4.4).

### 3.3.4 Message Processing Events and Sequencing Rules

Methods in RPC Opnum Order

| Method                      | Description  |
|-----------------------------|--|
| FrsRpcSendCommPkt           | Transfers change order commands, files, and folders on an FRS connection.<br>Opnum: 0  |
| FrsRpcVerifyPromotionParent | Perform no actions and always returns ERROR_CALL_NOT_IMPLEMENTED.<br>Opnum: 1  |
| FrsRpcStartPromotionParent  | Requests that a server start up a volatile connection for the purpose of seeding the system volume (SYSVOL) folder on the container.<br>Opnum: 2 |
| FrsNOP                      | Verifies that the RPC connection for this interface is functioning properly.<br>Opnum: 3   |
| Opnum4NotUsedOnWire         | Reserved for local use.<br>Opnum: 4  |
| Opnum5NotUsedOnWire         | Reserved for local use.<br>Opnum: 5  |
| Opnum6NotUsedOnWire         | Reserved for local use.<br>Opnum: 6  |
| Opnum7NotUsedOnWire         | Reserved for local use.<br>Opnum: 7  |
| Opnum8NotUsedOnWire         | Reserved for local use.<br>Opnum: 8  |
| Opnum9NotUsedOnWire         | Reserved for local use.<br>Opnum: 9  |
| Opnum10NotUsedOnWire        | Reserved for local use.<br>Opnum: 10   |

All methods MUST NOT throw exceptions.

### 3.3.4.1 Change Orders

When a change happens in the local file system for the replica tree, the upstream partner MUST construct a change order to be sent to all downstream partners through FrsSendCommPkt.

The following are the six types of changes that the upstream partner MUST monitor.

- File is added or updated (section 3.3.4.1.1).
- File is removed (section 3.3.4.1.2).
- File is renamed (section 3.3.4.1.3).
- Folder is created or updated (section 3.3.4.1.4).
- Folder is removed (section 3.3.4.1.5).
- Folder is renamed (section 3.3.4.1.6).

This section specifies the common details for these six types of changes.

For each type of change, the upstream partner MUST call FrsRpcSendCommPkt to send a COMM\_PACKET object to a downstream partner on each outbound connection.

The COMM\_PACKET object MUST be constructed as specified in section 2.2.3.5 with the following values.

**COMM\_BOP:** MUST be 0.

**COMM\_COMMAND:** 0x218 (CMD\_REMOTE\_CO).

**COMM\_TO:**

- **GUID:** MUST be the ObjectGuid of the downstream partner's NTFRS member object (see section 2.3.2.3).
- **Name:** SHOULD<64> be the FQDN of the downstream partner. <65>

**COMM\_FROM:**

- **GUID:** SHOULD be the objectGuid of the local member's NTFRS member object. <66>
- **Name:** SHOULD<67> be the FQDN of the local member. <68>

**COMM\_REPLICA:**

- **GUID:** MUST be the objectGuid of the downstream partner's NTFRS member object.
- **Name:** SHOULD be the name of the NTFRS replica set object. <69>

**COMM\_CXTION:**

- **GUID:** MUST be the outbound connection GUID
- **Name:** SHOULD be the downstream partner's DNS hostname. <70>

**COMM\_JOIN\_GUID:** MUST be the GUID of the current connection session (see section 3.1.1.5).

**COMM\_LAST\_JOIN\_TIME:** MUST be the Join time of the current successful connection session.

**COMM\_REMOTE\_CO:**

- **SequenceNumber:** A 32-bit, unsigned integer that indicates the sequence number in the change order command. It MUST be initialized to 0. The sequence number is incremented with each change order that goes into the outbound log. The sequence number is unique per replica set per machine.
- **State:** MUST be 0x14. (Request outbound propagation.)
- **Flags:** Specified in sections 3.3.4.2 through 3.3.4.6.
- **IFlags:** Specified in sections 3.3.4.2 through 3.3.4.6.
- **ContentCmd:** Specified in sections 3.3.4.2 through 3.3.4.6.
- **LocationCmd:** Specified in sections 3.3.4.2 through 3.3.4.6.
- **FileAttributes:** File attributes, as specified in [MS-FSCC].
- **FileVersionNumber:** The number of times that this file is closed after being modified. It is used in conflict resolution.
- **PartnerAckSeqNumber:** MUST be the same as SequenceNumber.

- **FileSize:** For files, MUST be the file size. For folders, it MUST be 0. <71>
- **FileOffset:** MUST be 0.
- **FrsVsn:** MUST be the volume sequence number (VSN) for the file.
- **FileUsn:** A 64-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member. This value is only meaningful on the current replica member and has no meaning on any other replica member.
- **JrnlUsn:** A 64-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member. This value is only meaningful on the current replica member and has no meaning on any other replica member.
- **JrnlFirstUsn:** A 64-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member. This value is only meaningful on the current replica member and has no meaning on any other replica member.
- **OriginalReplicaNum:** A 32-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member. This value is only meaningful on the current replica member and has no meaning on any other replica member.
- **NewReplicaNum:** A 32-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member. This value is only meaningful on the current replica member and has no meaning on any other replica member.
- **ChangeOrderGuid:** MUST be the GUID of the change order.
- **OriginatorGuid:** MUST be the originator GUID.
- **FileGuid:** MUST be the file GUID in the IDTable.
- **OldParentGuid:** MUST be the parent GUID.
- **NewParentGuid:** MUST be the parent GUID.
- **CxtionGuid:** MUST be the connection GUID.
- **AckVersion:** A 64-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member. This value is only meaningful on the current replica member and has no meaning on any other replica member. After the change order is sent to another replica member, the receiving partner MUST NOT update this field.
- **Extension:** MUST be ignored.
- **EventTime:** MUST be the time stamp when the change happened in FILETIME format. <72>
- **FileNameLength:** MUST be the byte count of the FileName.
- **FileName:** MUST be the file name.
- **Spare2Ull:** MUST be 0.
- **Spare1Guid:** MUST be 0.
- **Spare2Guid:** MUST be 0.
- **Spare1Wcs:** Must be 0
- **Spare2Wcs:** MUST be 0.
- **Spare2Bin:** MUST be 0.

**COMM\_CO\_EXTENSION\_2:** Constructed as specified in section 2.2.3.6.22.

**COMM\_EOP:** MUST be 0xffffffff.

### 3.3.4.1.1 File Is Added/Updated on the Upstream Partner

**Flags:**

If the staging file is not compressed: MUST be a bitwise OR of the following flags.

- CO\_FLAG\_LOCALCO if originating on the upstream partner.
- CO\_FLAG\_LOCATION\_CMD if creating a file.
- CO\_FLAG\_CONTENT\_CMD if creating a file that is non zero in size.

If the staging file is compressed: MUST be a bitwise OR of the following flags.

- CO\_FLAG\_COMPRESSED\_STAGE
- CO\_FLAG\_LOCALCO if originating on the upstream partner.
- CO\_FLAG\_LOCATION\_CMD if creating a file.
- CO\_FLAG\_CONTENT\_CMD if creating a file that is non zero in size.

**IFlags:** MUST be 0.

**ContentCmd:** MUST be a bitwise OR of zero or more of the following flags based on the nature of the update to the file.

- REASON\_BASIC\_INFO\_CHANGE
- REASON\_DATA\_EXTEND
- REASON\_DATA\_OVERWRITE

**LocationCmd:**

Creating a new file: MUST be 0. (CO\_LOCATION\_CREATE on the file.)

Updating an existing file: LocationCmd is not needed. MUST be 0xE. (CO\_LOCATION\_NO\_CMD on file.)

For an example, see section 4.5.1. <73>

### 3.3.4.1.2 File Is Removed on the Upstream Partner

**Flags:** MUST be a bitwise OR of the following flags.

- CO\_FLAG\_LOCALCO if originating on the upstream partner.
- CO\_FLAG\_LOCATION\_CMD

**IFlags:** MUST be 0.

**ContentCmd:** MUST be 0.

**LocationCmd:** MUST be 0x2. (CO\_LOCATION\_DELETE on file.)

For an example, see section 4.5.3. <39>

### 3.3.4.1.3 File Is Renamed on the Upstream Partner

**Flags:**

If the staging file is not compressed: MUST be a bitwise OR of the following flags.

- CO\_FLAG\_LOCALCO if originating on the upstream partner.
- CO\_FLAG\_CONTENT\_CMD

If the staging file is compressed: MUST be a bitwise OR of the following flags.

- CO\_FLAG\_COMPRESSED\_STAGE
- CO\_FLAG\_LOCALCO if originating on the upstream partner.
- CO\_FLAG\_CONTENT\_CMD

**IFlags:** MUST be 0.

**ContentCmd:** MUST be 0x2000. (REASON\_RENAME\_NEW\_NAME)

**LocationCmd:** MUST be 0x000E. (CO\_LOCATION\_NO\_CMD on the file.)

For an example of renaming a file, see section 4.5.2.

### 3.3.4.1.4 Folder Is Created/Updated on the Upstream Partner

**Flags:**

If the staging file is not compressed: MUST be a bitwise OR of the following flags.

- CO\_FLAG\_LOCALCO if originating on the upstream partner.
- CO\_FLAG\_LOCATION\_CMD if creating a folder.

If the staging file is compressed: MUST be a bitwise OR of the following flags.

- CO\_FLAG\_COMPRESSED\_STAGE
- CO\_FLAG\_LOCALCO if originating on the upstream partner.
- CO\_FLAG\_LOCATION\_CMD if creating a folder.
- CO\_FLAG\_CONTENT\_CMD if folder is updated.

**IFlags:** MUST be 0.

**ContentCmd:** MUST be a bitwise OR of zero or more of the following flags based on the nature of the update to the folder.

- REASON\_BASIC\_INFO\_CHANGE

**LocationCmd:**

Creating a folder: MUST be 1. (CO\_LOCATION\_CREATE on folder.)

Updating an existing folder: MUST be 0xF. (CO\_LOCATION\_NO\_CMD on folder.)

For an example, see section 4.5.4. <74>

### 3.3.4.1.5 Folder Is Removed on the Upstream Partner

**Flags:** MUST be a bitwise OR of the following flags.

- CO\_FLAG\_LOCALCO if originating on the upstream partner.
- CO\_FLAG\_LOCATION\_CMD

**IFlags:** MUST be 0.

**ContentCmd:** MUST be 0.

**LocationCmd:** MUST be 0x3. (CO\_LOCATION\_DELETE on folder.)

For an example, see section 4.4.5. <75>

### 3.3.4.1.6 Folder Is Renamed on the Upstream Partner

#### Flags:

If the staging file is not compressed: MUST be a bitwise OR of the following flags.

- CO\_FLAG\_LOCALCO if originating on the upstream partner.
- CO\_FLAG\_CONTENT\_CMD

If the staging file is compressed: MUST be a bitwise OR of the following flags.

- CO\_FLAG\_COMPRESSED\_STAGE
- CO\_FLAG\_LOCALCO if originating on the upstream partner.
- CO\_FLAG\_CONTENT\_CMD

**IFlags:** MUST be 0.

**ContentCmd:** MUST be 0x2000. (REASON\_RENAME\_NEW\_NAME)

**LocationCmd:** MUST be 0xF. (CO\_LOCATION\_NO\_CMD on the folder.)

For an example of renaming an empty folder, see section 4.5.6.

### 3.3.4.2 FrsRpcStartPromotionParent Message (Opnum 2)

The FrsRpcStartPromotionParent method requests the server that is the Upstream Partner to start a volatile connection for the purpose of seeding the system volume (SYSVOL) folder on the container.

```

unsigned long FrsRpcStartPromotionParent(
    [in] handle_t Handle,
    [in, string, unique] PWCHAR ParentAccount,
    [in, string, unique] PWCHAR ParentPassword,
    [in, string, unique] PWCHAR ReplicaSetName,
    [in, string, unique] PWCHAR ReplicaSetType,
    [in, string, unique] PWCHAR CxtionName,
    [in, string, unique] PWCHAR PartnerName,
    [in, string, unique] PWCHAR PartnerPrincName,
    [in] unsigned long PartnerAuthLevel,
    [in, range(sizeof(GUID),sizeof(GUID))]
        unsigned long GuidSize,
    [in, size_is(GuidSize), unique]
        unsigned char* CxtionGuid,
    [in, size_is(GuidSize), unique]
        unsigned char* PartnerGuid,
    [in, out, size_is(GuidSize), unique]
        unsigned char* ParentGuid
);

```

**Handle:** Binding handle obtained when the partner authenticates with the file replication service.

**ParentAccount:** Valid account on the DC that functions as the parent. MUST be NULL (indicating to use the current connecting client's credential).

**ParentPassword:** Password for an account on the DC. MUST be NULL.

**ReplicaSetName:** RDN name of the replica set that the caller requested.

**ReplicaSetType:** String that specifies the type of the replica set that the caller requested. The string MUST be treated in a case-insensitive manner, and there MUST NOT be white space in the string. The value MUST be one of the following.

| Value        | Meaning                 |
|--------------|-------------------------|
| "Enterprise" | Enterprise Replica Set. |
| "Domain"     | Domain Replica Set.     |

**CxtionName:** Fully qualified domain name of the machine receiving this call.

**PartnerName:** Fully qualified domain name of the caller.

**PartnerPrincName:** Name of the caller to use for Kerberos authentication in the format of [Domain Name]\[Host Name]\$.

**PartnerAuthLevel:** Authentication type and level. MUST be one of the following values. <76>

| Value      | Meaning            |
|------------|--------------------|
| 0x00000000 | Encrypted Kerberos |
| 0x00000001 | No authentication  |

**GuidSize:** Size of the array allocated to hold the globally unique identifier (GUID). MUST be 0x00000010.

**CxtionGuid:** Temporary value for the connection. The GUID is generated randomly and is not persisted in Active Directory.

**PartnerGuid:** GUID that identifies the replica set on the replication partner.

**ParentGuid:** GUID value that identifies the parent for the inbound connection.

**Return Values:** The method MUST return 0 on success; otherwise, it MUST return a nonzero error code, as defined by the vendor.

To create a \SYSVOL volume (required for the server to operate as a DC), the FrsRpcStartPromotionParent method requests a volatile connection on the server for the purpose of seeding a \SYSVOL folder. This is necessary because of the following: During DC promotion, the newly promoted DC creates a set of Active Directory objects, which includes the connection objects. When the newly promoted DC is trying to replicate the contents from another DC in the domain (the process is called initial sync), the other DC might not know the newly created connection, either because the DS replication service has not replicated the connection out to the other DC or because the other DC has not polled the Directory Service yet. The newly created DC creates a volatile connection for only the initial sync and calls this RPC function to inform the other DC about this volatile connection. In this way, the other DC does not reject the initial sync request. <77> Once the initial sync is completed, NTFRS starts using the connection generated by Knowledge Consistency Checker (KCC) for normal synchronization.



On receiving this message, the server MUST validate the following:

- Verify that the current local member is really a DC by querying the local member's computer object in the domain, as specified in [MS-ADSC] section 2.21.
- Verify that the client MUST be really a DC by querying the client's computer object in the domain, as specified in [MS-ADSC] section 2.21.
- Verify GuidSize MUST be 0x10.
- Verify ReplicaSetName is not NULL and MUST exist on the server.
- Verify ReplicaSetType MUST be either Enterprise or Domain.
- Verify CxtionName MUST NOT be NULL.
- Verify PartnerName MUST NOT be NULL.
- Verify PartnerPrincName MUST NOT be NULL.
- Verify CxtionGuid MUST NOT be NULL.
- Verify PartnerGuid MUST NOT be NULL.
- Verify ParentGuid MUST NOT be NULL.
- Verify PartnerAuthLevel MUST be either 0 (Encrypted Kerberos) or 1 (no authentication).

If the parameter validation fails, the server MUST fail the operation with ERROR\_INVALID\_PARAMETER, as specified in [MS-ERREF] section 2.2; otherwise, the server MUST do the following:

- Find the local replica as identified by ReplicaSetName by reading the Active Directory objects.
- Set the local replica's GUID as ParentGuid.
- Create a new connection in the local replica with the following parameters:
  - Connection direction MUST be set to Outbound.
  - Mark the connection to be volatile.
  - Connection name MUST be set to CxtionGuid.
  - Connection GUID MUST be set to CxtionGuid.
  - Connection partner name MUST be set to PartnerName.
  - Connection partner GUID MUST be set to PartnerGuid.
  - Connection state MUST be set to unjoined state.
  - The first time promotion for SYSVOL MUST wait for connection establishment request (section 3.3.4.4.4).

### 3.3.4.3 FrsNOP Message (Opnum 3)

The FrsNOP method verifies that the RPC connection for this interface is functioning properly. This method is not used by FRS.

```
unsigned long FrsNOP();
```

This method has no parameters.

| Return value/code | Description               |
|-------------------|---------------------------|
| 0x00000000        | The method MUST return 0. |

On receiving this message, the server MUST do nothing and return 0.

### 3.3.4.4 FrsRpcSendCommPkt (Opnum 0)

The FrsRpcSendCommPkt method transfers change order commands, files, and folders on an FRS connection.

```
unsigned long FrsRpcSendCommPkt(  
    [in] handle_t Handle,  
    [in] PCOMM_PACKET CommPkt  
);
```

**Handle:** Binding handle obtained when the partner authenticates with the FRS.

**CommPkt:** Command and data sent from the replication partner.

**Return Values:** The method MUST return 0 on success or a nonzero error code on failure. For protocol purposes, all nonzero values MUST be treated as equivalent failures unless otherwise specified.

| Return value/code           | Description                        |
|-----------------------------|------------------------------------|
| 0x00000000<br>ERROR_SUCCESS | The method completed successfully. |

**Exceptions Thrown:** No exceptions are thrown beyond those thrown by the underlying RPC protocol [MS-RPCE].

FRS uses FrsRpcSendCommPkt to send COMM\_PACKET objects (see section 2.2.3.5) between replica members to replicate files and folders inside a Replica Set. See sections 4.1 and 4.2.

In the following section, these two sentences have the same meaning:

- Machine A sends a COMM\_PACKET to machine B.
- Machine A calls the FrsRpcSendCommPkt method on machine B.

There are 12 types of COMM\_PACKETs (see section 2.2.3.6.2). If machine A sends a COMM\_PACKET packet to machine B with COMM\_PACKET.COMM\_COMMAND set to CMD\_XXXX, a simple way to describe this is as follows: Machine A sends CMD\_XXXX to machine B.

#### 3.3.4.4.1 Common Details

On receiving an FrsRpcSendCommPkt call, the server MUST validate the COMM\_PACKET object.

Major attribute MUST be 0.

MUST start with COMM\_BOP and end with COMM\_EOP.

CsId MUST be 1.

Command MUST be a valid value (see section 2.2.3.6.2).

The values of individual elements MUST be set as specified in section 2.2.3.6. If parameter validation fails, the server MUST fail the operation immediately with an implementation-defined failure value.

If parameter validation succeeded, the server confirms the following:

- Connection identified by COMM\_CXTION exists on the local replica member.
- Replica set identified by COMM\_REPLICA exists on the local replica member.

If confirmation fails, the server MUST fail the operation immediately with an implementation-defined failure value. The client MUST tear down the established connection session and re-establish the connection session (see section 3.3.4.6). <78>

If the preceding test passed, the server can call the client back through FrsRpcSendCommPkt with a new COMM\_PACKET object to reply to the client (see sections 3.3.4.4.2 through 3.3.4.4.12).

In the following discussion, these abbreviations (and their meanings) are used:

**P\_IN:** Packet received in the original FrsRpcSendCommPkt call.

**P\_OUT:** On receiving a valid FrsRpcSendCommPkt call, if a reply is needed, P\_OUT represents the packet to be sent out, and the local replica member MUST reply by calling FrsRpcSendCommPkt to the original FrsRpcSendCommPkt caller.

**CO\_IN:** P\_IN.COMM\_REMOTE\_CO (The change order that is embedded in P\_IN.)

**CO\_OUT:** P\_OUT.COMM\_REMOTE\_CO (The change order that is embedded in P\_OUT.)

On receiving a COMM\_PACKET, FRS MUST determine the type of packet received. If a reply is needed, FRS MUST reply with a different type of COMM\_PACKET. Sections 3.3.4.4.2 through 3.3.4.4.12 specify how a response packet MUST be constructed based on the input COMM\_PACKET type.

If a reply is needed, P\_OUT MUST be in the following format:

**COMM\_BOP:** 0.

**COMM\_TO:**

- GUID: MUST be P\_IN.COMM\_FROM.GUID.
- Name: SHOULD<79> be FQDN of the downstream partner.

**COMM\_FROM:**

- GUID: MUST be P\_IN.COMM\_TO.GUID.
- Name: SHOULD<80> be the FQDN of the local member.

**COMM\_REPLICA:**

- GUID: MUST be P\_IN.COMM\_FROM.GUID.
- Name: MUST be P\_IN.COMM\_REPLICA.Name.

**COMM\_CXTION:** MUST be P\_IN.COMM\_CXTION.

Other elements are specified in sections 3.3.4.4.2 through 3.3.4.4.12.

**COMM\_EOP:** 0xffffffff.

#### **3.3.4.4.2 COMM\_COMMAND Is CMD\_NEED\_JOIN**

The downstream partner sent CMD\_NEED\_JOIN to inform the upstream partner that a Join operation is needed (section 3.3.4.6). The upstream partner MUST respond with a CMD\_START\_JOIN packet.

COMM\_COMMAND MUST be CMD\_START\_JOIN (0x122).

COMM\_JOIN\_GUID MUST be all 0 for new connection.

COMM\_LAST\_JOIN\_TIME MUST be 1.

Other elements are specified in section 3.3.4.4.1.

See sections 4.1 and 4.4.1 for examples of the use of this command.

#### **3.3.4.4.3 COMM\_COMMAND Is CMD\_START\_JOIN**

The upstream Partner sent CMD\_START\_JOIN to indicate that it is alive and a connection session MUST be established. The downstream partner MUST respond with a CMD\_JOINING packet to initiate the connection session establishment process.

COMM\_COMMAND MUST be CMD\_JOINING (0x130).

COMM\_JOIN\_GUID MUST be a new GUID that is generated locally.

COMM\_LAST\_JOIN\_TIME MUST be 1 for a new connection or MUST be the LastJoinTime value for an existing session.

COMM\_VVECTOR is the local member version vector. If the local member knows m originators in its version vector, there MUST be m COMM\_VVECTOR elements in the packet, one for each originator.

COMM\_JOIN\_TIME MUST be the current time.

COMM\_REPLICA\_VERSION\_GUID MUST be the unique GUID that is generated for the local member when the replica set is first initialized on the local member machine.

COMM\_COMPRESSION\_GUID MUST be the unique GUID (see section 2.2.3.6.12).

Other elements are specified in section 3.3.4.4.1.

See sections 4.1 and 4.4.2 for examples of the use of this command.

#### **3.3.4.4.4 COMM\_COMMAND Is CMD\_JOINING**

A downstream partner sent CMD\_JOINING, trying to establish a connection session (3.3.4.6). The server MUST be the upstream partner. The server MUST have a corresponding outbound connection to the downstream partner. This connection can be a volatile connection.

The upstream partner MUST perform additional verification for the CMD\_JOINING packet.

P\_IN.COMM\_JOIN\_GUID MUST exist and be unique, or the server MUST NOT reply to the client (section 3.3.4.6). P\_IN.REPLICA\_VERSION\_GUID MUST exist and be unique, or the server MUST NOT reply to the client.

If the local member is still in the initial sync state, FRS MUST NOT reply to the client. FRS MUST NOT Join a downstream partner while in the initial sync state.

If the connection is not an outbound connection, FRS MUST NOT reply to the client.

A connection session cannot be established if the partner times are out of sync by an implementation-specific Maximum Partner Clock Skew value (unless this is a volatile connection for SYSVOL seeding).

FRS MUST compare the time when the packet was built by the partner (P\_IN.COMM\_JOIN\_TIME) to the time when the packet was received (the current time). If the time difference is larger than the Maximum Partner Clock Skew value, the connection session establishment process MUST fail by not replying to the client.<81>

If all tests are passed, FRS MUST set P\_IN.COMM\_JOIN\_GUID to be its outbound-connection connection session GUID (see section 3.1.1.5). It MUST also set P\_IN.COMM\_VVECTOR to be its outbound-connection version vector.

It MUST set the outbound connection state to be JOINED. If P\_IN.COMM\_LAST\_JOIN\_TIME is 1, or the last connection session establishment time of the outbound connection is 1, FRS MUST perform a connection VVJoin.

The upstream partner MUST answer the connection session establishment request with a CMD\_JOINED packet:

- COMM\_COMMAND MUST be CMD\_JOINED (0x128).
- COMM\_JOIN\_GUID MUST be P\_IN.COMM\_JOIN\_GUID.
- COMM\_LAST\_JOIN\_TIME MUST be the current time.
- Other elements are specified in section 3.3.4.4.1.

After the CMD\_JOINED packet is sent:

- If a connection VVJoin needs to be performed, FRS MUST go through the process specified in the following section.
- If a connection VVJoin does not need to be performed, the upstream partner MUST send unknown change orders to the downstream partner based on the downstream partner version vector.

#### **3.3.4.4.4.1 Connection VVJoin**

FRS MUST perform Connection VVJoin when either of the following conditions is true:

- P\_IN.COMM\_LAST\_JOIN\_TIME is 1.
- Last connection session establishment (Join) time of the outbound connection is 1.

During VVJoin, the upstream partner MUST send out a CMD\_REMOTE\_CO packet for each file or folder that the downstream partner does not know yet. A parent folder change order MUST be sent before its child's change order. Once all change orders that the downstream partner does not know yet are sent to the downstream partner and are processed, the upstream partner MUST send out CMD\_VVJOIN\_DONE to the downstream partner to indicate that the Connection VVJoin is done.

##### **3.3.4.4.4.1.1 Common Details for Initial Syncing a File and a Folder**

When initial syncing either a file or folder, the upstream partner MUST call the downstream partner through FrsRpcSendCommPkt to send a CMD\_REMOTE\_CO packet containing a change order. Section COMM\_COMMAND Is CMD\_JOINED specifies what the downstream partner MUST do when receiving a CMD\_REMOTE\_CO packet.

The change order MUST be constructed as follows.

**SequenceNumber:** MUST be a number larger than any previous SequenceNumber in the outbound log.

**State:** MUST be 0x14. (Request outbound propagation.)

**Flags:** Specified in sections Initial Syncing a File and Initial Syncing a Folder.

**IFlags:** Specified in sections Initial Syncing a File and Initial Syncing a Folder.

**ContentCmd:** Specified in sections Initial Syncing a File and Initial Syncing a Folder.

**LocationCmd:** Specified in sections Initial Syncing a File and Initial Syncing a Folder.

**FileAttributes:** File attributes, as specified in [MS-FSCC].

**FileVersionNumber:** The number of times that this file is closed after being modified. It is used in conflict resolution (see section COMM\_COMMAND Is CMD\_JOINED).

**PartnerAckSeqNumber:** MUST be the same as SequenceNumber.

**FileSize:** File size. For folders, it MUST be 0.<82>

**FileOffset:** MUST be 0.

**FrsVsn:** MUST be a volume sequence number for the file.

**FileUsn:** A 64-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member.

**JrnlUsn:** A 64-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member.

**JrnlFirstUsn:** A 64-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member.

**OriginalReplicaNum:** A 32-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member.

**NewReplicaNum:** A 32-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member.

**ChangeOrderGuid:** MUST be the GUID of the change order.

**OriginatorGuid:** MUST be the originator GUID.

**FileGuid:** MUST be the file GUID in the IDTable.

**OldParentGuid:** MUST be the parent GUID.

**NewParentGuid:** MUST be the parent GUID.

**CxtionGuid:** MUST be the connection GUID.

**AckVersion:** MUST be the downstream partner AckVersion. The vendor can set it to any value based on the vendor implementation, or set it to 0 and ignore it.

**Extension:** MUST be ignored.

**EventTime:** MUST be the time stamp when the change happened.<83>

**FileNameLength:** MUST be the byte number of the file name.

**FileName:** MUST be the file name.

**Spare2Ull:** MUST be 0.

**Spare1GUID:** MUST be 0.

**Spare2GUID:** MUST be 0.

**Spare1Wcs:** MUST be 0.

**Spare2Wcs:** MUST be 0.

**Spare2Bin:** MUST be 0.

#### **3.3.4.4.1.2 Initial Syncing a File**

If the change order that the downstream partner has not known yet is a file, the change order MUST be constructed as follows.

**Flags:** MUST be a Bitwise OR of the following flags:

- CO\_FLAG\_VVJOIN\_TO\_ORIG
- CO\_FLAG\_LOCALCO
- CO\_FLAG\_LOCATION\_CMD

**IFlags:** MUST be 0.

**ContentCmd:** MUST be 0x100 (REASON\_FILE\_CREATE).

**LocationCmd:** MUST be 0.

Other attributes are specified in section 3.3.4.4.1.1.

#### **3.3.4.4.1.3 Initial Syncing a Folder**

If the change order that the downstream partner has not known yet is a folder, the change order MUST be constructed as follows.

**Flags:** MUST be a Bitwise OR of the following flags:

- CO\_FLAG\_VVJOIN\_TO\_ORIG
- CO\_FLAG\_LOCALCO
- CO\_FLAG\_LOCATION\_CMD

**IFlags:** MUST be 0.

**ContentCmd:** MUST be 0x100 (REASON\_FILE\_CREATE). Other reasons can be used instead of REASON\_FILE\_CREATE, but REASON\_FILE\_CREATE is the most obvious.

**LocationCmd:** MUST be 1.

Other attributes are specified in section 3.3.4.4.1.1.

#### **3.3.4.4.1.4 CMD\_VVJOIN\_DONE Once Initial Sync Is Done**

Once all the change orders that the downstream partner does not know are sent to the downstream partner and are processed, the upstream partner MUST send out a CMD\_VVJOIN\_DONE communication packet through FrsRpcSendCommPkt.

#### **3.3.4.4.5 COMM\_COMMAND Is CMD\_JOINED**

The server MUST be the downstream partner to receive this call. On receiving this packet, the upstream partner MUST send out any change orders that are unknown to the downstream partner through the CMD\_REMOTE\_CO packet. The server (or downstream partner) MUST NOT reply to this

CMD\_JOINED packet. The server MUST wait for further packets from the upstream partner for either initial sync or normal sync.

See sections 4.1 and 4.4.4 for examples of the use of this command.

### **3.3.4.4.6 COMM\_COMMAND Is CMD\_REMOTE\_CO**

The server MUST be the downstream partner to receive this call. The upstream partner has sent a remote change order to the downstream partner.

The downstream partner MUST determine if the remote change order is for a folder or file.

If CO\_FLAG\_LOCATION\_CMD is set inside CO\_IN.Flags:

- If CO\_IN.LocationCmd.DirOrFile is 1, the remote change order is for a folder.
- If CO\_IN.LocationCmd.DirOrFile is 0, the remote change order is for a file.

If CO\_FLAG\_LOCATION\_CMD is NOT set inside CO\_IN.Flags:

- If FILE\_ATTRIBUTE\_DIRECTORY(0x10) flag is set in CO\_IN.FileAttributes, the remote change order is for a folder.
- If FILE\_ATTRIBUTE\_DIRECTORY(0x10) flag is NOT set in CO\_IN.FileAttributes, the remote change order is for a file.

If CO\_LOCATION\_DELETE or CO\_LOCATION\_MOVEOUT flag is set in CO\_IN.LocationCmd, the change order specifies a deletion for either an endpoint file or folder.

When the change order is not for a deletion, if CO\_FLAG\_CONTENT\_CMD is set in CO\_IN.Flags, CO\_IN.ContentCmd specifies the nature of the change:

- REASON\_RENAME\_NEW\_NAME specifies a renaming.
- All other reasons are treated as a copy command.

If the change order is not a deletion or a renaming, the change order is treated as a copying.

If CO\_FLAG\_OUT\_OF\_ORDER flag is set in the change order, the change order MUST NOT be dampened (a dampened change order does not need to be processed because it does not contain new information about a file or folder).

If CO\_FLAG\_OUT\_OF\_ORDER flag is not set in the change order, find the corresponding VSN for CO\_IN.OriginatorGuid inside the local replica member version vector. If the VSN is larger or equal to CO\_IN.FrsVsn, the change order MUST be dampened.

If a change order is dampened, the downstream partner MUST send out CMD\_REMOTE\_CO\_DONE (see section 3.3.4.4.6.2).

If the change order is not dampened, the downstream partner MUST verify if there are any naming (multiple replica members create files or folders with the same name under the replica tree) or update conflicts (multiple replica members update the same files or folders). If there are conflicts, the downstream partner MUST resolve the conflicts to make data consistent among the replica members.

If identically named files are created or modified on two or more replica members, the downstream partner MUST use a "last writer wins" rule; this means that the most recently created or modified version of a file becomes the version that is replicated to the other replica members. All change orders have event times in Coordinated Universal Time (UTC).

These event times are used for comparison to the IDTable:



- Event Time delta > 30 minutes: Last writer wins.
- Event Time delta <= 30 minutes: Highest version wins (that is, most prolific writer wins).
- If both the event times and versions match, the file with a larger size wins.
- If the file size also matches, the change order with a larger Originator GUID wins.
- If the Originator GUID also matches, the local file loses.

If identically named folders are created on two or more replica members, the downstream partner identifies the conflict during replication and renames the folder that was most recently created (using a morphed name). Both folders MUST be replicated to all servers in the replica set. The morphed names MUST have a suffix of the form `_NTFRS_XXXXXXXX` added to the end of the folder name, where "XXXXXXXX" represents eight random hexadecimal digits. Administrators MAY be notified about morphed names.

The downstream partner MAY <84> suppress identical files and folders that it receives more than once.

When a conflict is detected, if the change order inside `P_IN` loses, the change order is abandoned and the server MUST send out `CMD_REMOTE_CO_DONE` packet (see section 3.3.4.4.6.2).

If the change order is the winner or there is no conflict, the downstream partner MUST put this change order in the inbound connection inbound log. The downstream partner MUST check the `CO_FLAG_VVJOIN_TO_ORIG` flag inside `P_IN.COMM_REMOTE_CO.Flags`:

- If `CO_FLAG_VVJOIN_TO_ORIG` flag is set, the downstream partner MUST perform initial sync.
- Otherwise, the downstream partner MUST perform normal sync.

The downstream partner MUST determine if a staging file is needed for `CO_IN`.

If `CO_FLAG_LOCATION_CMD` is set inside `CO_IN.Flags`:

- If `CO_IN.LocationCmd` is one of the following, a staging file MUST be needed:
  - `CO_LOCATION_CREATE`
  - `CO_LOCATION_MOVEIN`
  - `CO_LOCATION_MOVEIN2`
- If `CO_IN.LocationCmd` is one of the following, no staging file is needed:
  - `CO_LOCATION_DELETE`
  - `CO_LOCATION_MOVEOUT`
- If `CO_IN.LocationCmd` is either `CO_LOCATION_MOVERS` or `CO_LOCATION_MOVEDIR`, the downstream partner MUST check `CO_IN.ContentCmd` to determine if a staging file is needed.

If the downstream partner cannot determine if a staging file is needed from `CO_IN.LocationCmd`, the downstream partner MUST check if the `CO_FLAG_CONTENT_CMD` flag is set in `CO_IN.Flags`:

- If `CO_FLAG_CONTENT_CMD` is not set in `CO_IN.Flags`, no staging file is needed.
- If `CO_FLAG_CONTENT_CMD` is set in `CO_IN.Flags`, the downstream partner MUST determine if one or more values are set in `CO_IN.ContentCmd`:
  - If one or more values are set in `CO_IN.ContentCmd`, a staging file is needed.

- If no value is set in CO\_IN.ContentCmd, a staging file is not needed.

If a staging file is not needed, the downstream partner MUST NOT request a staging file from the upstream partner. The downstream partner MUST send out the CMD\_REMOTE\_CO\_DONE packet (see section 3.3.4.4.6.2).

If a staging file is needed, the downstream partner MUST request a staging file from the upstream partner through CMD\_SEND\_STAGE packets (see section 3.3.4.4.6.1). For each CMD\_SEND\_STAGE packet, the upstream partner MUST send a fixed size of staging file data to the downstream partner through the CMD\_RECEIVING packet. The last CMD\_RECEIVING packet might have less than the fixed size of staging file data. (The downstream partner does not try to pad the last packet to make all CMD\_RECEIVING packets the same size.)<85>

After all parts of the staging file are downloaded (or no staging file is required), the downstream partner MUST send the upstream partner a CMD\_REMOTE\_CO\_DONE packet to acknowledge that the change order has been successfully processed on the downstream partner (see section 3.3.4.4.6.2). The downstream partner MUST update the local replica member version vector accordingly, except if the CO\_FLAG\_SKIP\_VV\_UPDATE flag is set in CO\_IN; then this downstream partner MUST forward this change order to all other downstream partners by CMD\_REMOTE\_CO packet.

Both CMD\_REMOTE\_CO\_DONE and CMD\_SEND\_STAGE MUST include a change order in their packets. Following are common details for those change orders (these details are not listed in sections 3.3.4.4.6.1 and 3.3.4.4.6.2).

**SequenceNumber:** MUST be a number that is larger than any change order sequence number in the current member outbound log. The sequence number MUST start at one and MUST be incremented with each change order that goes into the outbound log. The sequence number MUST be unique per replica set per machine.

**FileAttribute:** MUST be CO\_IN.FileAttribute.

**FileVersionNumber:** MUST be CO\_IN.FileVersionNumber.

**PartnerAckSeqNumber:** MUST be CO\_IN.PartnerAckSeqNumber.

**FileSize:** MUST be CO\_IN.FileSize.

**FrsVsn:** MUST be CO\_IN.FrsVsn.

**FileUSN:** A 64-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member. This value is only meaningful on the current replica member and has no meaning on any other replica member. Once the change order is sent to another replica member, the receiving partner MUST ignore this field.

**JrnIUSN:** A 64-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member. This value is only meaningful on the current replica member and has no meaning on any other replica member. Once the change order is sent to another replica member, the receiving partner MUST ignore this field.

**JrnIFirstUSN:** A 64-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member. This value is only meaningful on the current replica member and has no meaning on any other replica member. Once the change order is sent to another replica member, the receiving partner MUST ignore this field.

**OriginalReplicaNumber:** A 32-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member. This value is only meaningful on the current replica member and has no meaning on any other replica member. Once the change order is sent to another replica member, the receiving partner MUST ignore this field.

**NewReplicaNumber:** A 32-bit, unsigned integer that MUST indicate the internal implementation-specific data on the current replica member. This value is only meaningful on the current replica member and has no meaning on any other replica member. Once the change order is sent to another replica member, the receiving partner MUST ignore this field.

**ChangeOrderGuid:** MUST be CO\_IN.ChangeOrderGuid.

**OriginatorGuid:** MUST be CO\_IN.OriginatorGuid.

**FileGuid:** MUST be CO\_IN.FileGuid.

**OldParentGuid:** MUST be the local replica tree root GUID if the parent is the replica tree root; otherwise, set it to CO\_IN.OldParentGuid.

**NewParentGuid:** MUST be the local replica tree root GUID if the parent is the replica tree root; otherwise, set it to CO\_IN.NewParentGuid.

**CxtionGuid:** MUST be CO\_IN.CxtionGuid.

**AckVersion:** MUST be CO\_IN.AckVersion.

**Flags:** Specified in sections 3.3.4.4.6.1 and 3.3.4.4.6.2.

**IFlags:** Specified in sections 3.3.4.4.6.1 and 3.3.4.4.6.2.

**State:** Specified in sections 3.3.4.4.6.1 and 3.3.4.4.6.2.

**ContentCmd:** Specified in sections 3.3.4.4.6.1 and 3.3.4.4.6.2.

**FileOffset:** Specified in sections 3.3.4.4.6.1 and 3.3.4.4.6.2.

### 3.3.4.4.6.1 Requesting a Staging File

On receiving CMD\_REMOTE\_CO, if the downstream partner determines that a staging file is needed (see section 3.3.4.4.6), the downstream partner MUST request the staging file by CMD\_SEND\_STAGE packet through FrsRpcSendCommPkt.

The server MUST construct the CMD\_SEND\_STAGE packet as follows.

COMM\_COMMAND: MUST be 0x228 (CMD\_SEND\_STAGE).

COMM\_JOIN\_GUID: MUST be P\_IN.COMM\_JOIN\_GUID.

COMM\_LAST\_JOIN\_TIME: MUST be P\_IN.COMM\_LAST\_JOIN\_TIME.

COMM\_BLOCK\_SIZE: MUST be 0.

COMM\_FILE\_SIZE

- If this is the first CMD\_SEND\_STAGE request to request the beginning of the staging file, then COMM\_FILE\_SIZE MUST be 0.
- Otherwise, COMM\_FILE\_SIZE MUST be the file size, which is returned from the first CMD\_RECEIVING\_STAGE.

COMM\_FILE\_OFFSET: MUST be set to  $n \times 0x100$ , where  $n$  is the sequence number of the request, starting from 0 (0, 0x100, 0x200 ...until  $n \times 0x100$ ).

COMM\_CO\_GUID: MUST be CO\_IN.ChangeOrderGuid.

COMM\_CO\_SEQUENCE\_NUMBER: MUST be 0.

COMM\_CO\_EXTENSION\_2: If the CMD\_REMOTE\_CO packet is for an existing file, the **Data** field in DATA\_EXTENSION\_CHECKSUM SHOULD be set as specified in section 2.2.3.7. Otherwise, the **Data** field SHOULD be set to NULL.

#### **CO\_OUT.Flags:**

When copying files.

- Initial Sync:

Compressed staging file: MUST be a bitwise OR of the following flags:

- CO\_FLAG\_VVJOIN\_TO\_ORIG
- CO\_FLAG\_COMPRESSED\_STAGE
- CO\_FLAG\_NEW\_FILE
- CO\_FLAG\_LOCATION\_CMD
- CO\_FLAG\_CONTENT\_CMD

Uncompressed staging file: MUST be a bitwise OR of the following flags:

- CO\_FLAG\_VVJOIN\_TO\_ORIG
- CO\_FLAG\_NEW\_FILE
- CO\_FLAG\_LOCATION\_CMD
- CO\_FLAG\_CONTENT\_CMD

- Normal Sync:

Compressed staging file: MUST be a bitwise OR of the following flags:

- CO\_FLAG\_COMPRESSED\_STAGE
- CO\_FLAG\_NEW\_FILE (if a new file)
- CO\_FLAG\_LOCATION\_CMD
- CO\_FLAG\_CONTENT\_CMD

Uncompressed staging file: MUST be a bitwise OR of the following flags:

- CO\_FLAG\_NEW\_FILE (if a new file)
- CO\_FLAG\_LOCATION\_CMD
- CO\_FLAG\_CONTENT\_CMD

When renaming existing file.

- Normal Sync:

Compressed staging file: MUST be a bitwise OR of the following flags:

- CO\_FLAG\_COMPRESSED\_STAGE
- CO\_FLAG\_CONTENT\_CMD

Uncompressed staging file: MUST be a bitwise OR of the following flags:

- CO\_FLAG\_CONTENT\_CMD

When copying folder.

- Initial Sync: MUST be a bitwise OR of the following flags:
  - CO\_FLAG\_VVJOIN\_TO\_ORIG
  - CO\_FLAG\_NEW\_FILE
  - CO\_FLAG\_LOCATION\_CMD
- Normal Sync: MUST be a bitwise OR of the following flags:

Compressed staging file:

- CO\_FLAG\_COMPRESSED\_STAGE
- CO\_FLAG\_NEW\_FILE (if a new folder)
- CO\_FLAG\_LOCATION\_CMD

Uncompressed staging file: MUST be a bitwise OR of the following flags:

- CO\_FLAG\_NEW\_FILE (if a new folder)
- CO\_FLAG\_LOCATION\_CMD

When renaming existing folder in the same replica set.

- Normal Sync:
  - Compressed staging file: MUST be a bitwise OR of the following flags:
    - CO\_FLAG\_COMPRESSED\_STAGE
    - CO\_FLAG\_CONTENT\_CMD
  - Uncompressed staging file:
    - CO\_FLAG\_CONTENT\_CMD

**CO\_OUT.IFlags:** MUST be 0.

**CO\_OUT.State:**

- Initial Sync: MUST be 5 (allocating staging file space for remote change order).
- Normal Sync: MUST be 6 (remote change order staging file fetch has started).

**CO\_OUT.ContentCmd:**

- Copying file: MUST be 0x2 (REASON\_DATA\_EXTEND).
- Renaming existing file or folder: MUST be 0x2000 (REASON\_RENAME\_NEW\_FILE).
- Copying folder: MUST be 0.

**LocationCmd:** MUST be CO\_IN.LocationCmd.

**CO\_OUT.FileOffset:** MUST be 0.

### 3.3.4.4.6.2 Acknowledging the Change Is Done

Once the change order is processed by the local replica member, the server MUST send CMD\_REMOTE\_CO\_DONE to the upstream partner to indicate that the change order is processed.

The server MUST construct the CMD\_REMOTE\_CO\_DONE packet as follows.

COMM\_COMMAND: MUST be 0x250 (CMD\_REMOTE\_CO\_DONE).

COMM\_JOIN\_GUID: MUST be P\_IN.COMM\_JOIN\_GUID.

COMM\_LAST\_JOIN\_TIME: MUST be P\_IN.COMM\_LAST\_JOIN\_TIME.

COMM\_BLOCK\_SIZE: MUST be 0.

COMM\_FILE\_SIZE: MUST be 0.

COMM\_FILE\_OFFSET: MUST be 0.

COMM\_GVSN: MUST be the current member GVSN.

COMM\_CO\_GUID: MUST be CO\_IN.ChangeOrderGuid.

COMM\_CO\_SEQUENCE\_NUMBER: MUST be CO\_IN.PartnerAckSeqNumber.

#### **CO\_OUT.Flags:**

When the change order is dampened (see section 3.3.4.4.5).

- Compressed staging file: MUST be a bitwise OR of the following flags:
  - CO\_FLAG\_SKIP\_VV\_UPDATE
  - CO\_FLAG\_COMPRESSED\_STAGE
  - CO\_FLAG\_DIRECTED\_CO
  - CO\_FLAG\_RETRY
  - CO\_FLAG\_CONTENT\_CMD
  - CO\_FLAG\_VV\_ACTIVATED
  - CO\_FLAG\_ABORT\_CO
- Uncompressed staging file: MUST be a bitwise OR of the following flags:
  - CO\_FLAG\_SKIP\_VV\_UPDATE
  - CO\_FLAG\_DIRECTED\_CO
  - CO\_FLAG\_RETRY
  - CO\_FLAG\_CONTENT\_CMD
  - CO\_FLAG\_VV\_ACTIVATED
  - CO\_FLAG\_ABORT\_CO

When the change order is not dampened but is the conflict loser.

- Compressed staging file: MUST be a bitwise OR of the following flags:
  - CO\_FLAG\_COMPRESSED\_STAGE

- CO\_FLAG\_CONTENT\_CMD
- CO\_FLAG\_VV\_ACTIVATED
- CO\_FLAG\_ABORT\_CO
- Uncompressed staging file: MUST be a bitwise OR of the following flags:
  - CO\_FLAG\_CONTENT\_CMD
  - CO\_FLAG\_VV\_ACTIVATED
  - CO\_FLAG\_ABORT\_CO

When the change order is neither dampened nor the conflict loser.

- Copying file.

Initial Sync:

- Compressed staging file: MUST be a bitwise OR of the following flags:
  - CO\_FLAG\_VVJOIN\_TO\_ORIG
  - CO\_FLAG\_COMPRESSED\_STAGE
  - CO\_FLAG\_NEW\_FILE
  - CO\_FLAG\_LOCATION\_CMD
  - CO\_FLAG\_CONTENT\_CMD
  - CO\_FLAG\_ACTIVATED
- Uncompressed staging file: MUST be a bitwise OR of the following flags:
  - CO\_FLAG\_VVJOIN\_TO\_ORIG
  - CO\_FLAG\_NEW\_FILE
  - CO\_FLAG\_LOCATION\_CMD
  - CO\_FLAG\_CONTENT\_CMD
  - CO\_FLAG\_ACTIVATED

Normal Sync:

- Compressed staging file: MUST be a bitwise OR of the following flags:
  - CO\_FLAG\_COMPRESSED\_STAGE
  - CO\_FLAG\_NEW\_
  - FILE (if a new file)
  - CO\_FLAG\_LOCATION\_CMD
  - CO\_FLAG\_CONTENT\_CMD
  - CO\_FLAG\_ACTIVATED
- Uncompressed staging file: MUST be a bitwise OR of the following flags:

- CO\_FLAG\_NEW\_FILE (if a new file)
- CO\_FLAG\_LOCATION\_CMD
- CO\_FLAG\_CONTENT\_CMD
- CO\_FLAG\_ACTIVATED

When renaming a file or folder.

- Normal Sync:

Compressed staging file: MUST be a bitwise OR of the following flags:

- CO\_FLAG\_COMPRESSED\_STAGE
- CO\_FLAG\_CONTENT\_CMD
- CO\_FLAG\_ACTIVATED

Uncompressed staging file. MUST be a bitwise OR of the following flags:

- CO\_FLAG\_LOCATION\_CMD
- CO\_FLAG\_ACTIVATED

When copying a folder.

- Initial Sync:

Compressed staging file. MUST be a bitwise OR of the following flags:

- CO\_FLAG\_VVJOIN\_TO\_ORIG
- CO\_FLAG\_COMPRESSED\_STAGE
- CO\_FLAG\_NEW\_FILE
- CO\_FLAG\_LOCATION\_CMD
- CO\_FLAG\_ACTIVATED

Uncompressed staging file. MUST be a bitwise OR of the following flags:

- CO\_FLAG\_VVJOIN\_TO\_ORIG
- CO\_FLAG\_NEW\_FILE
- CO\_FLAG\_LOCATION\_CMD
- CO\_FLAG\_ACTIVATED

- Normal Sync:

Compressed staging file: MUST be a bitwise OR of the following flags:

- CO\_FLAG\_COMPRESSED\_STAGE
- CO\_FLAG\_NEW\_FILE (if a new folder)
- CO\_FLAG\_LOCATION\_CMD
- CO\_FLAG\_ACTIVATED



Uncompressed staging file: MUST be a bitwise OR of the following flags:

- CO\_FLAG\_NEW\_FILE (if a new folder)
- CO\_FLAG\_LOCATION\_CMD
- CO\_FLAG\_ACTIVATED

When deleting a file or folder: MUST be 0xA. bitwise OR:

- CO\_FLAG\_LOCATION\_CMD
- CO\_FLAG\_VV\_ACTIVATED

**CO\_OUT.IFlags:**

- If change order is dampened: MUST be 0.
- If change order is not dampened: MUST be 0x1 (CO\_IFLAG\_VVRETIRE\_EXEC).

**CO\_OUT.State:**

- If change order is dampened: MUST be 8. (Waiting to retry the fetch of the staging file for the remote change order.)
- If change order is not dampened but is the conflict loser: MUST be 0. (Change order is entered in the log.)
- If change order is neither dampened nor the conflict loser: MUST be 0x16. (Database state update started).

**CO\_OUT.ContentCmd:**

- Initial Sync: MUST be 0.
- Normal Sync: MUST be CO\_IN.ContentCmd.

**LocationCmd:**

- If change order is either dampened or the conflict loser:
  - If a file: MUST be 0xE.
  - If a folder: MUST be 0xF.
- Otherwise: MUST be CO\_IN.LocationCmd.

**CO\_OUT.FileOffset:** MUST be 0.

See sections 4.2, 4.4.5, 4.5.1.1, 4.5.2.1, 4.5.3.1, 4.5.4.1, 4.5.5.1, and 4.5.6.1.

### **3.3.4.4.7 COMM\_COMMAND Is CMD\_SEND\_STAGE**

The downstream partner is requesting the local staging file. The upstream partner MUST send the local staging file to the requesting downstream partner.

The server MUST validate the following:

- P\_IN.COMM\_CXTION MUST exist.
- CO\_IN.ChangeOrderGuid MUST be a GUID.

- P\_IN.COMM\_CXTION MUST be an outbound connection that has joined with the downstream partner.

If one of the above tests fails, the server MUST NOT reply to this call. If all of the above tests pass, the server MUST send CMD\_RECEIVING\_STAGE packet to the downstream partner:

**COMM\_COMMAND:** MUST be 0x238 (CMD\_RECEIVING\_STAGE).

**COMM\_JOIN\_GUID:** MUST be P\_IN.COMM\_JOIN\_GUID.

**COMM\_LAST\_JOIN\_TIME:** MUST be P\_IN.COMM\_LAST\_JOIN\_TIME.

**COMM\_BLOCK:**

**Stage Header:** MUST be attached if COMM\_FILE\_OFFSET in this packet is 0. Set the fields in the stage header as follows:

- Major MUST be set to 0.
- Minor SHOULD be set to 3. 0, 1, and 2 SHOULD NOT be used (see section 2.2.3.10).
- DataHigh MUST be set to 0.
- DataLow MUST be set to 0x400.
- Compression MUST be set to 0.
- A structure of type FILE\_NETWORK\_OPEN\_INFORMATION that describes the file or folder. For the definition of this structure, see [MS-FSCC] section 2.4.27.
- ChangeOrderCommand MUST be copied from the CMD\_SEND\_STAGE, except for the following fields:
  - Flags MUST bitwise OR CO\_FLAG\_LOCALCO if the change order originates locally on the upstream partner.
  - State:
    - For Initial Sync: SHOULD be set to 5 (allocating staging file space for remote change order).
    - For Normal Sync: SHOULD be set to 1 (allocating staging file space for local change order).
  - PartnerAckSeqNumber MUST be set to the sequence number received from the upstream partner in CMD\_REMOTE\_CO.
  - COMM\_REMOTE\_CO.FileUsn MUST be ignored.
  - COMM\_REMOTE\_CO.OldParentGuid MUST be set to the local replica tree root GUID if the parent is the replica tree root; otherwise, keep the original value from the CMD\_REMOTE\_CO packet received.
  - COMM\_REMOTE\_CO.NewParentGuid MUST be set to the local replica tree root GUID if the parent is the replica tree root; otherwise, keep the original value from the CMD\_REMOTE\_CO packet received.
  - AckVersion MUST be set to the version vector number received from the upstream partner in CMD\_REMOTE\_CO.

- CocExt MUST be set to 0 if the downstream partner's FRS Minor version is NTFRS\_COMM\_MINOR\_4 or above. If the downstream partner has a FRS Minor version below NTFRS\_COMM\_MINOR\_4, the structure MUST hold data as specified in section 2.2.3.6.21. <86>
- The Microsoft backup format data for the staging file being transmitted, as specified in [MS-BKUP]. The data can be compressed.

**COMM\_BLOCK\_SIZE:** MUST be the size of COMM\_BLOCK.Length (see section 2.2.3.6.13).

**COMM\_FILE\_SIZE:** MUST be the total size of the staging data and the staging header.

**COMM\_FILE\_OFFSET:** MUST be the P\_IN.COMM\_FILE\_OFFSET.

**COMM\_CO\_GUID:** MUST be set to CO\_IN.ChangeOrderGuid.

**COMM\_CO\_SEQUENCE\_NUMBER:** MUST be 0.

See sections 4.2, 4.4.6, 4.5.1.2, 4.5.1.4, 4.5.2.2, 4.5.4.2, and 4.5.6.2 for examples of the use of this command.

The downstream partner will receive the CMD\_ABORT\_FETCH (see section 3.3.4.4.10) or CMD\_RETRY\_FETCH (see section 3.3.4.4.11) packet if the upstream partner is unable to handle the CMD\_SEND\_STAGE packet for any of the following reasons.

When the upstream partner receives the CMD\_SEND\_STAGE request then the out connection MUST be in JOINED state. If the connection is not in JOINED state then server MUST send CMD\_RETRY\_FETCH back to downstream partner. The downstream partner MUST perform the initial sync and then send a CMD\_SEND\_STAGE to the upstream partner.

If the upstream partner is unable to access the staging file due to ERROR\_SHARING\_VIOLATION then the upstream partner MUST send a CMD\_RETRY\_FETCH back to the downstream partner.

If the staging file is missing then the upstream partner MUST try to generate the missing staging file. If the upstream partner is unable to create a missing staging file with ERROR\_DISK\_FULL then the upstream partner MUST send CMD\_RETRY\_FETCH to the downstream partner. ERROR\_DISK\_FULL is also generated when the staging quota is reached for the replica set; the error goes away when staging quota is increased manually or staging cleanup is done by the upstream partner. If the upstream partner fails to create the staging file because of any other failure (such as an ACCESS\_DENIED error) then the upstream partner MUST send a CMD\_ABORT\_FETCH to the downstream partner.

If the upstream partner is unable to fetch the staging file because it has already been deleted then the upstream partner MUST send a CMD\_RETRY\_FETCH to the downstream partner, which will cause the file to be recreated on the next CMD\_SEND\_STAGE request from the downstream partner.

If the upstream partner is unable to read the file due to file corruption then the upstream partner MUST delete the corrupted file and send CMD\_RETRY\_FETCH to the downstream partner, which will cause the file to be recreated on the next CMD\_SEND\_STAGE request from the downstream partner.

### 3.3.4.4.8 COMM\_COMMAND Is CMD\_RECEIVING\_STAGE

The upstream partner is sending the staging file to the downstream partner.

The server MUST validate the following:

- P\_IN.COMM\_CXTION MUST exist.
- CO\_IN.ChangeOrderGuid MUST be a GUID.
- P\_IN.COMM\_CXTION MUST be an inbound connection that has joined with the upstream partner.

- Validate the RPC call MUST use either `RPC_C_AUTHN_GSS_KERBEROS` or `RPC_C_AUTH_GSS_NEGOTIATE`.

If the above test fails, the server MUST NOT reply to the client.

If the above test passes:

- If all parts of the file have been sent, the server MUST extract the staging file header from the first blocks, and then link all the data together to form an array. Decompress the staging file if it is compressed. Then, perform a backup write (as specified in [MS-BKUP]) to the local file system to restore the file or folder. If the server is unable to link all the data together, to decompress the staging file, or to do a backup write to the local file system, it MUST abort the request. Otherwise, the server MUST send a `CMD_REMOTE_CO_DONE` packet (see section 3.3.4.4.6.2).
- If some part of the file has not been sent yet, the local member MUST acknowledge the call by a `CMD_SEND_STAGE` packet (see section 3.3.4.4.6.1) to request the next sequential block of staging files from the upstream partner. (The upstream partner chops the staging file into blocks. These blocks are sent to the downstream partner one by one, starting from the first block that contains the staging header and the first part of the staging file data. A later block MUST NOT be processed unless the previous block has been downloaded successfully.)

See sections 4.2, 4.4.7, 4.5.1.3, 4.5.1.5, 4.5.2.3, 4.5.4.3, and 4.5.6.3 for examples of the use of this command.

#### **3.3.4.4.9 COMM\_COMMAND Is CMD\_REMOTE\_CO\_DONE**

The receiving partner MUST be the upstream partner. This indicates the downstream partner has processed this change order. FRS MUST remove the change order from the Outbound Connection Outbound Log (OutLog) only if the `CO_FLAG_INSTALL_INCOMPLETE` flag is not set. FRS MUST NOT reply to this `CMD_REMOTE_CO_DONE` packet.

See sections 4.2, 4.4.8, 4.5.1.6, 4.5.2.4, 4.5.3.2, 4.5.4.4, 4.5.5.2, and 4.5.6.4 for examples of the use of this command.

#### **3.3.4.4.10 COMM\_COMMAND Is CMD\_ABORT\_FETCH**

The receiving partner MUST be the downstream partner. On receiving this packet, it indicates that the upstream partner cannot handle the `CMD_SEND_STAGE` packet now or ever. The server MUST discard the Remote Change Order from the inbound connection Inbound Log for the change order inside `CMD_SEND_STAGE` packet. FRS MUST NOT reply to this `CMD_ABORT_FETCH` packet.

#### **3.3.4.4.11 COMM\_COMMAND Is CMD\_RETRY\_FETCH**

The server MUST be the downstream partner. On receiving this packet, it indicates the upstream partner cannot handle the `CMD_SEND_STAGE` packet now. The server MUST resend the same `CMD_SEND_STAGE` packet after five seconds (see section 3.3.4.4.6.1).

In the new `CMD_SEND_STAGE` packet, `CO_FLAG_RETRY` MUST be set in the change order Flags field.

#### **3.3.4.4.12 COMM\_COMMAND Is CMD\_VVJOIN\_DONE**

This packet indicates that the initial sync is complete (that is, all change orders for this initial sync have been sent out to the downstream partner and are processed) between the two machines. FRS MUST NOT reply to this packet.

See section 4.4.9 for an example of the use of this command.

#### **3.3.4.4.13 COMM\_COMMAND Is CMD\_UNJOIN\_REMOTE**

On receiving this packet:

- For Distributed File System (DFS) replica set connections, the connection **MUST** be torn down. The File Replication Service (FRS) **MUST** stop replication on the connection. For more information, see the On/Off Schedule in section 3.3.2.1.2.
- For SYSVOL replica set connections, the connection **MUST** be torn down after all the change orders are installed on the downstream partner. FRS **MUST** stop replication on the connection afterward. For more information, see the Trigger Schedule in section 3.3.2.1.1.

FRS **MUST NOT** reply to this packet.

See section 4.4.10 for an example of the use of this command.

### 3.3.4.5 FrsRpcVerifyPromotionParent (Opnum 1)

The FrsRpcVerifyPromotionParent method does not perform any actions. It **MUST** return `ERROR_CALL_NOT_IMPLEMENTED`.

```
unsigned long FrsRpcVerifyPromotionParent(  
    [in] handle_t Handle,  
    [in, string, unique] PWCHAR ParentAccount,  
    [in, string, unique] PWCHAR ParentPassword,  
    [in, string, unique] PWCHAR ReplicaSetName,  
    [in, string, unique] PWCHAR ReplicaSetType,  
    [in] unsigned long PartnerAuthLevel,  
    [in] unsigned long GuidSize  
);
```

**Handle:** Binding handle obtained when the partner authenticates with the file replication service.

**ParentAccount:** **MUST** be 0. **MUST** be ignored on receipt.

**ParentPassword:** **MUST** be 0. **MUST** be ignored on receipt.

**ReplicaSetName:** **MUST** be 0. **MUST** be ignored on receipt.

**ReplicaSetType:** **MUST** be 0. **MUST** be ignored on receipt.

**PartnerAuthLevel:** **MUST** be 0. **MUST** be ignored on receipt.

**GuidSize:** **MUST** be 0. **MUST** be ignored on receipt.

| Return value/code                        | Description                                    |
|--|--|
| 0x00000078<br>ERROR_CALL_NOT_IMPLEMENTED | This function is not supported on this system. |

### 3.3.4.6 Establishing a Connection Session

Connections are established through the FrsRpcSendCommPkt calls:

- The connection session establishment process is started when the downstream partner sends out `CMD_NEED_JOIN`.

- Upon receiving CMD\_NEED\_JOIN, the upstream partner MUST reply by sending out CMD\_START\_JOIN to the downstream partner. The upstream partner can also start the process by sending out CMD\_START\_JOIN directly.
- Upon receiving CMD\_START\_JOIN, the downstream partner MUST reply by sending out CMD\_JOINING to the upstream partner. (See section 3.3.4.4.3.)
- Upon receiving CMD\_JOINING, the upstream partner MUST send out CMD\_JOINED to the downstream partner. (See section 3.3.4.4.4.)
- Upon receiving CMD\_JOINED, the downstream partner knows the connection is fully established and the upstream partner MUST start the Version Vector Join (VVJoin).

The partner MUST reply in a certain time frame or the process MUST fail.<87>

To send out a CMD\_NEED\_JOIN packet from the downstream partner:

- **COMM\_BOP:** 0.
- **COMM\_COMMAND:** 0x121 (CMD\_NEED\_JOIN).
- **COMM\_TO:**
  - GUID: MUST be the objectGuid of the upstream partner NTFRS member object.
  - Name: MUST be the DNS host name of the upstream partner.
- **COMM\_FROM:**
  - GUID: MUST be the objectGuid of the local member NTFRS member object.
  - Name: MUST be the host name of the local member.
- **COMM\_REPLICA:**
  - GUID: MUST be the objectGuid of the upstream partner NTFRS member object.
  - Name: MUST be the name of the NTFRS replica set object.
- **COMM\_CXTION:**
  - GUID: MUST be the inbound connection GUID.
  - Name: MUST be the upstream partner DNS host name (fully qualified domain name).
- **COMM\_JOIN\_GUID:** MUST be 0.
- **COMM\_LAST\_JOIN\_TIME:** MUST be 0x1.
- **COMM\_EOP:** ff ff ff ff.

To send out a CMD\_START\_JOIN packet from the upstream partner:

- **COMM\_BOP:** 0.
- **COMM\_COMMAND:** 0x122 (CMD\_START\_JOIN).
- **COMM\_TO:**
  - GUID: MUST be the objectGuid of the downstream partner inbound connection object.
  - Name: MUST be the DNS host name of the downstream partner.

- **COMM\_FROM:**
  - GUID: MUST be the objectGuid of the local member NTFRS member object.
  - Name: MUST be the host name of the local member.
- **COMM\_REPLICA:**
  - GUID: MUST be the objectGuid of the downstream partner NTFRS member object.
  - Name: MUST be the name of the NTFRS replica set object.
- **COMM\_CXTION:**
  - GUID: MUST be the outbound connection GUID.
  - Name: MUST be the downstream partner DNS host name (fully qualified domain name).
- **COMM\_JOIN\_GUID:** MUST be 0.
- **COMM\_LAST\_JOIN\_TIME:** MUST be 1.
- **COMM\_EOP:** ff ff ff ff.

### 3.3.5 Timer Events

Connection schedules define a set of timers that trigger connection session establishments and tear-downs.

- If a connection session is not established and the connection schedule indicates that File Replication Service (FRS) has to replicate (see section 2.3.1.2), a timer MUST be fired to instruct FRS to go through a connection session establishment process to setup a connection session (join).
- If a connection session is established (joined) and the connection schedule indicates that FRS has to not replicate (see section 2.3.1.2), a timer MUST be fired to instruct FRS to tear down the joined connection.

The connection session establishment process (join) (see section 3.3.4.6) is a handshake between two existing partners before they begin replicating. The process begins after any of the following events:

- Connection schedule opens.
- FRS is started while the schedule is open.
- Connection session is being re-established after loss of network connectivity or a server shutdown while the schedule is open.

### 3.3.6 Other Local Events

No other local events require special processing on the client.

## 3.4 PERFFRS Interface

The PERFFRS interface enables Perfmon (or other performance monitoring tools) to check the FRS performance counters. It defines the RPC interface between the Performance DLL and the ntfrs (server) application.

Since all of the RPC methods in this interface MUST NOT be called remotely, these RPC methods are out of the scope.

## 4 Protocol Examples

FRS uses `FrsRpcSendCommPkt` (see section 3.3.4.4) to send `COMM_PACKET` (see section 2.2.3.5) between replica members to replicate files and folders inside a replica set.

If machine A calls the `FrsRpcSendCommPkt` method on machine B, in the following sections, this is called: Machine A sends `COMM_PACKET` packet to machine B.

There are 12 types of `COMM_PACKETS` (see section 2.2.3.6.2):

- `CMD_START_JOIN`
- `CMD_JOINING`
- `CMD_JOINED`
- `CMD_REMOTE_CO`
- `CMD_SEND_STAGE`
- `CMD_RECEIVING_STAGE`
- `CMD_REMOTE_CO_DONE`
- `CMD_ABORT_FETCH`
- `CMD_RETRY_FETCH`
- `CMD_UNJOIN_REMOTE`
- `CMD_VVJOIN_DONE`

If machine A sends a `COMM_PACKET` to machine B, and if `COMM_PACKET.COMM_COMMAND` is `CMD_NEED_JOIN` (0x00000121), this is called: Machine A sends `CMD_NEED_JOIN` to machine B.

Section 4.1 gives the connection establishment sequence.

Section 4.2 gives the change order handling sequence after connections are established.

Section 4.3 gives an example to show the byte layout of `COMM_PACKET`.

Section 4.4 gives a complete initial sync example with details for every packet that is transmitted on the wire.

Section 4.5 gives a set of examples for normal sync with details for every packet that is transmitted on the wire, including:

- Copying a file.
- Renaming a file.
- Removing a file.
- Copying a folder.
- Removing a folder.
- Renaming a folder.

Sections 4.4 and 4.5 are intended to provide complete examples with every byte. These examples are used on the wire for both initial sync and normal sync.



## 4.1 Connection Establishment Sequence

This example shows how a connection is established using two machines:

- $M_A$
- $M_B$

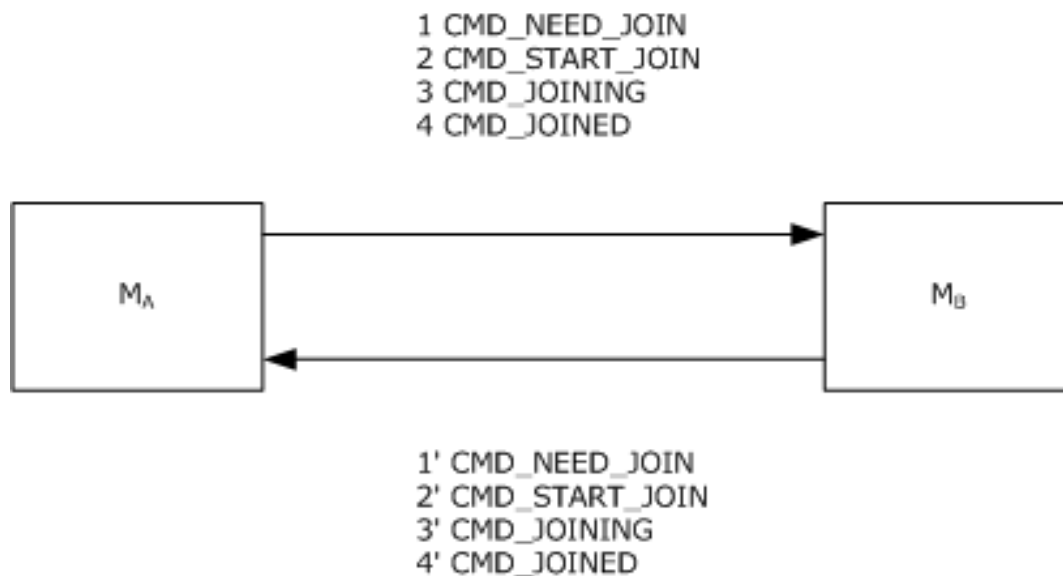
These two machines are members of a single replica set.

There are two connections linking the two machines:

- $C_{AB}$ : Connection from  $M_A$  to  $M_B$ .
- $C_{BA}$ : Connection from  $M_B$  to  $M_A$ .

The schedules on these two connections are always on.

The following diagram shows the connection establishment sequence (the number indicates the sequence numbers).



**Figure 3: Connection establishment sequence**

The connection establishment for  $C_{AB}$  and  $C_{BA}$  are independent of each other. Details on the connection establishment of  $C_{AB}$  follow (the connection establishment for  $C_{BA}$  is done in the same sequence):

- At time 1,  $M_B$  sends  $CMD\_NEED\_JOIN$  to  $M_A$ .
- At time 2, On receiving  $CMD\_NEED\_JOIN$  from  $M_B$ ,  $M_A$  sends  $CMD\_START\_JOIN$  to  $M_B$ .
- At time 3, On receiving  $CMD\_START\_JOIN$  from  $M_A$ ,  $M_B$  sends  $CMD\_JOINING$  to  $M_A$ , which includes the current version vector of  $M_B$  and a connection session GUID (see section 3.1.1.5) generated by  $M_B$  to represent the connection session that is being established.
- At time 4, on receiving  $CMD\_JOINING$ ,  $M_A$  retrieves the version vector of  $M_B$  and the connection session GUID generated by  $M_B$ , and saves both in the connection object to be used to generate change orders.

## 4.2 Change Order Handling Sequence

This example shows a change order handling sequence using three machines:

- $M_A$
- $M_B$
- $M_C$

The three machines are all members of a single replica set.

The replica tree root for each of the three machines is:

- $R_A$ : C:\ReplicaRoot
- $R_B$ : E:\Mydata\ReplicaRoot2
- $R_C$ : H:\Public\Data

There are six connections linking the three machines:

- $C_{AB}$ : Connection from  $M_A$  to  $M_B$ .
- $C_{BA}$ : Connection from  $M_B$  to  $M_A$ .
- $C_{AC}$ : Connection from  $M_A$  to  $M_C$ .
- $C_{CA}$ : Connection from  $M_C$  to  $M_A$ .
- $C_{BC}$ : Connection from  $M_B$  to  $M_C$ .
- $C_{CB}$ : Connection from  $M_C$  to  $M_B$ .

The schedules on the six connections are all always on. In this example, it is assumed that all six connections have been established.

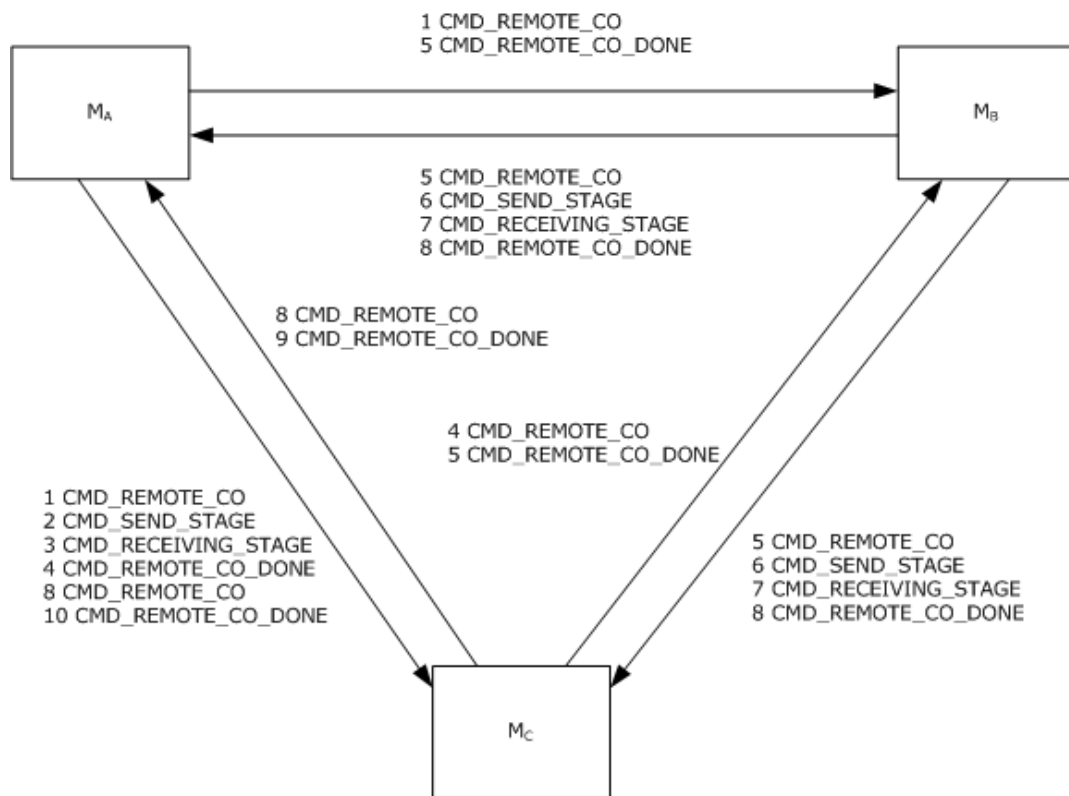
At time  $T_1$ , a file  $F_1$  is created on  $M_A$ . At time  $T_2$  ( $T_2 - T_1 = 2$  seconds), a file  $F_2$  is created on  $M_B$ . Both  $F_1$  and  $F_2$  have the same path and name relative to each of their replica tree roots:

- $F_1$ : C:\ReplicaRoot\Hello.txt
- $F_2$ : E:\MyData\ReplicaRoot2\Hello.txt

The size of  $F_1$  is eight bytes. The size of  $F_2$  is 10 bytes.

As  $F_1$  and  $F_2$  have the same name and path, each of the three machines has to resolve this as a file conflict.  $F_2$  is the winner on all three machines.

The change order handling sequence follows (the number indicates the sequence number; all the events happen after  $T_2$ ).



**Figure 4: Change order handling sequence**

At time 1:

- MA sends out the change order for F<sub>1</sub> to M<sub>B</sub> through C<sub>AB</sub>. MA sends out the change order for F<sub>1</sub> to M<sub>C</sub> through C<sub>AC</sub>.

At time 2:

When M<sub>C</sub> receives the change order for F<sub>1</sub>, M<sub>C</sub> checks its version vector and decides to accept this change order. At time 2, M<sub>C</sub> sends out CMD\_SEND\_STAGE to request the staging file for F<sub>1</sub> through C<sub>AC</sub>.

At time 3:

When M<sub>A</sub> receives the CMD\_SEND\_STAGE from M<sub>C</sub> for F<sub>1</sub> on C<sub>AC</sub>, at time 3, M<sub>A</sub> sends out CMD\_RECEIVING\_STAGE to send the staging file for F<sub>1</sub> to M<sub>C</sub> through C<sub>AC</sub>. As Hello.txt is small, one CMD\_RECEIVING\_STAGE sends out the entire staging file for F<sub>1</sub>.

At time 4:

When M<sub>C</sub> receives CMD\_RECEIVING\_STAGE for F<sub>1</sub> (because the entire staging file is downloaded), M<sub>C</sub> does two things:

- M<sub>C</sub> installs F<sub>1</sub> and sends out CMD\_REMOTE\_CO\_DONE to M<sub>A</sub> through C<sub>AC</sub> at time 4.
- M<sub>C</sub> moves the change order for F<sub>1</sub> from inbound log to outbound log. When the change order for F<sub>1</sub> is in outbound log, M<sub>C</sub> sends the change order for F<sub>1</sub> to all other machines, which is M<sub>B</sub> in this example. So, at time 4, M<sub>C</sub> sends out the change order for F<sub>1</sub> to M<sub>B</sub> through C<sub>CB</sub>.

At time 5:

- When M<sub>B</sub> receives the change order for F<sub>1</sub> from M<sub>A</sub> on C<sub>AB</sub>, M<sub>B</sub> checks its version vector and decides to accept this change order. However, M<sub>B</sub> finds a name conflict with F<sub>2</sub> on M<sub>B</sub>, and F<sub>2</sub> is the winner. M<sub>B</sub> rejects the change order for F<sub>1</sub> by sending out CMD\_REMOTE\_CO with flag CO\_FLAG\_ABORT\_CO to M<sub>A</sub> through C<sub>AB</sub> at time 5.
- When M<sub>B</sub> receives the change order for F<sub>1</sub> from M<sub>C</sub> on C<sub>CB</sub>, M<sub>B</sub> checks its version vector and decides to accept this change order. However, M<sub>B</sub> finds a name conflict with F<sub>2</sub> on M<sub>B</sub>, and F<sub>2</sub> is the winner. M<sub>B</sub> rejects the change order for F<sub>1</sub> by sending out CMD\_REMOTE\_CO with flag CO\_FLAG\_ABORT\_CO to M<sub>C</sub> through C<sub>CB</sub> at time 5.
  - M<sub>B</sub> sends out the change order for F<sub>2</sub> to M<sub>A</sub> through C<sub>BA</sub>.
  - M<sub>B</sub> sends out the change order for F<sub>2</sub> to M<sub>C</sub> through C<sub>BC</sub>.

At time 6:

- When M<sub>A</sub> receives the change order for F<sub>2</sub> from M<sub>B</sub> on C<sub>BA</sub>, M<sub>A</sub> checks its version vector and decides to accept this change order. However, M<sub>B</sub> finds a name conflict with F<sub>1</sub> on M<sub>A</sub>, and F<sub>2</sub> is the winner. M<sub>A</sub> determines it needs to install this change order. At time 6, M<sub>A</sub> sends out CMD\_SEND\_STAGE to request the staging file for F<sub>2</sub> through C<sub>BA</sub>.
- When M<sub>C</sub> receives the change order for F<sub>2</sub> from M<sub>B</sub> on C<sub>BC</sub>, M<sub>C</sub> checks its version vector and decides to accept this change order. However, M<sub>B</sub> finds a name conflict with F<sub>1</sub> from M<sub>A</sub>, and F<sub>2</sub> is the winner. M<sub>C</sub> determines it needs to install this change order. At time 6, M<sub>C</sub> sends out CMD\_SEND\_STAGE to request the staging file for F<sub>2</sub> through C<sub>BC</sub>.

At time 7:

- When M<sub>B</sub> receives the CMD\_SEND\_STAGE from M<sub>A</sub> for F<sub>2</sub> on C<sub>BA</sub>, at time 7, M<sub>B</sub> sends out CMD\_RECEIVING\_STAGE to send the staging file for F<sub>2</sub> to M<sub>A</sub> through C<sub>BA</sub>. As Hello.txt is small, one CMD\_RECEIVING\_STAGE sends out the entire staging file for F<sub>2</sub>.
- When M<sub>C</sub> receives the CMD\_SEND\_STAGE from M<sub>B</sub> for F<sub>2</sub> on C<sub>BC</sub>, at time 7, M<sub>B</sub> sends out CMD\_RECEIVING\_STAGE to send the staging file for F<sub>2</sub> to M<sub>C</sub> through C<sub>BC</sub>. As Hello.txt is small, one CMD\_RECEIVING\_STAGE sends out the entire staging file for F<sub>2</sub>.

At time 8:

- When M<sub>A</sub> receives CMD\_RECEIVING\_STAGE for F<sub>2</sub>, because the entire staging file is downloaded, M<sub>A</sub> does two things:
  - M<sub>A</sub> installs F<sub>2</sub> and sends out CMD\_REMOTE\_CO\_DONE to M<sub>B</sub> through C<sub>BA</sub> at time 8. Thus, F<sub>1</sub> is overwritten by F<sub>2</sub>.
  - M<sub>A</sub> moves the change order for F<sub>2</sub> from inbound log to outbound log. When the change order for F<sub>2</sub> is in outbound log, M<sub>A</sub> sends the change order for F<sub>2</sub> to all other machines, which is M<sub>C</sub> in this example. So, at time 8, M<sub>A</sub> sends out the change order for F<sub>2</sub> to M<sub>C</sub> through C<sub>AC</sub>.
- When M<sub>C</sub> receives CMD\_RECEIVING\_STAGE for F<sub>2</sub>, because the entire staging file is downloaded, M<sub>C</sub> does two things:
  - M<sub>C</sub> installs F<sub>2</sub> and sends out CMD\_REMOTE\_CO\_DONE to M<sub>B</sub> through C<sub>BA</sub> at time 8. Thus, F<sub>1</sub> is overwritten by F<sub>2</sub>.
  - M<sub>C</sub> moves the change order for F<sub>2</sub> from inbound log to outbound log. When the change order for F<sub>2</sub> is in outbound log, M<sub>C</sub> sends the change order for F<sub>2</sub> to all other machines, which is M<sub>A</sub> in this example. So, at time 8, M<sub>C</sub> sends out the change order for F<sub>2</sub> to M<sub>A</sub> through C<sub>CA</sub>.

At time 9:

When M<sub>A</sub> receives the change order for F<sub>2</sub> from M<sub>C</sub>, M<sub>A</sub> checks its version vector and decides to dampen this change order because this change order for F<sub>2</sub> is readily known. At time 9, M<sub>A</sub> sends out CMD\_REMOTE\_CO\_DONE with flag CO\_FLAG\_ABORT\_CO to M<sub>C</sub> through C<sub>CA</sub>.

At time 10:

When M<sub>C</sub> receives the change order for F<sub>2</sub> from M<sub>A</sub>, M<sub>C</sub> checks its version vector and decides to dampen this change order because this change order for F<sub>2</sub> is readily known. At time 10, M<sub>C</sub> sends out CMD\_REMOTE\_CO\_DONE with flag CO\_FLAG\_ABORT\_CO to M<sub>A</sub> through C<sub>AC</sub>.

### 4.3 COMM\_PACKET

This is an example to show COMM\_PACKET structure:

```
+0x000 Major : 0
+0x004 Minor : 8
+0x008 CsId : 1
+0x00c MemLen : 0x200
+0x010 PktLen : 0x184
+0x014 UpkLen : 0
+0x018 Pkt : 0x001a2df0
+0x01c DataName : null
+0x020 DataHandle : null
```

The bytes CommPkt->Pkt points to are:

```
01 00 04 00 00 00 00 00 00 00 02 00 04 00 00 00 22 01
00 00 03 00 32 00 00 00 10 00 00 00 0c 3e 3b 8c 27 64
f6 40 8f fd a4 5a 89 cf 5d 12 1a 00 00 00 43 00 46 00
53 00 2d 00 33 00 31 00 58 00 2d 00 32 00 41 00 30 00
32 00 00 00 04 00 32 00 00 00 10 00 00 00 e7 91 90 20
91 bd 34 45 bf e1 02 02 8f 21 4c eb 1a 00 00 00 43 00
46 00 53 00 2d 00 33 00 31 00 58 00 2d 00 32 00 41 00
30 00 31 00 00 00 05 00 60 00 00 00 10 00 00 00 0c 3e
3b 8c 27 64 f6 40 8f fd a4 5a 89 cf 5d 12 48 00 00 00
44 00 4f 00 4d 00 41 00 49 00 4e 00 20 00 53 00 59 00
53 00 54 00 45 00 4d 00 20 00 56 00 4f 00 4c 00 55 00
4d 00 45 00 20 00 28 00 53 00 59 00 53 00 56 00 4f 00
4c 00 20 00 53 00 48 00 41 00 52 00 45 00 29 00 00 00
08 00 62 00 00 00 10 00 00 00 fd f7 aa 2c 59 be d9 48
87 cf 5b e1 a9 23 16 ca 4a 00 00 00 38 00 30 00 38 00
42 00 39 00 38 00 46 00 45 00 2d 00 45 00 38 00 36 00
37 00 2d 00 34 00 46 00 33 00 36 00 2d 00 42 00 34 00
39 00 44 00 2d 00 44 00 44 00 36 00 36 00 36 00 39 00
38 00 36 00 36 00 42 00 44 00 33 00 00 00 06 00 14 00
00 00 10 00 00 00 9b 2d 92 76 3d 3f 2a 62 8a 20 b2 da
fe 56 80 ef 12 00 08 00 00 00 9c 43 1d 03 ef 84 c6 01
13 00 04 00 00 00 ff ff ff ff
```

In CommPkt->Pkt (see section 2.2.3.5), COMM\_COMMAND (see section 2.2.3.6.2) is:

```
02 00 04 00 00 00 22 01 00 00
```

This indicates a CMD\_START\_JOIN (0x0122) packet.

CMD\_START\_JOIN command is composed of the following sequence:

COMM\_BOP

COMM\_COMMAND

COMM\_TO

COMM\_FROM

COMM\_REPLICA

COMM\_CXTION

COMM\_JOIN\_GUID

COMM\_LAST\_JOIN\_TIME

COMM\_EOP

To interpret CommPkt->Pkt:

COMM\_BOP:

```
00 1a 2d f0 01 00 04 00 00 00 00 00 00 00
```

Every packet starts with this.

COMM\_COMMAND:

```
02 00 04 00 00 00 22 01 00 00
```

This is CMD\_START\_JOIN command - 0x0122.

COMM\_TO:

```
03 00 32 00 00 00 10 00 00 00 0c 3e 3b 8c 27 64 f6 40
8f fd a4 5a 89 cf 5d 12 1a 00 00 00 43 00 46 00 53 00
2d 00 33 00 31 00 58 00 2d 00 32 00 41 00 30 00 32 00
00 00
```

GUID: 8c3b3e0c-6427-40f6-8ffda45a89cf5d12; name: CFS-31X-2a02

COMM\_FROM:

```
04 00 32 00 00 00 10 00 00 00 e7 91 90 20 91 bd 34 45
bf e1 02 02 8f 21 4c eb 1a 00 00 00 43 00 46 00 53 00
2d 00 33 00 31 00 58 00 2d 00 32 00 41 00 30 00 31 00
00 00
```

GUID: 209091e7-bd91-4534-bfe102028f214ceb; name: CFS-31X-2a01

COMM\_REPLICA:

```
05 00 60 00 00 00 10 00 00 00 0c 3e 3b 8c 27 64 f6 40
8f fd a4 5a 89 cf 5d 12 48 00 00 00 44 00 4f 00 4d 00
41 00 49 00 4e 00 20 00 53 00 59 00 53 00 54 00 45 00
4d 00 20 00 56 00 4f 00 4c 00 55 00 4d 00 45 00 20 00
28 00 53 00 59 00 53 00 56 00 4f 00 4c 00 20 00 53 00
48 00 41 00 52 00 45 00 29 00 00 00
```

GUID: 8c3b3e0c-6427-40f6-8ffda45a89cf5d12; name: DOMAIN SYSTEM VOLUME (SYSVOL SHARE)

COMM\_CXTION:

```
08 00 62 00 00 00 10 00 00 00 fd f7 aa 2c 59 be d9 48 ..b.....,Y..H
87 cf 5b e1 a9 23 16 ca 4a 00 00 00 38 00 30 00 38 00 ..[#..J...8.0.8.
42 00 39 00 38 00 46 00 45 00 2d 00 45 00 38 00 36 00 B.9.8.F.E.-.E.8.6.
37 00 2d 00 34 00 46 00 33 00 36 00 2d 00 42 00 34 00 7.-.4.F.3.6.-.B.4.
39 00 44 00 2d 00 44 00 44 00 36 00 36 00 36 00 39 00 9.D.-.D.D.6.6.6.9.
38 00 36 00 36 00 42 00 44 00 33 00 00 00
```

GUID: 2caaf7fd-be59-48d9-87cf5be1a92316ca; name: 808B98FE-E867-4F36-B49D-DD6669866BD3

COMM\_JOIN\_GUID:

```
06 00 14 00 00 00 10 00 00 00 9b 2d 92 76 3d 3f 2a 62
8a 20 b2 da fe 56 80 ef
```

GUID: 9b 2d 92 76 3d 3f 2a 62 8a 20 b2 da fe 56 80 ef

COMM\_LAST\_JOIN\_TIME:

```
12 00 08 00 00 00 9c 43 1d 03 ef 84 c6 01
```

LastJoinTime: 9c431d03ef84c601.

COMM\_EOP:

```
13 00 04 00 00 00 ff ff ff ff
```

Every packet ends with this.

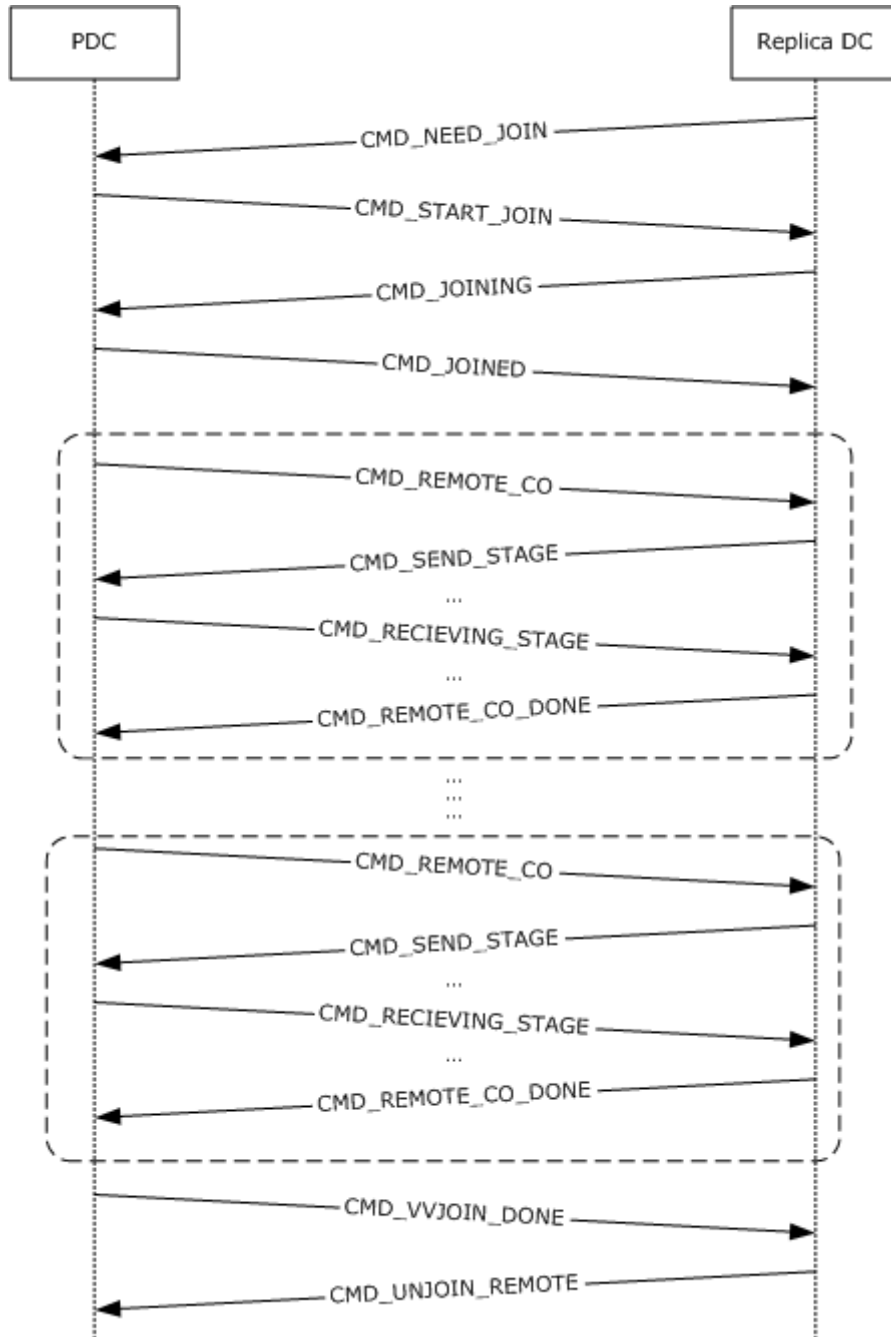
#### 4.4 SYSVOL Initial Sync

This is an example for initial sync for the SYSVOL folder:

- Primary domain controller (PDC): SHICO-TEMP-1
- Replica domain controller (DC): SHICO-TEMP-2

More specifically, communication packets are shown on the following connection: ConnGuid: 2d89345f-b2ac-4e89-8bdd0efa166b92e6 (from PDC to replica DC).

The following diagram shows the sequence of packets sent over the wire.



**Figure 5: Sequence of packets over the wire**

1. Replica DC sends out `CMD_NEED_JOIN` to PDC.
2. PDC sends out `CMD_START_JOIN` to replica DC.
3. Replica DC sends out `CMD_JOINING`.
4. PDC sends out `CMD_JOINED`.



Steps 1 through 4 show how a connection is joined.

5. PDC sends out CMD\_REMOTE\_CO.

Step 5 is repeated until all of the change orders are sent from the PDC to the replica DC.

6. Replica DC sends out CMD\_SEND\_STAGE.

7. PDC sends out CMD\_RECEIVING\_STAGE.

8. Replica DC sends out CMD\_REMOTE\_CO\_DONE.

Steps 5 through 8 are repeated until all the files are replicated from PDC to replica DC.

9. Once all of the change orders are sent out, the PDC sends out CMD\_VVJOIN\_DONE.

10. Replica DC sends out CMD\_UNJOIN\_REMOTE. At this moment, the PDC can still send out CMD\_RECEIVING\_STAGE to replica DC. But there is no more CMD\_REMOTE\_CO.

**Note** All symbols, such as CMD\_NEED\_JOIN, are specified in section 3.

#### 4.4.1 Replica DC Sends Out CMD\_NEED\_JOIN to PDC

COMM\_BOP (4 bytes):

```
00 00 00 00
```

COMM\_COMMAND (4 bytes):

```
21 01 00 00
```

COMMAND: CMD\_NEED\_JOIN

COMM\_TO (118 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 0
```

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: shico-temp-1.shico-office.nttest.microsoft.com

Notes:

54f4b21a-03fd-4374-8e3b2875e740d958 is objectGuid of the following object:

CN=SHICO-TEMP-1,CN=Domain System Volume (SYSVOL share),CN=File Replication Service,CN=System,DC=shico-office,DC=nttest,DC=microsoft,DC=com

shico-temp-1.shico-office.nttest.microsoft.com is the DNS host name of the primary domain controller (PDC).

COMM\_FROM (50 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 0
```

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: SHICO-TEMP-2

Notes:

- e5d187e6-12aa-48df-abc1d7940ae0804c is the objectGuid of the following object:
- CN=SHICO-TEMP-2,CN=Domain System Volume (SYSVOL share),CN=File Replication Service,CN=System,DC=shico-office,DC=nttest,DC=microsoft,DC=com

shico-temp-2 is the host name of the replica DC.

COMM\_REPLICA (96 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 0
```

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: DOMAIN SYSTEM VOLUME (SYSVOL SHARE)

Notes:

- 54f4b21a-03fd-4374-8e3b2875e740d958 is objectGuid of the following object:  
CN=SHICO-TEMP-1,CN=Domain System Volume (SYSVOL share),CN=File Replication Service,CN=System,DC=shico-office,DC=nttest,DC=microsoft,DC=com
- DOMAIN SYSTEM VOLUME (SYSVOL SHARE) is the name of the following object:  
CN=Domain System Volume (SYSVOL share),CN=File Replication Service,CN=System,DC=shico-office,DC=nttest,DC=microsoft,DC=com

COMM\_CXTION (118 bytes):

```
10 00 00 00 5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b
92 e6 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 00
```

GUID: 2d89345f-b2ac-4e89-8bdd0efa166b92e6

Name: shico-temp-1.shico-office.nttest.microsoft.com

Notes: In this case:

- 2d89345f-b2ac-4e89-8bdd0efa166b92e6 is the connection GUID of the SYSVOL volatile connection. During DCPromo, the replica DC dynamically creates a random GUID to be used as the volatile connection GUID, and then passes this connection GUID to PDC via the CxtionGuid parameter in FrsRpcStartPromotionParent.
- shico-temp-1.shico-office.nttest.microsoft.com is the DNS host name of the PDC.
- A volatile connection is used to seed SYSVOLs after dcpromo. If there is inactivity on an outbound volatile connection for more than 30 minutes, this connection is unjoined. An unjoin on an outbound volatile connection triggers a delete on that connection. This is to prevent the case where staging files are kept forever on the parent for a volatile connection.

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00
```

GUID: 00000000-0000-0000-0000000000000000

**Note** Since the join has not been started, the connection session GUID is set to all zeros.

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
01 00 00 00 00 00 00 00
```

COMM\_EOP (4 bytes):

```
ff ff ff ff
```

#### 4.4.2 PDC Sends Out CMD\_START\_JOIN to Replica DC

COMM\_BOP (4 bytes):

```
00 00 00 00
```

COMM\_COMMAND (4 bytes):

```
22 1 00 00
```

Command: CMD\_START\_JOIN

COMM\_TO (118 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 32 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
```

2e 00 63 00 6f 00 6d 00 00 0

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: shico-temp-2.shico-office.nttest.microsoft.com

COMM\_FROM (50 bytes):

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: SHICO-TEMP-1

COMM\_REPLICA (96 bytes):

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00  
45 00 29 00 00 00

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: DOMAIN SYSTEM VOLUME (SYSVOL SHARE)

COMM\_CXTION (118 bytes):

10 00 00 00 5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b  
92 e6 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00  
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00  
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00  
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00  
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00  
2e 00 63 00 6f 00 6d 00 00 0

GUID: 2d89345f-b2ac-4e89-8bdd0efa166b92e6

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_JOIN\_GUID (20 bytes):

10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00

GUID: 00000000-0000-0000-0000000000000000

COMM\_LAST\_JOIN\_TIME (8 bytes):

01 00 00 00 00 00 00 0

COMM\_EOP (4 bytes):

ff ff ff ff

#### 4.4.3 Replica DC Sends Out CMD\_JOINING

COMM\_BOP (4 bytes):

00 00 00 0

COMM\_COMMAND (4 bytes):

30 1 00 00

Command: CMD\_JOINING

COMM\_TO (118 bytes):

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00  
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00  
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00  
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00  
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00  
2e 00 63 00 6f 00 6d 00 00 00

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_FROM (50 bytes):

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: SHICO-TEMP-2

COMM\_REPLICA (96 bytes):

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00  
45 00 29 00 00 00

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: DOMAIN SYSTEM VOLUME (SYSVOL SHARE)

COMM\_CXTION (118 bytes):

```
10 00 00 00 5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b
92 e6 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 00
```

GUID: 2d89345f-b2ac-4e89-8bdd0efa166b92e6

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 48 61 c2 70 db 7e c9 39 bb 75 48 7f af b2
dc d5
```

GUID: 70c26148-7edb-39c9-bb75487fafb2dcd5

**Note** Replica domain controller (DC) creates this GUID.

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
01 00 00 00 00 00 00 00
```

COMM\_VVECTOR (28 bytes):

```
18 00 00 00 e3 4e 49 5a bb 89 c6 01 fe 3d 90 ff 4b c4
b9 45 8b d0 62 f5 34 97 37 28
```

VSN: 0x01c689bb 5a494ee3

GUID: ff903dfe-c44b-45b9-8bd062f534973728

Notes:

- ff903dfe-c44b-45b9-8bd062f534973728 is the originator GUID for this Replica DC. See COMM\_REPLICA\_VERSION\_GUID.
- The replica DC has received and processed all changes up to the Volume Sequence Number (VSN) 0x01c689bb 5a494ee3. Each volume maintains a sequence number (that increases by monotone) and adds 1 whenever a new VSN is required. During volume initialization, the VSN is initialized to the current time.

COMM\_JOIN\_TIME (12 bytes):

08 00 00 00 f4 e9 4 1b b5 89 c6 01

FILETIME: 0x01c689b5 1b04e9f4

COMM\_REPLICA\_VERSION\_GUID (20 bytes):

10 00 00 00 fe 3d 90 ff 4b c4 b9 45 8b d0 62 f5 34 97  
37 28

GUID: ff903dfe-c44b-45b9-8bd062f534973728

**Note** When a replica set is initialized on a member machine, the member machine generates a new unique GUID to identify itself; ff903dfe-c44b-45b9-8bd062f534973728 is generated by the replica DC to represent the replica DC.

COMM\_COMPRESSION\_GUID (16 bytes):

d2 f7 d2 64 95 26 6d 43 88 30 8d 3c 58 70 1e 15

GUID: 64d2f7d2-2695-436d-88308d3c58701e15

**Note** The compression algorithm that NTFRS uses is specified in section 2.2.3.6.12.

COMM\_EOP (4 bytes):

ff ff ff ff

#### 4.4.4 PDC Sends Out CMD\_JOINED

COMM\_BOP (4 bytes):

00 00 00 00

COMM\_COMMAND (4 bytes):

28 01 00 00

Command: CMD\_JOINED

COMM\_TO (118 bytes):

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00  
74 00 65 00 6d 00 70 00 2d 00 32 00 2e 00 73 00 68 00  
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00  
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00  
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00  
2e 00 63 00 6f 00 6d 00 00 00

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: shico-temp-2.shico-office.nttest.microsoft.com

COMM\_FROM (50 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 0
```

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: SHICO-TEMP-1

COMM\_REPLICA (96 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: DOMAIN SYSTEM VOLUME (SYSVOL SHARE)

COMM\_CXTION (118 bytes):

```
10 00 00 00 5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b
92 e6 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 00
```

GUID: 2d89345f-b2ac-4e89-8bdd0efa166b92e6

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 48 61 c2 70 db 7e c9 39 bb 75 48 7f af b2
dc d5
```

GUID: 70c26148-7edb-39c9-bb75487fafb2dcd5

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
4e 60 39 1b b5 89 c6 01
```



FILETIME: 0x01c689b5 1b39604e

COMM\_EOP (4 bytes):

ff ff ff ff

#### 4.4.5 PDC Sends Out CMD\_REMOTE\_CO

COMM\_BOP (4 bytes):

00 00 00 00

COMM\_COMMAND (4 bytes):

18 2 00 00

Command: CMD\_REMOTE\_CO

COMM\_TO (118 bytes):

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00  
74 00 65 00 6d 00 70 00 2d 00 32 00 2e 00 73 00 68 00  
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00  
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00  
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00  
2e 00 63 00 6f 00 6d 00 00 00

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: shico-temp-2.shico-office.nttest.microsoft.com

COMM\_FROM (50 bytes):

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: SHICO-TEMP-1

COMM\_REPLICA (96 bytes):

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00  
45 00 29 00 00 00

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: DOMAIN SYSTEM VOLUME (SYSVOL SHARE)

COMM\_CXTION (118 bytes):

```
10 00 00 00 5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b
92 e6 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 00
```

GUID: 2d89345f-b2ac-4e89-8bdd0efa166b92e6

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 48 61 c2 70 db 7e c9 39 bb 75 48 7f af b2
dc d5
```

GUID: 70c26148-7edb-39c9-bb75487fafb2dcd5

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
4e 60 39 1b b5 89 c6 01
```

FILETIME: 0x01c689b5 1b39604e

COMM\_REMOTE\_CO (796 bytes):

```
18 03 00 00 03 00 00 00 28 00 04 00 00 00 00 00 14 00
00 00 00 01 00 00 01 00 00 00 10 00 00 00 00 00 00 00
03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 7c 59 ee 3d b8 89 c6 01 f8 8d bc 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 01 00 00 00 01 00 00 00 3c 8f 6e 16 2c 97 d4 47
bf 34 e6 32 75 13 5b 2b 76 65 78 79 63 b8 da 41 b1 1a
41 63 46 eb be b3 4e c8 ec 1a 7a 35 0f 4e 99 a4 f2 ea
94 ff ab 08 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0 80 4c
5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b 92 e6 9a 9b
34 1b b5 89 c6 01 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
08 c5 e8 00 00 00 00 00 2e b6 2e dc af 89 c6 01 10 00
50 00 6f 00 6c 00 69 00 63 00 69 00 65 00 73 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



```

(0x01c689b5 1b349b9a)
+0x0c8 Spare2Ull      : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid    : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid    : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs     : 0 0 0 0
+0x0f4 Spare2Wcs     : 0 0 0 0
+0x0f8 Extension     : 8 c5 e8 0
+0x0fc Spare2Bin     : 0 0 0 0
+0x100 EventTime     : 2e b6 2e dc af 89 c6 1
+0x108 FileNameLength : 10 0 (16 bytes)
+0x10a FileName      : 50 0 6f 0 6c 0 69 0
                       63 0 69 0 65 0 73 0 (Policies)

```

#### Notes:

- SequenceNumber in the change order is initialized to 0. Once it is inserted into outbound log, it assumes the largest value among all change orders in outbound log plus one.
- Spare1Wcs, Spare2Wcs, Extension, and Spare2Bin occupy 16 bytes on 32-bit architectures and occupy 32 bytes on 64-bit architectures. The CO is included in the staging file header and in the change order comm packet, causing 32-64 interoperation problems in both the staging file processing and comm transfers between 32- and 64-bit machines. The contents of these pointers are irrelevant in both comm packets and staging files since they point to allocated buffers. To preserve the size of the change order in the staging file on 64-bit machines, the unused field Spare2Guid is unioned with the two 8-byte pointers Spare1Wcs and Spare2Wcs, saving 16 bytes. On 32-bit architectures, these fields are left separate.

#### COMM\_CO\_EXTENSION\_2 (72 bytes):

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 00 5c 87 40 1b b5 89 c6 01

```

#### Type\_CHANGE\_ORDER\_RECORD\_EXTENSION\_\*

```

+0x000 FieldSize      : 48 0 0 0 (0x48)
+0x004 Major         : 1 0 (1)
+0x006 OffsetCount   : 2 0 (2)
+0x008 Offset        : 18 0 0 0 30 0 0 0 (0x00000018 and 0x00000030)
+0x010 OffsetLast    : 0 0 0 0 0 0 0 0 (0)
+0x018 DataChecksum  : _DATA_EXTENSION_CHECKSUM_
  +0x000 Prefix      : _DATA_EXTENSION_PREFIX_
  +0x000 Size         : 18 0 0 0 (0x18)
  +0x004 Type         : 1 0 0 0 (1)
  +0x008 Data         : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x030 DataRetryTimeout : _DATA_EXTENSION_RETRY_TIMEOUT_
  +0x000 Prefix      : _DATA_EXTENSION_PREFIX_
  +0x000 Size         : 18 0 0 0 (0x18)
  +0x004 Type         : 2 0 0 0 (2)
  +0x008 Count        : 0 0 0 0 0 0 0 0 (0)
  +0x010 FirstTryTime : 5c 87 40 1b b5 89 c6 1

```

#### COMM\_EOP (4 bytes):

```
ff ff ff ff
```

Step 5 is repeated until all of the change orders are sent from the PDC to the replica DC.

#### 4.4.6 Replica DC Sends Out CMD\_SEND\_STAGE

COMM\_BOP (4 bytes):

```
00 00 00 00
```

COMM\_COMMAND (4 bytes):

```
28 2 00 00
```

Command: CMD\_SEND\_STAGE

COMM\_TO (118 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00  
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00  
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00  
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00  
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00  
2e 00 63 00 6f 00 6d 00 00 00
```

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_FROM (50 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: SHICO-TEMP-2

COMM\_REPLICA (96 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00  
45 00 29 00 00 00
```

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: DOMAIN SYSTEM VOLUME (SYSVOL SHARE)

COMM\_CXTION (118 bytes):

```
10 00 00 00 5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b
92 e6 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 00
```

**GUID:** 2d89345f-b2ac-4e89-8bdd0efa166b92e6

**Name:** shico-temp-1.shico-office.nttest.microsoft.com

**COMM\_JOIN\_GUID (20 bytes):**

```
10 00 00 00 48 61 c2 70 db 7e c9 39 bb 75 48 7f af b2
dc d5
```

**GUID:** 70c26148-7edb-39c9-bb75487fafb2dcd5

**COMM\_LAST\_JOIN\_TIME (8 bytes):**

```
4e 60 39 1b b5 89 c6 01
```

**FILETIME:** 0x01c689b5 1b39604e

**COMM\_BLOCK\_SIZE (8 bytes):**

```
00 00 00 00 00 00 00 00
```

**COMM\_FILE\_SIZE (8 bytes):**

```
00 00 00 00 00 00 00 00
```

**COMM\_FILE\_OFFSET (8 bytes):**

```
00 00 00 00 00 00 00 00
```

**COMM\_CO\_GUID (20 bytes):**

```
10 00 00 00 3c 8f 6e 16 2c 97 d4 47 bf 34 e6 32 75 13
5b 2b
```

**COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):**

```
00 00 00 00
```



```

+0x038 FrsVsn          : 7c 59 ee 3d b8 89 c6 1
(0x01c689b8 3dee597c)
+0x040 FileUsn        : 70 ed c7 0 0 0 0 0
(0x00000000 00c7ed70)
+0x048 JrnlUsn        : 0 0 0 0 0 0 0 0
+0x050 JrnlFirstUsn   : 0 0 0 0 0 0 0 0
+0x058 OriginalReplicaNum : 1 0 0 0 (1)
+0x05c NewReplicaNum  : 1 0 0 0 (1)
+0x060 ChangeOrderGuid : 3c 8f 6e 16 2c 97 d4 47
                          bf 34 e6 32 75 13 5b 2b
(166e8f3c-972c-47d4-bf34e63275135b2b)
+0x070 OriginatorGuid : 76 65 78 79 63 b8 da 41
                          b1 la 41 63 46 eb be b3
(79786576-b863-41da-b11a416346ebbeb3)
+0x080 FileGuid       : 4e c8 ec 1a 7a 35 f 4e
                          99 a4 f2 ea 94 ff ab 8
(1aecc84e-357a-4e0f-99a4f2ea94ffab08)
+0x090 OldParentGuid  : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
(1e3431bb-5b35-47a5-ae555ba62c216d7)
+0x0a0 NewParentGuid  : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
(1e3431bb-5b35-47a5-ae555ba62c216d7)
+0x0b0 CxtionGuid     : 5f 34 89 2d ac b2 89 4e
                          8b dd e fa 16 6b 92 e6
(2d89345f-b2ac-4e89-8bdd0efa166b92e6)
+0x0c0 AckVersion     : 9a 9b 34 1b b5 89 c6 1
(0x01c689b5 1b349b9a)
+0x0c8 Spare2Ull      : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid     : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid     : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs      : 0 0 0 0
+0x0f4 Spare2Wcs      : 0 0 0 0
+0x0f8 Extension      : d8 49 a1 1
+0x0fc Spare2Bin      : 0 0 0 0
+0x100 EventTime      : 2e b6 2e dc af 89 c6 1
+0x108 FileNameLength : 10 0 (16 bytes)
+0x10a FileName       : 50 0 6f 0 6c 0 69 0
                          63 0 69 0 65 0 73 0 (Policies)

```

**COMM\_CO\_EXTENSION\_2 (72 bytes):**

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 00 b4 cb e9 1b b5 89 c6 01

```

**Type\_CHANGE\_ORDER\_RECORD\_EXTENSION\_\***

```

+0x000 FieldSize      : 48 0 0 0
+0x004 Major          : 1 0
+0x006 OffsetCount    : 2 0
+0x008 Offset         : 18 0 0 0 30 0 0 0
+0x010 OffsetLast     : 0 0 0 0 0 0 0 0
+0x018 DataChecksum   : _DATA_EXTENSION_CHECKSUM_
+0x000 Prefix         : _DATA_EXTENSION_PREFIX_
+0x000 Size           : 18 0 0 0
+0x004 Type           : 1 0 0 0
+0x008 Data           : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x030 DataRetryTimeout : _DATA_EXTENSION_RETRY_TIMEOUT_
+0x000 Prefix         : _DATA_EXTENSION_PREFIX_
+0x000 Size           : 18 0 0 0
+0x004 Type           : 2 0 0 0

```



+0x008 Count : 0 0 0 0 0 0 0 0  
+0x010 FirstTryTime : b4 cb e9 1b b5 89 c6 1

COMM\_EOP (4 bytes):

ff ff ff ff

#### 4.4.7 PDC Sends Out CMD\_RECEIVING\_STAGE

COMM\_BOP (4 bytes):

00 00 00 00

COMM\_COMMAND (4 bytes):

38 02 00 00

Command: CMD\_RECEIVING\_STAGE

COMM\_TO (118 bytes):

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00  
74 00 65 00 6d 00 70 00 2d 00 32 00 2e 00 73 00 68 00  
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00  
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00  
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00  
2e 00 63 00 6f 00 6d 00 00 00

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: shico-temp-2.shico-office.nttest.microsoft.com

COMM\_FROM (50 bytes):

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: SHICO-TEMP-1

COMM\_REPLICA (96 bytes):

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00

45 00 29 00 00 00

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: DOMAIN SYSTEM VOLUME (SYSVOL SHARE)

COMM\_CXTION (118 bytes):

```
10 00 00 00 5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b
92 e6 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 00
```

GUID: 2d89345f-b2ac-4e89-8bdd0efa166b92e6

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 48 61 c2 70 db 7e c9 39 bb 75 48 7f af b2
dc d5
```

GUID: 70c26148-7edb-39c9-bb75487fafb2dcd5

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
4e 60 39 1b b5 89 c6 1
```

FILETIME: 0x01c689b5 1b39604e

COMM\_BLOCK (1,266 bytes):

```
ee 04 00 00 00 00 00 00 03 00 00 00 00 00 00 00 04
00 00 00 00 00 00 00 00 00 00 00 f2 7a c2 29 af 89 c6 01
e4 b1 8b dc af 89 c6 01 00 a2 c9 29 af 89 c6 01 e4 b1
8b dc af 89 c6 01 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 10 00 00 00 00 00 00 00 00 01 00 00 08 04
04 01 00 00 00 00 05 00 00 00 00 00 00 00 01 00 00 00
10 00 00 00 00 00 00 00 03 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 7c 59 ee 3d
b8 89 c6 01 70 ed c7 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 01 00 00 00 01 00 00 00
3c 8f 6e 16 2c 97 d4 47 bf 34 e6 32 75 13 5b 2b 76 65
78 79 63 b8 da 41 b1 1a 41 63 46 eb be b3 4e c8 ec 1a
7a 35 0f 4e 99 a4 f2 ea 94 ff ab 08 bb 31 34 1e 35 5b
a5 47 ae e5 55 ba 62 c2 16 d7 bb 31 34 1e 35 5b a5 47
ae e5 55 ba 62 c2 16 d7 5f 34 89 2d ac b2 89 4e 8b dd
0e fa 16 6b 92 e6 9a 9b 34 1b b5 89 c6 01 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2e b6
2e dc af 89 c6 01 10 00 50 00 6f 00 6c 00 69 00 63 00
69 00 65 00 73 00 00 00 00 00 00 00 00 00 00 00 00 00
```



```

(0x01c689af dc8bb1e4)
+0x020 AllocationSize : 0 0 0 0 0 0 0 0 (0)
+0x028 EndOfFile : 0 0 0 0 0 0 0 0 (0)
+0x030 FileAttributes : 10 0 0 0 0 0 0 0 (0x10)
+0x050 ChangeOrderCommand : _CHANGE_ORDER_RECORD_
+0x000 SequenceNumber : 1 0 0 0 (1)
+0x004 Flags : 8 4 4 1
(0x01040408 - CO_FLAG_COMPRESSED_STAGE | CO_FLAG_VVJOIN_TO_ORIG | CO_FLAG_NEW_FILE |
CO_FLAG_LOCATION_CMD)
+0x008 IFlags : 0 0 0 0 (0)
+0x00c State : 5 0 0 0
(0x5 - Allocating Staging File space for remote Change Order)
+0x010 ContentCmd : 0 0 0 0
+0x014 LocationCmd : 1 0 0 0 (1)
+0x018 FileAttributes : 10 0 0 0 (0x10)
+0x01c FileVersionNumber : 0 0 0 0 (0)
+0x020 PartnerAckSeqNumber : 3 0 0 0 0 0 0 0 (3)
+0x028 FileSize : 0 0 0 0 0 0 0 0 (0)
+0x030 FileOffset : 0 0 0 0 0 0 0 0 (0)
+0x038 FrsVsn : 7c 59 ee 3d b8 89 c6 1
(01c689b8 3dee597c)
+0x040 FileUsn : 70 ed c7 0 0 0 0 0 (0xc7ed70)
+0x048 JrnlUsn : 0 0 0 0 0 0 0 0 (0)
+0x050 JrnlFirstUsn : 0 0 0 0 0 0 0 0 (0)
+0x058 OriginalReplicaNum : 1 0 0 0 (1)
+0x05c NewReplicaNum : 1 0 0 0 (1)
+0x060 ChangeOrderGuid : 3c 8f 6e 16 2c 97 d4 47
bf 34 e6 32 75 13 5b 2b
(166e8f3c-972c-47d4-bf34e63275135b2b)
+0x070 OriginatorGuid : 76 65 78 79 63 b8 da 41
b1 1a 41 63 46 eb be b3
(79786576-b863-41da-b11a416346ebbeb3)
+0x080 FileGuid : 4e c8 ec 1a 7a 35 f 4e
99 a4 f2 ea 94 ff ab 8
(1aecc84e-357a-4e0f-99a4f2ea94ffab08)
+0x090 OldParentGuid : bb 31 34 1e 35 5b a5 47
ae e5 55 ba 62 c2 16 d7
(1e3431bb-5b35-47a5-ae555ba62c216d7)
+0x0a0 NewParentGuid : bb 31 34 1e 35 5b a5 47
ae e5 55 ba 62 c2 16 d7
(1e3431bb-5b35-47a5-ae555ba62c216d7)
+0x0b0 CxtionGuid : 5f 34 89 2d ac b2 89 4e
8b dd e fa 16 6b 92 e6
(2d89345f-b2ac-4e89-8bdd0efa166b92e6)
+0x0c0 AckVersion : 9a 9b 34 1b b5 89 c6 1
(0x01c689b5 1b349b9a)
+0x0c8 Spare2Ull : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid : 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid : 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs : 0 0 0 0
+0x0f4 Spare2Wcs : 0 0 0 0
+0x0f8 Extension : 0 0 0 0
+0x0fc Spare2Bin : 0 0 0 0
+0x100 EventTime : 2e b6 2e dc af 89 c6 1
(0x01c689af dc2eb62e)
+0x108 FileNameLength : 10 0 (0x10)
+0x10a FileName : 50 0 6f 0 6c 0 69 0 63
0 69 0 65 0 73 0 0 0 0
(Policies)
+0x368 FileObjId : _FILE_OBJECTID_BUFFER
+0x000 ObjectId : 4e c8 ec 1a 7a 35 f 4e
99 a4 f2 ea 94 ff ab 8
(1aecc84e-357a-4e0f-99a4f2ea94ffab08)
+0x010 BirthVolumeId : 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
+0x020 BirthObjectId : 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
+0x030 DomainId : 0 0 0 0 0 0 0 0

```

```

0 0 0 0 0 0 0 0
+0x3a8 CocExt          :  _CO_RECORD_EXTENSION_WIN2K_
+0x000 FieldSize      :  0 0 0 0
+0x004 Major          :  0 0
+0x006 OffsetCount    :  0 0
+0x008 Offset         :  0 0 0 0
+0x00c OffsetLast     :  0 0 0 0
+0x010 DataChecksum   :  _DATA_EXTENSION_CHECKSUM_
+0x000 Prefix         :  0 0 0 0 0 0 0 0
+0x008 Data           :  0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0
+0x3d0 CompressionGuid :  d2 f7 d2 64 95 26 6d 43
                        88 30 8d 3c 58 70 1e 15
(64d2f7d2-2695-436d-88308d3c58701e15)
+0x3e0 EncryptedDataHigh :  0 0 0 0 (0)
+0x3e4 EncryptedDataLow  :  0 0 0 0 (0)
+0x3e8 EncryptedDataSize :  0 0 0 0 0 0 0 (0)
+0x3f0 ReparseDataPresent :  0 0 0 0 (0)
+0x3f4 ReparsePointDataHigh :  0 0 0 0 (0)
+0x3f8 ReparsePointDataLow :  0 4 0 0 (0x400)

```

The real compressed data is:

```

00 00 00 00 eb b0 20 03 00 00 00 02 00 30 78 01 83 00
40 04 20 01 00 14 9c 14 00 58 aa 24 00 18 5c 00 5c 30
00 1c 01 01 90 50 00 00 05 20 00 2c 20 00 2c 01 09 03
bc 05 12 00 1e 02 00 2c 01 02 0c 00 0e 00 00 14 00 a9
00 54 12 00 05 36 0b 01 26 0b 01 92 00 b2 a0 0a 26 00
18 03 4e 08 a6 25 01 0b 74 0b 18 01 31 a0 09 6b 02 17
00 2f ff 58 01 1f 00 09 17 01 83 00 03 2f 10 af 0e 17
00 87 02 2f 08 a3 00 03 87 10 05 87 7d 01 b7 00 16 57
82 59 85 15 01 91 80 4b 24 28 00 bf 01 80 6d 05 01 51
00 05 02 15 00 03 57 c2 d9 22 0f 2d 80 42 bc dd 3d e9
5b 08 82 5f 11 81 98 00 e0 01 98 11 02 00 1c d2 00 01
4b 02 4a 81 31 01 85 95 82 54 62 07 02 2d 00 00 40 84
03 01 03 4e 00 c8 ec 1a 7a 35 0f 4e 99 80 a4 f2 ea 94
ff ab 08 08 0d 01 22 05

```

COMM\_BLOCK\_SIZE (8 bytes):

```
ee 4 00 00 00 00 00 00
```

Size: 0x4ee

COMM\_FILE\_SIZE (8 bytes):

```
ee 4 00 00 00 00 00 00
```

Size: 0x4ee

COMM\_FILE\_OFFSET (8 bytes):

```
00 00 00 00 00 00 00 00
```

Offset: 0

COMM\_CO\_GUID (20 bytes):

```
10 00 00 00 3c 8f 6e 16 2c 97 d4 47 bf 34 e6 32 75 13
5b 2b
```

GUID: 166e8f3c-972c-47d4-bf34e63275135b2b

COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):

```
00 00 00 00
```

Sequence Number: 0

COMM\_EOP (4 bytes):

```
ff ff ff ff
```

#### 4.4.8 Replica DC Sends Out CMD\_REMOTE\_CO\_DONE

COMM\_BOP (4 bytes):

```
00 00 00 00
```

COMM\_COMMAND (4 bytes):

```
50 02 00 00
```

Command: CMD\_REMOTE\_CO\_DONE

COMM\_TO (118 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 00
```

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_FROM (50 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: SHICO-TEMP-2

COMM\_REPLICA (96 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: DOMAIN SYSTEM VOLUME (SYSVOL SHARE)

COMM\_CXTION (118 bytes):

```
10 00 00 00 5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b
92 e6 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 00
```

GUID: 2d89345f-b2ac-4e89-8bdd0efa166b92e6

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 48 61 c2 70 db 7e c9 39 bb 75 48 7f af b2
dc d5
```

GUID: 70c26148-7edb-39c9-bb75487fafb2dcd5

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
4e 60 39 1b b5 89 c6 01
```

FILETIME: 0x01c689b5 1b39604e

COMM\_BLOCK\_SIZE (8 bytes):

```
00 00 00 00 00 00 00 00
```

COMM\_FILE\_SIZE (8 bytes):

00 00 00 00 00 00 00 00

COMM\_FILE\_OFFSET (8 bytes):

00 00 00 00 00 00 00 00

COMM\_GVSN (28 bytes):

18 00 00 00 7c 59 ee 3d b8 89 c6 01 76 65 78 79 63 b8  
da 41 b1 1a 41 63 46 eb be b3

VSN: 7c 59 ee 3d b8 89 c6 1

GUID: 76 65 78 79 63 b8 da 41 b1 1a 41 63 46 eb be b3

COMM\_CO\_GUID (20 bytes):

10 00 00 00 3c 8f 6e 16 2c 97 d4 47 bf 34 e6 32 75 13  
5b 2b

GUID: 3c 8f 6e 16 2c 97 d4 47 bf 34 e6 32 75 13 5b 2b

COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):

03 00 00 00

Sequence Number: 3

COMM\_REMOTE\_CO (796 bytes):

18 03 00 00 01 00 00 00 0a 04 04 01 01 00 00 00 16 00  
00 00 00 00 00 00 01 00 00 00 10 00 00 00 00 00 00 00  
03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 7c 59 ee 3d b8 89 c6 01 80 f6 c7 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 01 00 00 00 01 00 00 00 3c 8f 6e 16 2c 97 d4 47  
bf 34 e6 32 75 13 5b 2b 76 65 78 79 63 b8 da 41 b1 1a  
41 63 46 eb be b3 4e c8 ec 1a 7a 35 0f 4e 99 a4 f2 ea  
94 ff ab 08 bb 31 34 1e 35 5b a5 47 ae e5 55 ba 62 c2  
16 d7 bb 31 34 1e 35 5b a5 47 ae e5 55 ba 62 c2 16 d7  
5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b 92 e6 9a 9b  
34 1b b5 89 c6 01 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
d8 49 a1 01 00 00 00 00 2e b6 2e dc af 89 c6 01 10 00  
50 00 6f 00 6c 00 69 00 63 00 69 00 65 00 73 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00





```

+0x0c8 Spare2Ull      : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid    : 0 0 0 0 0 0 0 0
                    : 0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid    : 0 0 0 0 0 0 0 0
                    : 0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs     : 0 0 0 0
+0x0f4 Spare2Wcs     : 0 0 0 0
+0x0f8 Extension     : d8 49 a1 1
+0x0fc Spare2Bin     : 0 0 0 0
+0x100 EventTime     : 2e b6 2e dc af 89 c6 1
(0x01c689af dc2eb62e)
+0x108 FileNameLength : 10 0 (16 bytes)
+0x10a FileName      : 50 0 6f 0 6c 0 69 0
                    : 63 0 69 0 65 0 73 0 (Policies)

```

#### COMM\_CO\_EXTENSION\_2 (72 bytes):

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 00 b4 cb e9 1b b5 89 c6 01

```

#### Type\_CHANGE\_ORDER\_RECORD\_EXTENSION\_\*

```

+0x000 FieldSize      : 48 0 0 0
+0x004 Major          : 1 0
+0x006 OffsetCount    : 2 0
+0x008 Offset         : 18 0 0 0 30 0 0 0
+0x010 OffsetLast     : 0 0 0 0 0 0 0 0
+0x018 DataChecksum   : _DATA_EXTENSION_CHECKSUM_
  +0x000 Prefix       : _DATA_EXTENSION_PREFIX_
  +0x000 Size          : 18 0 0 0
  +0x004 Type          : 1 0 0 0
  +0x008 Data          : 0 0 0 0 0 0 0 0
                    : 0 0 0 0 0 0 0 0
+0x030 DataRetryTimeout : _DATA_EXTENSION_RETRY_TIMEOUT_
  +0x000 Prefix       : _DATA_EXTENSION_PREFIX_
  +0x000 Size          : 18 0 0 0
  +0x004 Type          : 2 0 0 0
  +0x008 Count        : 0 0 0 0 0 0 0 0
  +0x010 FirstTryTime : b4 cb e9 1b b5 89 c6 1
(0x01c689b5 1be9cbb4)

```

#### COMM\_EOP (4 bytes):

```
ff ff ff ff
```

Steps 7 through 9 are repeated until all the files are replicated from PDC to replica DC.

### 4.4.9 Once All Change Orders are Sent Out, PDC Sends Out CMD\_VVJOIN\_DONE

#### COMM\_BOP (4 bytes):

```
00 00 00 00
```

#### COMM\_COMMAND (4 bytes):

36 01 00 00

Command: CMD\_VVJOIN\_DONE

COMM\_TO (118 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 32 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 00
```

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: shico-temp-2.shico-office.nttest.microsoft.com

COMM\_FROM (50 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: SHICO-TEMP-1

COMM\_REPLICA (96 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 0
```

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: DOMAIN SYSTEM VOLUME (SYSVOL SHARE)

COMM\_CXTION (118 bytes):

```
10 00 00 00 5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b
92 e6 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 0
```

GUID: 2d89345f-b2ac-4e89-8bdd0efa166b92e6

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_JOIN\_GUID (20 bytes):

10 00 00 00 48 61 c2 70 db 7e c9 39 bb 75 48 7f af b2  
dc d5

GUID: 70c26148-7edb-39c9-bb75487fafb2dcd5

COMM\_LAST\_JOIN\_TIME (8 bytes):

4e 60 39 1b b5 89 c6 01

FILETIME: 0x01c689b5 1b39604e

COMM\_EOP (4 bytes):

ff ff ff ff

#### 4.4.10 Replica DC Sends Out CMD\_UNJOIN\_REMOTE

COMM\_BOP (4 bytes):

00 00 00 00

COMM\_COMMAND (4 bytes):

48 1 00 00

Command: CMD\_UNJOIN\_REMOTE

COMM\_TO (118 bytes):

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00  
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00  
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00  
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00  
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00  
2e 00 63 00 6f 00 6d 00 00 00

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_FROM (50 bytes):

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

GUID: e5d187e6-12aa-48df-abc1d7940ae0804c

Name: SHICO-TEMP-2

COMM\_REPLICA (96 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

GUID: 54f4b21a-03fd-4374-8e3b2875e740d958

Name: DOMAIN SYSTEM VOLUME (SYSVOL SHARE)

COMM\_CXTION (118 bytes):

```
10 00 00 00 5f 34 89 2d ac b2 89 4e 8b dd 0e fa 16 6b
92 e6 5e 00 00 00 73 00 68 00 69 00 63 00 6f 00 2d 00
74 00 65 00 6d 00 70 00 2d 00 31 00 2e 00 73 00 68 00
69 00 63 00 6f 00 2d 00 6f 00 66 00 66 00 69 00 63 00
65 00 2e 00 6e 00 74 00 74 00 65 00 73 00 74 00 2e 00
6d 00 69 00 63 00 72 00 6f 00 73 00 6f 00 66 00 74 00
2e 00 63 00 6f 00 6d 00 00 00
```

GUID: 2d89345f-b2ac-4e89-8bdd0efa166b92e6

Name: shico-temp-1.shico-office.nttest.microsoft.com

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 48 61 c2 70 db 7e c9 39 bb 75 48 7f af b2
dc d5
```

GUID: 70c26148-7edb-39c9-bb75487fafb2dcd5

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
4e 60 39 1b b5 89 c6 1
```

FILETIME: 0x01c689b5 1b39604e

COMM\_EOP (4 bytes):

```
ff ff ff ff
```

## 4.5 Normal Sync

SHICO-TEMP-1 and SHICO-TEMP-2 are already in sync through initial sync. The following examples show how changes from SHICO-TEMP-1 are replicated down to SHICO-TEMP-2.

The upstream partner is always SHICO-TEMP1. The downstream partner is always SHICO-TEMP-2. All changes happen on connection 08f7a3f0-e05c-4994-9a60e7cd4057ec6f.

## 4.5.1 Copy a File

Copy Regedit.exe to SYSVOL and replicate from SHICO-TEMP-1 to SHICO-TEMP-2. Because this file cannot be sent in one packet, two packets are used. This is to show how files are segmented.

### 4.5.1.1 Upstream Partner Sends Out CMD\_REMOTE\_CO

COMM\_BOP (4 bytes):

```
00 00 00 00
```

COMM\_COMMAND (4 bytes): 18 2 00 0

COMM\_TO (50 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

COMM\_FROM (50 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 0
```

COMM\_REPLICA (96 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 0
```

COMM\_CXTION (98 bytes):

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 ae 1b b7 65 7a c4 40 db c9 f8 bd 7e 32 65
43 41
```

COMM\_LAST\_JOIN\_TIME (8 bytes):



```

+0x014 LocationCmd           : 0 0 0 0 (0x0)
+0x018 FileAttributes       : 20 0 0 0
+0x01c FileVersionNumber   : 0 0 0 0
+0x020 PartnerAckSeqNumber : 1a 0 0 0 0 0 0 0
+0x028 FileSize            : 0 0 2 0 0 0 0 0
+0x030 FileOffset          : 0 0 0 0 0 0 0 0
+0x038 FrsVsn              : 1d f8 9 e6 9 90 c6 1
+0x040 FileUsn              : 60 b1 c9 0 0 0 0 0
+0x048 JrnlUsn              : b0 b0 c9 0 0 0 0 0
+0x050 JrnlFirstUsn        : b0 b0 c9 0 0 0 0 0
+0x058 OriginalReplicaNum  : 1 0 0 0
+0x05c NewReplicaNum       : 1 0 0 0
+0x060 ChangeOrderGuid     : 4d c d2 35 d b9 9 4d
                               bd 87 e2 c6 b0 8d 59 21
+0x070 OriginatorGuid      : 76 65 78 79 63 b8 da 41
                               b1 1a 41 63 46 eb be b3
+0x080 FileGuid            : de fb 44 7f b7 58 d3 4a
                               8d 5 2a 3a b ec 17 d0
+0x090 OldParentGuid       : e6 87 d1 e5 aa 12 df 48
                               ab c1 d7 94 a e0 80 4c
+0x0a0 NewParentGuid       : e6 87 d1 e5 aa 12 df 48
                               ab c1 d7 94 a e0 80 4c
+0x0b0 CxtionGuid          : 5e 1b 8b 23 ea ac 7 4e
                               be ca e7 ad 76 85 16 ad
+0x0c0 AckVersion          : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull           : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid          : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid          : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs           : 0 0 0 0
+0x0f4 Spare2Wcs           : 0 0 0 0
+0x0f8 Extension           : f8 1d e8 0
+0x0fc Spare2Bin           : 0 0 0 0
+0x100 EventTime           : ca 58 93 64 f1 8f c6 1
+0x108 FileNameLength      : 16 0
+0x10a FileName            : 72 0 65 0 67 0 65 0
                               64 0 69 0 74 0 2e 0
                               65 0 78 0 65 0 (regedit.exe)

```

**COMM\_CO\_EXTENSION\_2 (72 bytes):**

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00
00 00 00 00 00 00 00 18 00 00 00 01 00 00 00 57 68
ac c0 53 63 0b ac fb b1 81 81 1e bf 80 32 18 00 00
00 02 00 00 00 00 00 00 00 00 00 00 00 28 56 77 66
f1 8f c6 01

```

**COMM\_EOP (4 bytes):**

```
ff ff ff ff
```

**4.5.1.2 Downstream Partner Sends Out CMD\_SEND\_STAGE 1**

**COMM\_BOP (4 bytes):**

```
00 00 00 00
```

**COMM\_COMMAND (4 bytes):**



28 2 00 00

**COMM\_TO (50 bytes):**

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7  
40 d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00  
2d 00 54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

**COMM\_FROM (50 bytes):**

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a  
e0 80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00  
2d 00 54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

**COMM\_REPLICA (96 bytes):**

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7  
40 d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00  
4e 00 20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20  
00 56 00 4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00  
53 00 59 00 53 00 56 00 4f 00 4c 00 20 00 53 00 48  
00 41 00 52 00 45 00 29 00 00 00

**COMM\_CXTION (98 bytes):**

10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57  
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00  
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00  
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00  
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00  
30 00 30 00 44 00 00 00

**COMM\_JOIN\_GUID (20 bytes):**

10 00 00 00 ae 1b b7 65 7a c4 40 db c9 f8 bd 7e 32 65  
43 41

**COMM\_LAST\_JOIN\_TIME (8 bytes):**

14 71 d5 42 f1 8f c6 01

**COMM\_BLOCK\_SIZE (8 bytes):**

00 00 00 00 00 00 00 00

**COMM\_FILE\_SIZE (8 bytes):**



```
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00
```

Type \_CHANGE\_ORDER\_RECORD\_\*

```
+0x000 SequenceNumber : 1 0 0 0
+0x004 Flags : c 4 0 1 (0x100040c - CO_FLAG_COMPRESSED_STAGE | CO_FLAG_NEW_FILE
| CO_FLAG_LOCATION_CMD | CO_FLAG_CONTENT_CMD)
+0x008 IFlags : 0 0 0 0 (0x0)
+0x00c State : 6 0 0 0 (Remote changing order staging File fetch has started)
+0x010 ContentCmd : 3 80 0 0 (0x8003 - REASON_BASIC_INFO_CHANGE | REASON_DATA_EXTEND
| REASON_DATA_OVERWRITE)
+0x014 LocationCmd : 0 0 0 0 (0x0)
+0x018 FileAttributes : 20 0 0 0
+0x01c FileVersionNumber : 0 0 0 0
+0x020 PartnerAckSeqNumber : 1a 0 0 0 0 0 0 0
+0x028 FileSize : 0 0 2 0 0 0 0 0
+0x030 FileOffset : 0 0 0 0 0 0 0 0
+0x038 FrsVsn : 1d f8 9 e6 9 90 c6 1
+0x040 FileUsn : e0 b3 d4 0 0 0 0 0
+0x048 JrnlUsn : b0 b0 c9 0 0 0 0 0
+0x050 JrnlFirstUsn : b0 b0 c9 0 0 0 0 0
+0x058 OriginalReplicaNum : 1 0 0 0
+0x05c NewReplicaNum : 1 0 0 0
+0x060 ChangeOrderGuid : 4d c d2 35 d b9 9 4d bd 87 e2 c6 b0 8d 59 21
+0x070 OriginatorGuid : 76 65 78 79 63 b8 da 41 b1 1a 41 63 46 eb be b3
+0x080 FileGuid : de fb 44 7f b7 58 d3 4a 8d 5 2a 3a b ec 17 d0
+0x090 OldParentGuid : bb 31 34 1e 35 5b a5 47 ae e5 55 ba 62 c2 16 d7
+0x0a0 NewParentGuid : bb 31 34 1e 35 5b a5 47 ae e5 55 ba 62 c2 16 d7
+0x0b0 CxtionGuid : f0 a3 f7 8 5c e0 94 49 9a 60 e7 cd 40 57 ec 6f
+0x0c0 AckVersion : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs : 0 0 0 0
+0x0f4 Spare2Wcs : 0 0 0 0
+0x0f8 Extension : 40 50 a1 1
+0x0fc Spare2Bin : 0 0 0 0
+0x100 EventTime : ca 58 93 64 f1 8f c6 1
+0x108 FileNameLength : 16 0
+0x10a FileName : 72 0 65 0 67 0 65 0 64 0 69 0 74 0 2e 0 65 0 78 0 65 0 0 0 74 0
```

COMM\_CO\_EXTENSION\_2 (72 bytes):

```
48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 57 68 ac c0
53 63 0b ac fb b1 81 81 1e bf 80 32 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 5f a4 b4 66 f1 8f c6 01
```

COMM\_EOP (4 bytes):

```
ff ff ff ff
```

### 4.5.1.3 Upstream Partner Sends Out CMD\_RECEIVING\_STAGE 1

COMM\_BOP (4 bytes):

```
00 00 00 00
```

COMM\_COMMAND (4 bytes):

```
38 02 00 00
```

COMM\_TO (50 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

COMM\_FROM (50 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

COMM\_REPLICA (96 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00  
45 00 29 00 00 00
```

COMM\_CXTION (98 bytes):

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57  
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00  
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00  
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00  
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00  
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 ae 1b b7 65 7a c4 40 db c9 f8 bd 7e 32 65  
43 41
```

COMM\_LAST\_JOIN\_TIME (8 bytes):



```

10 00 60 00 04 01 3c 10 14 00 40 a9 00 12 00 01 01 01
18 00 ...
... 10 8d 45 dc 50 00 73 e8 d6 f1 14 ff ff 01 17 d7 00
21 eb 58 8d 00 4d e8 e8 55 07 00 00 6a a1 02 2d e8 e8
78 04 05 2d e8 02 2d 44 52 fc 01 2d e8 e8 66 00 23 eb
80 2a 8d 4d f4 e8 68 02 03 59 50 f4 e8 79 01

```

To interpret the bytes above, Header is dumped as follows:

```

Local var @ 0x3c6fb14 Type _STAGE_HEADER_
+0x000 Major : 0 0 0 0
+0x004 Minor : 3 0 0 0
+0x008 DataHigh : 0 0 0 0
+0x00c DataLow : 0 4 0 0
+0x010 Compression : 0 0 0 0 0 0 0 0
+0x018 Attributes : _FILE_NETWORK_OPEN_INFORMATION
+0x000 CreationTime : 70 f6 90 64 f1 8f c6 1
+0x008 LastAccessTime : ca 58 93 64 f1 8f c6 1
+0x010 LastWriteTime : c0 5a d7 de e6 7a c6 1
+0x018 ChangeTime : a4 7e 5f 66 f1 8f c6 1
+0x020 AllocationSize : 0 0 2 0 0 0 0 0
+0x028 EndOfFile : 0 fc 1 0 0 0 0 0
+0x030 FileAttributes : 20 0 0 0 0 0 0 0
+0x050 ChangeOrderCommand : _CHANGE_ORDER_RECORD_
+0x000 SequenceNumber : 1 0 0 0
+0x004 Flags : 2c 4 0 1
(0x100042c - CO_FLAG_COMPRESSED_STAGE | CO_FLAG_NEW_FILE |
CO_FLAG_LOCALCO | CO_FLAG_LOCATION_CMD | CO_FLAG_CONTENT_CMD)
+0x008 IFlags : 0 0 0 0 (0x0)
+0x00c State : 1 0 0 0
(Allocating staging File space for local Change Order)
+0x010 ContentCmd : 3 80 0 0
(0x8003 - REASON_BASIC_INFO_CHANGE | REASON_DATA_EXTEND |
REASON_DATA_OVERWRITE)
+0x014 LocationCmd : 0 0 0 0 (0x0)
+0x018 FileAttributes : 20 0 0 0
+0x01c FileVersionNumber : 0 0 0 0
+0x020 PartnerAckSeqNumber : 0 0 0 0 0 0 0 0
+0x028 FileSize : 0 0 2 0 0 0 0 0
+0x030 FileOffset : 0 0 0 0 0 0 0 0
+0x038 FrsVsn : 1d f8 9 e6 9 90 c6 1
+0x040 FileUsn : 60 b1 c9 0 0 0 0 0
+0x048 JrnlUsn : b0 b0 c9 0 0 0 0 0
+0x050 JrnlFirstUsn : b0 b0 c9 0 0 0 0 0
+0x058 OriginalReplicaNum : 1 0 0 0
+0x05c NewReplicaNum : 1 0 0 0
+0x060 ChangeOrderGuid : 4d c d2 35 d b9 9 4d
bd 87 e2 c6 b0 8d 59 21
+0x070 OriginatorGuid : 76 65 78 79 63 b8 da 41
b1 1a 41 63 46 eb be b3
+0x080 FileGuid : de fb 44 7f b7 58 d3 4a
8d 5 2a 3a b ec 17 d0
+0x090 OldParentGuid : e1 99 a7 ae 27 b4 22 45
81 59 5e 55 e0 93 ba 24
+0x0a0 NewParentGuid : e1 99 a7 ae 27 b4 22 45
81 59 5e 55 e0 93 ba 24
+0x0b0 CxtionGuid : 5e 1b 8b 23 ea ac 7 4e
be ca e7 ad 76 85 16 ad
+0x0c0 AckVersion : 0 0 0 0 0 0 0 0
+0x0c8 Spare2Ull : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid : 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid : 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs : 0 0 0 0
+0x0f4 Spare2Wcs : 0 0 0 0
+0x0f8 Extension : 0 0 0 0

```

```

+0x0fc Spare2Bin      : 0 0 0 0
+0x100 EventTime     : ca 58 93 64 f1 8f c6 1
+0x108 FileNameLength : 16 0
+0x10a FileName      : 72 0 65 0 67 0 65 0
                   : 64 0 69 0 74 0 2e 0
                   : 65 0 78 0 65 0
+0x368 FileObjId     : _FILE_OBJECTID_BUFFER
+0x000 ObjectId      : de fb 44 7f b7 58 d3 4a
                   : 8d 5 2a 3a b ec 17 d0
+0x010 BirthVolumeId : 0 0 0 0 0 0 0 0
                   : 0 0 0 0 0 0 0 0
+0x020 BirthObjectId : 0 0 0 0 0 0 0 0
                   : 0 0 0 0 0 0 0 0
+0x030 DomainId     : 0 0 0 0 0 0 0 0
                   : 0 0 0 0 0 0 0 0
+0x3a8 CocExt        : _CO_RECORD_EXTENSION_WIN2K_
+0x000 FieldSize     : 0 0 0 0
+0x004 Major         : 0 0
+0x006 OffsetCount   : 0 0
+0x008 Offset        : 0 0 0 0
+0x00c OffsetLast    : 0 0 0 0
+0x010 DataChecksum  : _DATA_EXTENSION_CHECKSUM_
+0x000 Prefix        : 0 0 0 0 0 0 0 0
+0x008 Data          : 0 0 0 0 0 0 0 0
                   : 0 0 0 0 0 0 0 0
+0x3d0 CompressionGuid : d2 f7 d2 64 95 26 6d 43
                   : 88 30 8d 3c 58 70 1e 15
+0x3e0 EncryptedDataHigh : 0 0 0 0
+0x3e4 EncryptedDataLow  : 0 0 0 0
+0x3e8 EncryptedDataSize : 0 0 0 0 0 0 0 0
+0x3f0 ReparseDataPresent : 0 0 0 0
+0x3f4 ReparsePointDataHigh : 0 0 0 0
+0x3f8 ReparsePointDataLow : 0 4 0 0

```

**Real compressed data:**

```

00 00 00 00 cf bb a0 03 00 00 00 02 00 30 a0 00 30 41
05 20 01 00 04 84 14 00 60 24 15 04 80 40 00 1c 01 01
90 00 00 05 ca 20 00 2c 20 00 2c 01 05 00 20 00 3c 02
15 00 0c 57 c2 d9 22 0f 2d 40 42 bc dd 3d e9 5b 01 4e
02 10 00 60 00 04 01 3c 10 14 00 40 a9 00 12 00 01 01
01 18 00 ...
... 10 8d 45 dc 50 00 73 e8 d6 f1 14 ff ff 01 17 d7 00
21 eb 58 8d 00 4d e8 e8 55 07 00 00 6a a1 02 2d e8 e8
78 04 05 2d e8 02 2d 44 52 fc 01 2d e8 e8 66 00 23 eb
80 2a 8d 4d f4 e8 68 02 03 59 50 f4 e8 79 01

```

**COMM\_BLOCK\_SIZE (8 bytes):**

```
00 00 1 00 00 00 00 00
```

**COMM\_FILE\_SIZE (8 bytes):**

```
62 5c 1 00 00 00 00 00
```

**COMM\_FILE\_OFFSET (8 bytes):**

00 00 00 00 00 00 00 00

COMM\_CO\_GUID (20 bytes):

10 00 00 00 4d 0c d2 35 0d b9 09 4d bd 87 e2 c6 b0 8d  
59 21

COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):

00 00 00 00

COMM\_EOP (4 bytes):

ff ff ff ff

#### 4.5.1.4 Downstream Partner Sends Out CMD\_SEND\_STAGE 2

COMM\_BOP (4 bytes):

00 00 00 00

COMM\_COMMAND (4 bytes):

28 2 00 00

COMM\_TO (50 bytes):

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

COMM\_FROM (50 bytes):

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

COMM\_REPLICA (96 bytes):

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00  
45 00 29 00 00 00



COMM\_CXTION (98 bytes):

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 ae 1b b7 65 7a c4 40 db c9 f8 bd 7e 32 65
43 41
```

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
14 71 d5 42 f1 8f c6 01
```

COMM\_BLOCK\_SIZE (8 bytes):

```
00 00 00 00 00 00 00 00
```

COMM\_FILE\_SIZE (8 bytes):

```
62 5c 1 00 00 00 00 00
```

COMM\_FILE\_OFFSET (8 bytes):

```
00 00 1 00 00 00 00 00
```

COMM\_CO\_GUID (20 bytes):

```
10 00 00 00 4d 0c d2 35 0d b9 09 4d bd 87 e2 c6 b0 8d
59 21
```

COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):

```
00 00 00 00
```

COMM\_REMOTE\_CO (796 bytes):

```
18 03 00 00 01 00 00 00 0c 04 00 01 00 00 00 06 00
```



```

+0x058 OriginalReplicaNum : 1 0 0 0
+0x05c NewReplicaNum      : 1 0 0 0
+0x060 ChangeOrderGuid   : 4d c d2 35 d b9 9 4d
                          bd 87 e2 c6 b0 8d 59 21
+0x070 OriginatorGuid    : 76 65 78 79 63 b8 da 41
                          b1 1a 41 63 46 eb be b3
+0x080 FileGuid          : de fb 44 7f b7 58 d3 4a
                          8d 5 2a 3a b ec 17 d0
+0x090 OldParentGuid     : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
+0x0a0 NewParentGuid     : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
+0x0b0 CxtionGuid        : f0 a3 f7 8 5c e0 94 49
                          9a 60 e7 cd 40 57 ec 6f
+0x0c0 AckVersion         : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull         : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid        : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid        : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs         : 0 0 0 0
+0x0f4 Spare2Wcs         : 0 0 0 0
+0x0f8 Extension         : 40 50 a1 1
+0x0fc Spare2Bin         : 0 0 0 0
+0x100 EventTime         : ca 58 93 64 f1 8f c6 1
+0x108 FileNameLength    : 16 0
+0x10a FileName          : 72 0 65 0 67 0 65 0
                          64 0 69 0 74 0 2e 0
                          65 0 78 0 65 0

```

**COMM\_CO\_EXTENSION\_2 (72 bytes):**

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 57 68 ac c0
53 63 0b ac fb b1 81 81 1e bf 80 32 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 5f a4 b4 66 f1 8f c6 01

```

**COMM\_EOP (4 bytes):**

```
ff ff ff ff
```

**4.5.1.5 Upstream Partner Sends Out CMD\_RECEIVING\_STAGE 2**

**COMM\_BOP (4 bytes):**

```
00 00 00 00
```

**COMM\_COMMAND (4 bytes):**

```
38 2 00 00
```

**COMM\_TO (50 bytes):**

```

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

```

COMM\_FROM (50 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

COMM\_REPLICA (96 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 0
```

COMM\_CXTION (98 bytes):

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 ae 1b b7 65 7a c4 40 db c9 f8 bd 7e 32 65
43 41
```

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
14 71 d5 42 f1 8f c6 01
```

COMM\_BLOCK (23,654 bytes):

```
62 5c 00 00 85 15 f4 82 15 1d 22 ff 81 15 f4 e8 6c 80
10 5f 5e 88 c9 c2 c 8 67 5d e9 26 80 e 1 7 6f b8 8 18
00 00 e8 63 00 17 00 00 a1 f0 c1 01 01 00 33 c5 89 45
fc 8b 45 14 00 56 57 8b 7d 08 89 85 f8 20 e7 ff ff be
00 00 45 56 8d a0 85 ...
... 55 10 3c 5c 3c 84 60 5c 9c 3c a8 00 3c b4 3c c0 3c
cc 3c d1 00 3c de 3c f2 3c fe 3c 1c 00 3d 20 3d 3c 3d
40 3d 5c 00 3d 60 3d 68 3d 6c 3d 84 00 3d 88 3d a4 3d
a8 3d 00 10 c0 01 00 24 e0 39 c8 30 cc 00 30 d0 30 d4
30 d8 30 dc 40 30 c4 31 c8 31 cc a0 77 d4 e0 31 ec 31
1c 32 80 03 5f 00 3f 04 ff 3f 04 3f 04 3f 04 3f 04 3f
04 3f 04 3f 04 3f 04 ff 3f 04 3f 04 3f 04 3f 04 3f 04
3f 04 3f 04 3f 04 ff 3f 04 3f 04 3f 04 3f 04 3f 04 3f
04 1f 02 1f 01 ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01
1f 01 1f 01 ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f
01 1f 01 ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01
1f 01 ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f
```

```
01 ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01
ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 ff
1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 ff 1f
01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 15 18 01
07 b4 00 40 38 01 de fb 44 00 7f b7 58 d3 4a 8d 05 2a
e0 3a b ec 17 d0 a8 01 af 00 1f 01 00 00
```

**Real compressed data:**

```
62 5c 00 00 85 15 f4 82 15 1d 22 ff 81 15 f4 e8 6c 80
10 5f 5e 88 c9 c2 0c 08 67 5d e9 26 80 0e 01 07 6f b8
08 18 00 00 e8 63 00 17 00 00 a1 f0 c1 01 01 00 33 c5
89 45 fc 8b 45 14 00 56 57 8b 7d 08 89 85 f8 20 e7 ff
ff be 00 00 45 56 8d a0 85 ...
... 55 10 3c 5c 3c 84 60 5c 9c 3c a8 00 3c b4 3c c0 3c
cc 3c d1 00 3c de 3c f2 3c fe 3c 1c 00 3d 20 3d 3c 3d
40 3d 5c 00 3d 60 3d 68 3d 6c 3d 84 00 3d 88 3d a4 3d
a8 3d 00 10 c0 01 00 24 e0 39 c8 30 cc 00 30 d0 30 d4
30 d8 30 dc 40 30 c4 31 c8 31 cc a0 77 d4 e0 31 ec 31
1c 32 80 03 5f 00 3f 04 ff 3f 04 3f 04 3f 04 3f 04 3f
04 3f 04 3f 04 3f 04 ff 3f 04 3f 04 3f 04 3f 04 3f 04
3f 04 3f 04 3f 04 ff 3f 04 3f 04 3f 04 3f 04 3f 04 3f
04 1f 02 1f 01 ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01
1f 01 1f 01 ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f
01 1f 01 ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01
1f 01 ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f
01 ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01
ff 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 ff
1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 ff 1f
01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 1f 01 15 18 01
07 b4 00 40 38 01 de fb 44 00 7f b7 58 d3 4a 8d 05 2a
e0 3a b ec 17 d0 a8 01 af 00 1f 01 00 00
```

**COMM\_BLOCK\_SIZE (8 bytes):**

```
62 5c 00 00 00 00 00 00
```

**COMM\_FILE\_SIZE (8 bytes):**

```
62 5c 01 00 00 00 00 00
```

**COMM\_FILE\_OFFSET (8 bytes):**

```
00 00 01 00 00 00 00 00
```

**COMM\_CO\_GUID (20 bytes):**

```
10 00 00 00 4d 0c d2 35 0d b9 9 4d bd 87 e2 c6 b0 8d
59 21
```

**COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):**

00 00 00 00

COMM\_EOP (4 bytes):

ff ff ff ff

#### 4.5.1.6 Downstream Partner Sends Out CMD\_REMOTE\_CO\_DONE

COMM\_BOP (4 bytes):

00 00 00 00

COMM\_COMMAND (4 bytes):

50 02 00 00

COMM\_TO (50 bytes):

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

COMM\_FROM (50 bytes):

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

COMM\_REPLICA (96 bytes):

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00  
45 00 29 00 00 00

COMM\_CXTION (98 bytes):

10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57  
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00  
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00  
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00  
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00  
30 00 30 00 44 00 00 00

COMM\_JOIN\_GUID (20 bytes):

10 00 00 00 ae 1b b7 65 7a c4 40 db c9 f8 bd 7e 32 65  
43 41

COMM\_LAST\_JOIN\_TIME (8 bytes):

14 71 d5 42 f1 8f c6 01

COMM\_BLOCK\_SIZE (8 bytes):

00 00 00 00 00 00 00 00

COMM\_FILE\_SIZE (8 bytes):

00 00 00 00 00 00 00 00

COMM\_FILE\_OFFSET (8 bytes):

00 00 00 00 00 00 00 00

COMM\_GVSN (28 bytes):

18 00 00 00 1d f8 09 e6 09 90 c6 01 76 65 78 79 63 b8  
da 41 b1 1a 41 63 46 eb be b3

COMM\_CO\_GUID (20 bytes):

10 00 00 00 4d 0c d2 35 0d b9 09 4d bd 87 e2 c6 b0 8d  
59 21

COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):

1a 00 00 00

COMM\_REMOTE\_CO (796 bytes):

18 03 00 00 01 00 00 00 0e 04 00 01 01 00 00 00 16 00  
00 00 03 80 00 00 00 00 00 00 20 00 00 00 00 00 00  
1a 00 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00  
00 00 00 00 00 00 1d f8 09 e6 09 90 c6 01 20 bf d4 00  
00 00 00 00 b0 b0 c9 00 00 00 00 00 b0 b0 c9 00 00 00





```

+0x060 ChangeOrderGuid : 4d c d2 35 d b9 9 4d
                          bd 87 e2 c6 b0 8d 59 21
+0x070 OriginatorGuid  : 76 65 78 79 63 b8 da 41
                          b1 1a 41 63 46 eb be b3
+0x080 FileGuid        : de fb 44 7f b7 58 d3 4a
                          8d 5 2a 3a b ec 17 d0
+0x090 OldParentGuid   : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
+0x0a0 NewParentGuid   : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
+0x0b0 CxtionGuid      : f0 a3 f7 8 5c e0 94 49
                          9a 60 e7 cd 40 57 ec 6f
+0x0c0 AckVersion      : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull       : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid     : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid     : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs      : 0 0 0 0
+0x0f4 Spare2Wcs      : 0 0 0 0
+0x0f8 Extension      : 40 50 a1 1
+0x0fc Spare2Bin      : 0 0 0 0
+0x100 EventTime      : ca 58 93 64 f1 8f c6 1
+0x108 FileNameLength : 16 0
+0x10a FileName       : 72 0 65 0 67 0 65 0
                          64 0 69 0 74 0 2e 0
                          65 0 78 0 65 0

```

**COMM\_CO\_EXTENSION\_2 (72 bytes):**

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 57 68 ac c0
53 63 0b ac fb b1 81 81 1e bf 80 32 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 5f a4 b4 66 f1 8f c6 01

```

**COMM\_EOP (4 bytes):**

```
ff ff ff ff
```

**4.5.2 Rename a File**

On SHICO-TEMP-1, rename file file.txt to newfile.txt.

**4.5.2.1 Upstream Partner Sends Out CMD\_REMOTE\_CO**

**COMM\_BOP (4 bytes):**

```
00 00 00 00
```

**COMM\_COMMAND (4 bytes):**

```
18 2 00 00
```

**COMM\_TO (50 bytes):**

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
```

```
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

COMM\_FROM (50 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

COMM\_REPLICA (96 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

COMM\_CXTION (98 bytes):

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 a5 75 98 1b fd dc 4f bb f5 c1 c8 4d e9 4e
06 e4
```

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
d0 bd 3f 36 fe 8f c6 1
```

COMM\_REMOTE\_CO (796 bytes):

```
18 03 00 00 23 00 00 00 24 00 00 01 00 00 00 14 00
00 00 00 20 00 00 0e 00 00 20 00 00 01 00 00 00
23 00 00 00 00 00 00 10 00 00 00 00 00 00 00 00
00 00 00 00 00 26 00 e0 ce 33 90 c6 01 58 60 ca 00
00 00 00 58 60 ca 00 00 00 00 58 60 ca 00 00 00
00 00 01 00 00 01 00 00 06 3f 87 f6 85 86 a3 4c
be 65 62 eb 66 a5 4f 3f 76 65 78 79 63 b8 da 41 b1 1a
41 63 46 eb be b3 16 79 07 75 5b c1 cf 43 9e 3b f8 1d
69 1e 0e 5f e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0 80 4c
03 65 f8 d9 3c da 06 4f 88 2c b8 bc e4 77 b8 b2 8e 89
bf 35 b5 89 c6 01 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



```

+0x0c0 AckVersion      : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull      : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid     : 0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid     : 0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs      : 0 0 0 0
+0x0f4 Spare2Wcs      : 0 0 0 0
+0x0f8 Extension      : 50 1d a2 1
+0x0fc Spare2Bin      : 0 0 0 0
+0x100 EventTime      : 5c 9e e9 38 fe 8f c6 1
+0x108 FileNameLength : 16 0
+0x10a FileName       : 6e 0 65 0 77 0 66 0
                        69 0 6c 0 65 0 2e 0
                        74 0 78 0 74 0

```

**COMM\_CO\_EXTENSION\_2 (72 bytes):**

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 d8 da 50 bb
7f 57 64 47 51 af 71 68 6c 89 79 b2 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 06 d7 c8 3a fe 8f c6 01

```

**COMM\_EOP (4 bytes):**

```
ff ff ff ff
```

**4.5.2.2 Downstream Partner Sends Out CMD\_SEND\_STAGE**

**COMM\_BOP (4 bytes):**

```
00 00 00 00
```

**COMM\_COMMAND (4 bytes):**

```
28 2 00 0
```

**COMM\_TO (50 bytes):**

```

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

```

**COMM\_FROM (50 bytes):**

```

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

```

**COMM\_REPLICA (96 bytes):**

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

**COMM\_CXTION (98 bytes):**

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 0
```

**COMM\_JOIN\_GUID (20 bytes):**

```
10 00 00 00 a5 75 98 1b fd dc 4f bb f5 c1 c8 4d e9 4e
06 e4
```

**COMM\_LAST\_JOIN\_TIME (8 bytes):**

```
d0 bd 3f 36 fe 8f c6 01
```

**COMM\_BLOCK\_SIZE (8 bytes):**

```
00 00 00 00 00 00 00 00
```

**COMM\_FILE\_SIZE (8 bytes):**

```
00 00 00 00 00 00 00 00
```

**COMM\_FILE\_OFFSET (8 bytes):**

```
00 00 00 00 00 00 00 00
```

**COMM\_CO\_GUID (20 bytes):**

```
10 00 00 00 06 3f 87 f6 85 86 a3 4c be 65 62 eb 66 a5
4f 3f
```

**COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):**



```

+0x01c FileVersionNumber : 1 0 0 0
+0x020 PartnerAckSeqNumber : 23 0 0 0 0 0 0 0
+0x028 FileSize : 10 0 0 0 0 0 0 0
+0x030 FileOffset : 0 0 0 0 0 0 0 0
+0x038 FrsVsn : 26 0 e0 ce 33 90 c6 1
+0x040 FileUsn : 50 5a d5 0 0 0 0 0
+0x048 JrnlUsn : 58 60 ca 0 0 0 0 0
+0x050 JrnlFirstUsn : 58 60 ca 0 0 0 0 0
+0x058 OriginalReplicaNum : 1 0 0 0
+0x05c NewReplicaNum : 1 0 0 0
+0x060 ChangeOrderGuid : 6 3f 87 f6 85 86 a3 4c
                        be 65 62 eb 66 a5 4f 3f
+0x070 OriginatorGuid : 76 65 78 79 63 b8 da 41
                        b1 1a 41 63 46 eb be b3
+0x080 FileGuid : 16 79 7 75 5b c1 cf 43
                  9e 3b f8 1d 69 1e e 5f
+0x090 OldParentGuid : bb 31 34 1e 35 5b a5 47
                      ae e5 55 ba 62 c2 16 d7
+0x0a0 NewParentGuid : bb 31 34 1e 35 5b a5 47
                      ae e5 55 ba 62 c2 16 d7
+0x0b0 CxtionGuid : f0 a3 f7 8 5c e0 94 49
                   9a 60 e7 cd 40 57 ec 6f
+0x0c0 AckVersion : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs : 0 0 0 0
+0x0f4 Spare2Wcs : 0 0 0 0
+0x0f8 Extension : a0 45 a1 1
+0x0fc Spare2Bin : 0 0 0 0
+0x100 EventTime : 5c 9e e9 38 fe 8f c6 1
+0x108 FileNameLength : 16 0
+0x10a FileName : 6e 0 65 0 77 0 66 0
                  69 0 6c 0 65 0 2e 0
                  74 0 78 0 74 0

```

**COMM\_CO\_EXTENSION\_2 (72 bytes):**

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 d8 da 50 bb
7f 57 64 47 51 af 71 68 6c 89 79 b2 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 eb 85 02 3b fe 8f c6 01

```

**COMM\_EOP (4 bytes):**

```
ff ff ff ff
```

**4.5.2.3 Upstream Partner Sends Out CMD\_RECEIVING\_STAGE**

**COMM\_BOP (4 bytes):**

```
00 00 00 00
```

**COMM\_COMMAND (4 bytes):**

```
38 02 00 00
```

COMM\_TO (50 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

COMM\_FROM (50 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

COMM\_REPLICA (96 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 0
```

COMM\_CXTION (98 bytes):

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 a5 75 98 1b fd dc 4f bb f5 c1 c8 4d e9 4e
06 e4
```

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
d0 bd 3f 36 fe 8f c6 1
```

COMM\_BLOCK (1,206 bytes):

```
b2 04 00 00 00 00 00 00 03 00 00 00 00 00 00 00 04
00 00 00 00 00 00 00 00 00 00 00 fa 46 f4 14 fe 8f c6 01
3e 5a 53 18 fe 8f c6 01 fc aa 7d 16 fe 8f c6 01 5c 9e
e9 38 fe 8f c6 01 10 00 00 00 00 00 00 00 0a 00 00 00
00 00 00 00 20 00 00 00 00 00 00 00 01 00 00 00 24 00
00 01 00 00 00 00 01 00 00 00 00 20 00 00 0e 00 00 00
20 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 10 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 26 00 e0 ce
```





```

+0x010 Compression      : 0 0 0 0 0 0 0 0
+0x018 Attributes      : _FILE_NETWORK_OPEN_INFORMATION
+0x000 CreationTime    : fa 46 f4 14 fe 8f c6 1
+0x008 LastAccessTime  : 3e 5a 53 18 fe 8f c6 1
+0x010 LastWriteTime   : fc aa 7d 16 fe 8f c6 1
+0x018 ChangeTime      : 5c 9e e9 38 fe 8f c6 1
+0x020 AllocationSize  : 10 0 0 0 0 0 0 0
+0x028 EndOfFile       : a 0 0 0 0 0 0 0
+0x030 FileAttributes  : 20 0 0 0 0 0 0 0
+0x050 ChangeOrderCommand : _CHANGE_ORDER_RECORD_
+0x000 SequenceNumber  : 1 0 0 0
+0x004 Flags           : 24 0 0 1
(0x1000024 - CO_FLAG_COMPRESSED_STAGE | CO_FLAG_LOCALCO |
CO_FLAG_CONTENT_CMD)
+0x008 IFlags          : 0 0 0 0 (0x0)
+0x00c State           : 1 0 0 0
(Allocating staging File space for local Change Order)
+0x010 ContentCmd      : 0 20 0 0
(0x2000 - REASON_RENAME_NEW_NAME)
+0x014 LocationCmd    : e 0 0 0 (0xe)
+0x018 FileAttributes  : 20 0 0 0
+0x01c FileVersionNumber : 1 0 0 0
+0x020 PartnerAckSeqNumber : 0 0 0 0 0 0 0 0
+0x028 FileSize       : 10 0 0 0 0 0 0 0
+0x030 FileOffset     : 0 0 0 0 0 0 0 0
+0x038 FrsVsn         : 26 0 e0 ce 33 90 c6 1
+0x040 FileUsn        : 58 60 ca 0 0 0 0 0
+0x048 JrnlUsn        : 58 60 ca 0 0 0 0 0
+0x050 JrnlFirstUsn   : 58 60 ca 0 0 0 0 0
+0x058 OriginalReplicaNum : 1 0 0 0
+0x05c NewReplicaNum  : 1 0 0 0
+0x060 ChangeOrderGuid : 6 3f 87 f6 85 86 a3 4c
                        be 65 62 eb 66 a5 4f 3f
+0x070 OriginatorGuid : 76 65 78 79 63 b8 da 41
                        b1 1a 41 63 46 eb be b3
+0x080 FileGuid       : 16 79 7 75 5b c1 cf 43
                        9e 3b f8 1d 69 1e e 5f
+0x090 OldParentGuid  : e1 99 a7 ae 27 b4 22 45
                        81 59 5e 55 e0 93 ba 24
+0x0a0 NewParentGuid  : e1 99 a7 ae 27 b4 22 45
                        81 59 5e 55 e0 93 ba 24
+0x0b0 CxtionGuid     : 3 65 f8 d9 3c da 6 4f
                        88 2c b8 bc e4 77 b8 b2
+0x0c0 AckVersion     : 0 0 0 0 0 0 0 0
+0x0c8 Spare2Ull      : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid     : 0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid     : 0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs      : 0 0 0 0
+0x0f4 Spare2Wcs      : 0 0 0 0
+0x0f8 Extension      : 0 0 0 0
+0x0fc Spare2Bin      : 0 0 0 0
+0x100 EventTime      : 5c 9e e9 38 fe 8f c6 1
+0x108 FileNameLength : 16 0
+0x10a FileName       : 6e 0 65 0 77 0 66 0
                        69 0 6c 0 65 0 2e 0
                        74 0 78 0 74 0
+0x368 FileObjId     : _FILE_OBJECTID_BUFFER
+0x000 ObjectId       : 16 79 7 75 5b c1 cf 43
                        9e 3b f8 1d 69 1e e 5f
+0x010 BirthVolumeId  : 0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0
+0x020 BirthObjectId  : 0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0
+0x030 DomainId       : 0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0
+0x3a8 CocExt         : _CO_RECORD_EXTENSION_WIN2K_
+0x000 FieldSize      : 0 0 0 0
+0x004 Major          : 0 0

```

```
+0x006 OffsetCount      : 0 0
+0x008 Offset          : 0 0 0 0
+0x00c OffsetLast      : 0 0 0 0
+0x010 DataChecksum    : _DATA_EXTENSION_CHECKSUM_
    +0x000 Prefix       : 0 0 0 0 0 0 0 0
    +0x008 Data         : 0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0
+0x3d0 CompressionGuid : d2 f7 d2 64 95 26 6d 43
                        88 30 8d 3c 58 70 1e 15
+0x3e0 EncryptedDataHigh : 0 0 0 0
+0x3e4 EncryptedDataLow  : 0 0 0 0
+0x3e8 EncryptedDataSize : 0 0 0 0 0 0 0 0
+0x3f0 ReparseDataPresent : 0 0 0 0
+0x3f4 ReparsePointDataHigh : 0 0 0 0
+0x3f8 ReparsePointDataLow : 0 4 0 0
```

#### Compressed data:

```
00 00 00 00 af b0 a0 03 00 00 00 02 00 30 a0 00 30 41
05 20 01 00 04 84 14 00 60 24 15 04 80 40 00 1c 01 01
90 00 00 05 ca 20 00 2c 20 00 2c 01 05 00 20 00 3c 02
15 00 0c 57 c2 d9 22 0f 2d 40 42 bc dd 3d e9 5b 01 4e
02 10 00 60 00 04 01 3c 10 14 00 40 a9 00 12 00 01 01
01 18 00 b4 05 0b 02 26 18 03 26 08 9e 25 01 0b 80 10
18 00 ff 01 1f 00 01 3f bb 00 26 06 67 00 00 43 02 17
04 43 12 00 1c 31 03 0a 00 00 0a 00 0b 05 02 66 69 00
6c 65 2e 74 78 74 0d 0a 0e 07 04 12 01 b5 05 1d 16 79
07 75 00 5b c1 cf 43 9e 3b f8 1d 30 69 1e 0e 5f 05 17
25 07
```

#### COMM\_BLOCK\_SIZE (8 bytes):

```
b2 04 00 00 00 00 00 00
```

#### COMM\_FILE\_SIZE (8 bytes):

```
b2 04 00 00 00 00 00 00
```

#### COMM\_FILE\_OFFSET (8 bytes):

```
00 00 00 00 00 00 00 00
```

#### COMM\_CO\_GUID (20 bytes):

```
10 00 00 00 06 3f 87 f6 85 86 a3 4c be 65 62 eb 66 a5
4f 3f
```

#### COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):

```
00 00 00 00
```

COMM\_EOP (4 bytes):

```
ff ff ff ff
```

#### 4.5.2.4 Downstream Partner Sends Out CMD\_REMOTE\_CO\_DONE

COMM\_BOP (4 bytes):

```
00 00 00 00
```

COMM\_COMMAND (4 bytes):

```
50 2 00 00
```

COMM\_TO (50 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

COMM\_FROM (50 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

COMM\_REPLICA (96 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00  
45 00 29 00 00 00
```

COMM\_CXTION (98 bytes):

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57  
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00  
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00  
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00  
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00  
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 a5 75 98 1b fd dc 4f bb f5 c1 c8 4d e9 4e
```

06 e4

COMM\_LAST\_JOIN\_TIME (8 bytes):

d0 bd 3f 36 fe 8f c6 01

COMM\_BLOCK\_SIZE (8 bytes):

00 00 00 00 00 00 00 00

COMM\_FILE\_SIZE (8 bytes):

00 00 00 00 00 00 00 00

COMM\_FILE\_OFFSET (8 bytes):

00 00 00 00 00 00 00 00

COMM\_GVSN (28 bytes):

18 00 00 00 26 00 e0 ce 33 90 c6 01 76 65 78 79 63 b8  
da 41 b1 1a 41 63 46 eb be b3

COMM\_CO\_GUID (20 bytes):

10 00 00 00 06 3f 87 f6 85 86 a3 4c be 65 62 eb 66 a5  
4f 3f

COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):

23 00 00 00

COMM\_REMOTE\_CO (796 bytes):

18 03 00 00 01 00 00 00 06 00 00 01 01 00 00 00 16 00  
00 00 00 20 00 00 0e 00 00 00 20 00 00 00 01 00 00 00  
23 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 26 00 e0 ce 33 90 c6 01 20 76 d5 00  
00 00 00 00 58 60 ca 00 00 00 00 00 58 60 ca 00 00 00  
00 00 01 00 00 00 01 00 00 00 06 3f 87 f6 85 86 a3 4c  
be 65 62 eb 66 a5 4f 3f 76 65 78 79 63 b8 da 41 b1 1a  
41 63 46 eb be b3 16 79 07 75 5b c1 cf 43 9e 3b f8 1d  
69 1e 0e 5f bb 31 34 1e 35 5b a5 47 ae e5 55 ba 62 c2  
16 d7 bb 31 34 1e 35 5b a5 47 ae e5 55 ba 62 c2 16 d7



```

ae e5 55 ba 62 c2 16 d7
+0x0a0 NewParentGuid : bb 31 34 1e 35 5b a5 47
ae e5 55 ba 62 c2 16 d7
+0x0b0 CxtionGuid : f0 a3 f7 8 5c e0 94 49
9a 60 e7 cd 40 57 ec 6f
+0x0c0 AckVersion : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs : 0 0 0 0
+0x0f4 Spare2Wcs : 0 0 0 0
+0x0f8 Extension : a0 45 a1 1
+0x0fc Spare2Bin : 0 0 0 0
+0x100 EventTime : 5c 9e e9 38 fe 8f c6 1
+0x108 FileNameLength : 16 0
+0x10a FileName : 6e 0 65 0 77 0 66 0
69 0 6c 0 65 0 2e 0
74 0 78 0 74 0

```

**COMM\_CO\_EXTENSION\_2 (72 bytes):**

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 d8 da 50 bb
7f 57 64 47 51 af 71 68 6c 89 79 b2 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 eb 85 02 3b fe 8f c6 01

```

**COMM\_EOP (4 bytes):**

```
ff ff ff ff
```

**4.5.3 Remove a File**

**4.5.3.1 Upstream Partner Sends Out CMD\_REMOTE\_CO**

**COMM\_BOP (4 bytes):**

```
00 00 00 00
```

**COMM\_COMMAND (4 bytes):**

```
18 02 00 00
```

**COMM\_TO (50 bytes):**

```

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

```

**COMM\_FROM (50 bytes):**

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
```

d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

COMM\_REPLICA (96 bytes):

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00  
45 00 29 00 00 00

COMM\_CXTION (98 bytes):

10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57  
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00  
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00  
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00  
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00  
30 00 30 00 44 00 00 00

COMM\_JOIN\_GUID (20 bytes):

10 00 00 00 3d 83 01 59 fa fb 15 5d 1c df b4 23 53 29  
6b 85

COMM\_LAST\_JOIN\_TIME (8 bytes):

ec 57 9a 14 f1 8f c6 01

COMM\_REMOTE\_CO (796 bytes):

18 03 00 00 19 00 00 00 28 00 00 00 00 00 00 00 14 00  
00 00 00 00 00 00 02 00 00 00 20 00 00 00 01 00 00 00  
19 00 00 00 00 00 00 00 10 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 1c 90 45 84 01 90 c6 01 30 9b c9 00  
00 00 00 00 30 9b c9 00 00 00 00 00 30 9b c9 00 00 00  
00 00 01 00 00 00 01 00 00 00 5d 5a fd e5 03 cb 00 4d  
90 f5 ba 52 9e 85 11 f7 76 65 78 79 63 b8 da 41 b1 1a  
41 63 46 eb be b3 8d 4d 34 5e 48 ef 3a 4d a2 4f b9 70  
d0 29 b0 64 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0 80 4c  
42 30 cd 7e 53 07 85 48 9c db c2 02 4f de 6c e4 8e 89  
bf 35 b5 89 c6 01 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
68 1d e8 00 00 00 00 00 6c 48 95 17 f1 8f c6 01 1a 00  
73 00 6d 00 61 00 6c 00 6c 00 66 00 69 00 6c 00 65 00  
2e 00 74 00 78 00 74 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00





```
+0x10a FileName          : 73 0 6d 0 61 0 6c 0
                          6c 0 66 0 69 0 6c 0
                          65 0 2e 0 74 0 78 0
                          74 0
```

**COMM\_CO\_EXTENSION\_2 (72 bytes):**

```
48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 2e 34 a1 17 f1 8f c6 01
```

**COMM\_EOP (4 bytes):**

```
ff ff ff ff
```

### 4.5.3.2 Downstream Partner Sends Out CMD\_REMOTE\_CO\_DONE

**COMM\_BOP (4 bytes):**

```
00 00 00 00
```

**COMM\_COMMAND (4 bytes):**

```
50 2 00 00
```

**COMM\_TO (50 bytes):**

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

**COMM\_FROM (50 bytes):**

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

**COMM\_REPLICA (96 bytes):**

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

COMM\_CXTION (98 bytes):

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 3d 83 01 59 fa fb 15 5d 1c df b4 23 53 29
6b 85
```

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
ec 57 9a 14 f1 8f c6 01
```

COMM\_BLOCK\_SIZE (8 bytes):

```
00 00 00 00 00 00 00 00
```

COMM\_FILE\_SIZE (8 bytes):

```
00 00 00 00 00 00 00 00
```

COMM\_FILE\_OFFSET (8 bytes):

```
00 00 00 00 00 00 00 00
```

COMM\_GVSN (28 bytes):

```
18 00 00 00 1c 90 45 84 01 90 c6 01 76 65 78 79 63 b8
da 41 b1 1a 41 63 46 eb be b3
```

COMM\_CO\_GUID (20 bytes):

```
10 00 00 00 5d 5a fd e5 03 cb 00 4d 90 f5 ba 52 9e 85 11 f7
```

COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):

```
19 00 00 00
```



```

+0x040 FileUsn      : e0 7d d4 0 0 0 0 0
+0x048 JrnlUsn     : 30 9b c9 0 0 0 0 0
+0x050 JrnlFirstUsn : 30 9b c9 0 0 0 0 0
+0x058 OriginalReplicaNum : 1 0 0 0
+0x05c NewReplicaNum : 1 0 0 0
+0x060 ChangeOrderGuid : 5d 5a fd e5 3 cb 0 4d
                          90 f5 ba 52 9e 85 11 f7
+0x070 OriginatorGuid : 76 65 78 79 63 b8 da 41
                          b1 1a 41 63 46 eb be b3
+0x080 FileGuid    : 8d 4d 34 5e 48 ef 3a 4d
                          a2 4f b9 70 d0 29 b0 64
+0x090 OldParentGuid : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
+0x0a0 NewParentGuid : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
+0x0b0 CxtionGuid   : f0 a3 f7 8 5c e0 94 49
                          9a 60 e7 cd 40 57 ec 6f
+0x0c0 AckVersion   : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull    : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid   : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid   : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs    : 0 0 0 0
+0x0f4 Spare2Wcs    : 0 0 0 0
+0x0f8 Extension    : 60 55 a1 1
+0x0fc Spare2Bin    : 0 0 0 0
+0x100 EventTime    : 6c 48 95 17 f1 8f c6 1
+0x108 FileNameLength : 1a 0
+0x10a FileName     : 73 0 6d 0 61 0 6c 0
                          6c 0 66 0 69 0 6c 0
                          65 0 2e 0 74 0 78 0
                          74 0

```

**COMM\_CO\_EXTENSION\_2 (72 bytes):**

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 00 e7 dd db 17 f1 8f c6 01

```

**COMM\_EOP (4 bytes):**

```
ff ff ff ff
```

**4.5.4 Copy an Empty Folder**

**4.5.4.1 Upstream Partner Sends Out CMD\_REMOTE\_CO**

**COMM\_BOP (4 bytes):**

```
00 00 00 00
```

**COMM\_COMMAND (4 bytes):**

```
18 2 00 00
```

**COMM\_FROM (50 bytes):**

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

COMM\_REPLICA (96 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

COMM\_CXTION (98 bytes):

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 9f b1 95 83 0a d4 fc 23 d1 08 1d 2a b5 f4
65 39
```

COMM\_LAST\_JOIN\_TIME (8 bytes):

```
06 74 70 89 f2 8f c6 01
```

COMM\_REMOTE\_CO (796 bytes):

```
18 03 00 00 1d 00 00 00 28 00 00 01 00 00 00 00 14 00
00 00 00 00 00 00 01 00 00 00 10 00 00 00 00 00 00 00
1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 20 60 ce 47 12 90 c6 01 58 d6 c9 00
00 00 00 00 80 d5 c9 00 00 00 00 00 80 d5 c9 00 00 00
00 00 01 00 00 00 01 00 00 00 e1 44 39 86 52 b6 5b 4b
bf 40 17 82 9d 17 4e 55 76 65 78 79 63 b8 da 41 b1 1a
41 63 46 eb be b3 d8 70 3b df 92 9e 42 4e a8 9b ee b7
73 9a f1 4a e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0 80 4c
22 99 12 28 0f 2e 2e 43 87 21 1e 45 15 7a cf 14 8e 89
bf 35 b5 89 c6 01 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
f8 1d e8 00 00 00 00 00 0a 96 e5 8e f2 8f c6 01 0a 00
53 00 74 00 6f 00 6e 00 65 00 00 00 74 00 2e 00 74 00
78 00 74 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```



```
+0x0f8 Extension      : f8 1d e8 0
+0x0fc Spare2Bin     : 0 0 0 0
+0x100 EventTime     : a 96 e5 8e f2 8f c6 1
+0x108 FileNameLength : a 0
+0x10a FileName      : 53 0 74 0 6f 0 6e 0
                   : 65 0 0 0 74 0 2e 0
                   : 74 0 78 0 74 0
```

**COMM\_CO\_EXTENSION\_2 (72 bytes):**

```
48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 d4 6c 7a 42
f0 06 23 4a 34 42 0f 6e d4 5d 42 01 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 68 93 c9 90 f2 8f c6 01
```

**COMM\_EOP (4 bytes):**

```
ff ff ff ff
```

#### 4.5.4.2 Downstream Partner Sends Out CMD\_SEND\_STAGE

**COMM\_BOP (4 bytes):**

```
00 00 00 00
```

**COMM\_COMMAND (4 bytes):**

```
28 2 00 00
```

**COMM\_TO (50 bytes):**

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

**COMM\_FROM (50 bytes):**

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

**COMM\_REPLICA (96 bytes):**

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
```



45 00 29 00 00 00

**COMM\_CXTION (98 bytes):**

10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57  
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00  
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00  
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00  
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00  
30 00 30 00 44 00 00 00

**COMM\_JOIN\_GUID (20 bytes):**

10 00 00 00 9f b1 95 83 0a d4 fc 23 d1 8 1d 2a b5 f4  
65 39

**COMM\_LAST\_JOIN\_TIME (8 bytes):**

06 74 70 89 f2 8f c6 01

**COMM\_BLOCK\_SIZE (8 bytes):**

00 00 00 00 00 00 00 00

**COMM\_FILE\_SIZE (8 bytes):**

00 00 00 00 00 00 00 00

**COMM\_FILE\_OFFSET (8 bytes):**

00 00 00 00 00 00 00 00

**COMM\_CO\_GUID (20 bytes):**

10 00 00 00 e1 44 39 86 52 b6 5b 4b bf 40 17 82 9d 17  
4e 55

**COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):**

00 00 00 00

**COMM\_REMOTE\_CO (796 bytes):**



```

+0x050 JrnlFirstUsn      : 80 d5 c9 0 0 0 0 0
+0x058 OriginalReplicaNum : 1 0 0 0
+0x05c NewReplicaNum    : 1 0 0 0
+0x060 ChangeOrderGuid  : e1 44 39 86 52 b6 5b 4b
                          bf 40 17 82 9d 17 4e 55
+0x070 OriginatorGuid   : 76 65 78 79 63 b8 da 41
                          b1 1a 41 63 46 eb be b3
+0x080 FileGuid         : d8 70 3b df 92 9e 42 4e
                          a8 9b ee b7 73 9a f1 4a
+0x090 OldParentGuid    : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
+0x0a0 NewParentGuid    : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
+0x0b0 CxtionGuid       : f0 a3 f7 8 5c e0 94 49
                          9a 60 e7 cd 40 57 ec 6f
+0x0c0 AckVersion       : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull        : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid       : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid       : 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs        : 0 0 0 0
+0x0f4 Spare2Wcs        : 0 0 0 0
+0x0f8 Extension        : a0 45 a1 1
+0x0fc Spare2Bin        : 0 0 0 0
+0x100 EventTime        : a 96 e5 8e f2 8f c6 1
+0x108 FileNameLength   : a 0
+0x10a FileName         : 53 0 74 0 6f 0 6e 0
                          65 0 0 0 74 0 2e 0
                          74 0 78 0 74 0

```

COMM\_CO\_EXTENSION\_2 (72 bytes):

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 d4 6c 7a 42
f0 06 23 4a 34 42 0f 6e d4 5d 42 01 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 7b a0 0a 91 f2 8f c6 01

```

COMM\_EOP (4 bytes):

```
ff ff ff ff
```

#### 4.5.4.3 Upstream Partner Sends Out CMD\_RECEIVING\_STAGE

COMM\_BOP (4 bytes):

```
00 00 00 00
```

COMM\_COMMAND (4 bytes):

```
38 02 00 00
```

COMM\_TO (50 bytes):

```

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00

```

54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

**COMM\_FROM (50 bytes):**

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

**COMM\_REPLICA (96 bytes):**

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00  
45 00 29 00 00 00

**COMM\_CXTION (98 bytes):**

10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57  
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00  
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00  
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00  
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00  
30 00 30 00 44 00 00 00

**COMM\_JOIN\_GUID (20 bytes):**

10 00 00 00 9f b1 95 83 0a d4 fc 23 d1 08 1d 2a b5 f4  
65 39

**COMM\_LAST\_JOIN\_TIME (8 bytes):**

6 74 70 89 f2 8f c6 1

**COMM\_BLOCK (1,223 bytes):**

c3 04 00 00 00 00 00 00 03 00 00 00 00 00 00 00 04  
00 00 00 00 00 00 00 00 00 00 0a 96 e5 8e f2 8f c6 01  
0a 96 e5 8e f2 8f c6 01 0a 96 e5 8e f2 8f c6 01 8a 59  
af 90 f2 8f c6 01 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 10 00 00 00 00 00 00 00 01 00 00 00 28 04  
00 01 00 00 00 00 01 00 00 00 00 00 00 01 00 00 00  
10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 20 60 ce 47  
12 90 c6 01 58 d6 c9 00 00 00 00 00 80 d5 c9 00 00 00  
00 00 80 d5 c9 00 00 00 00 00 01 00 00 00 01 00 00 00  
e1 44 39 86 52 b6 5b 4b bf 40 17 82 9d 17 4e 55 76 65  
78 79 63 b8 da 41 b1 1a 41 63 46 eb be b3 d8 70 3b df  
92 9e 42 4e a8 9b ee b7 73 9a f1 4a e1 99 a7 ae 27 b4  
22 45 81 59 5e 55 e0 93 ba 24 e1 99 a7 ae 27 b4 22 45



```

+0x018 ChangeTime      : 8a 59 af 90 f2 8f c6 1
+0x020 AllocationSize  : 0 0 0 0 0 0 0 0
+0x028 EndOfFile      : 0 0 0 0 0 0 0 0
+0x030 FileAttributes  : 10 0 0 0 0 0 0 0
+0x050 ChangeOrderCommand : _CHANGE_ORDER_RECORD_
+0x000 SequenceNumber  : 1 0 0 0
+0x004 Flags           : 28 4 0 1
(0x1000428 - CO_FLAG_COMPRESSED_STAGE | CO_FLAG_NEW_FILE |
CO_FLAG_LOCALCO | CO_FLAG_LOCATION_CMD)
+0x008 IFlags          : 0 0 0 0 (0x0)
+0x00c State           : 1 0 0 0
(Allocating staging File space for local Change Order)
+0x010 ContentCmd      : 0 0 0 0 (0x0)
+0x014 LocationCmd     : 1 0 0 0 (0x1)
+0x018 FileAttributes  : 10 0 0 0
+0x01c FileVersionNumber : 0 0 0 0
+0x020 PartnerAckSeqNumber : 0 0 0 0 0 0 0 0
+0x028 FileSize       : 0 0 0 0 0 0 0 0
+0x030 FileOffset     : 0 0 0 0 0 0 0 0
+0x038 FrsVsn        : 20 60 ce 47 12 90 c6 1
+0x040 FileUsn        : 58 d6 c9 0 0 0 0 0
+0x048 JrnlUsn        : 80 d5 c9 0 0 0 0 0
+0x050 JrnlFirstUsn   : 80 d5 c9 0 0 0 0 0
+0x058 OriginalReplicaNum : 1 0 0 0
+0x05c NewReplicaNum  : 1 0 0 0
+0x060 ChangeOrderGuid : e1 44 39 86 52 b6 5b 4b
                          bf 40 17 82 9d 17 4e 55
+0x070 OriginatorGuid : 76 65 78 79 63 b8 da 41
                          b1 1a 41 63 46 eb be b3
+0x080 FileGuid       : d8 70 3b df 92 9e 42 4e
                          a8 9b ee b7 73 9a f1 4a
+0x090 OldParentGuid  : e1 99 a7 ae 27 b4 22 45
                          81 59 5e 55 e0 93 ba 24
+0x0a0 NewParentGuid  : e1 99 a7 ae 27 b4 22 45
                          81 59 5e 55 e0 93 ba 24
+0x0b0 CxtionGuid     : 22 99 12 28 f 2e 2e 43
                          87 21 1e 45 15 7a cf 14
+0x0c0 AckVersion     : 0 0 0 0 0 0 0 0
+0x0c8 Spare2Ull      : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid     : 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid     : 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs      : 0 0 0 0
+0x0f4 Spare2Wcs      : 0 0 0 0
+0x0f8 Extension      : 0 0 0 0
+0x0fc Spare2Bin      : 0 0 0 0
+0x100 EventTime      : a 96 e5 8e f2 8f c6 1
+0x108 FileNameLength : a 0
+0x10a FileName       : 53 0 74 0 6f 0 6e 0 65 0
+0x368 FileObjId      : _FILE_OBJECTID_BUFFER
+0x000 ObjectId       : d8 70 3b df 92 9e 42 4e
                          a8 9b ee b7 73 9a f1 4a
+0x010 BirthVolumeId  : 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0
+0x020 BirthObjectId  : 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0
+0x030 DomainId      : 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0
+0x3a8 CocExt         : _CO_RECORD_EXTENSION_WIN2K_
+0x000 FieldSize      : 0 0 0 0
+0x004 Major          : 0 0
+0x006 OffsetCount    : 0 0
+0x008 Offset         : 0 0 0 0
+0x00c OffsetLast     : 0 0 0 0
+0x010 DataChecksum   : _DATA_EXTENSION_CHECKSUM_
+0x000 Prefix         : 0 0 0 0 0 0 0 0
+0x008 Data           : 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0
+0x3d0 CompressionGuid : d2 f7 d2 64 95 26 6d 43

```

```
88 30 8d 3c 58 70 1e 15
(64d2f7d2-2695-436d-88308d3c58701e15)
+0x3e0 EncryptedDataHigh : 0 0 0 0 (0)
+0x3e4 EncryptedDataLow : 0 0 0 0 (0)
+0x3e8 EncryptedDataSize : 0 0 0 0 0 0 (0)
+0x3f0 ReparseDataPresent : 0 0 0 0 (0)
+0x3f4 ReparsePointDataHigh : 0 0 0 0 (0)
+0x3f8 ReparsePointDataLow : 0 4 0 0 (0x400)
```

**Real compressed data:**

```
00 00 00 00 c0 b0 20 03 00 00 00 02 00 30 0c 01 83 00
40 04 20 01 00 04 84 14 00 58 2a 24 04 78 40 00 1c 01
01 90 00 00 94 05 20 00 2c 20 00 2c 01 05 00 20 05 00
3c 15 00 0c 57 c2 d9 22 0f 80 2d 42 bc dd 3d e9 5b 01
4e 20 02 00 cc 00 09 01 3c 10 14 40 00 a9 00 12 00 01
03 b6 05 4a 0b 01 26 1b 01 b2 00 a0 0a 26 10 96 18 03
27 08 63 25 01 0b 1b 18 01 31 3e a0 01 53 00 08 02 7b
02 17 00 2f ff 01 ac 1f 00 09 17 01 93 00 03 2f 10 0e
17 b7 00 87 02 2f 04 87 12 01 55 03 87 10 05 87 f3 90
09 01 8d 00 07 81 15 80 01 81 81 04 05 00 00 d8 70 3b
df 92 9e 42 00 4e a8 9b ee b7 73 9a f1 06 4a 85 0b a5
03
```

**COMM\_BLOCK\_SIZE (8 bytes):**

```
c3 04 00 00 00 00 00 00
```

**COMM\_FILE\_SIZE (8 bytes):**

```
c3 04 00 00 00 00 00 00
```

**COMM\_FILE\_OFFSET (8 bytes):**

```
00 00 00 00 00 00 00 00
```

**COMM\_CO\_GUID (20 bytes):**

```
10 00 00 00 e1 44 39 86 52 b6 5b 4b bf 40 17 82 9d 17
4e 55
```

**COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):**

```
00 00 00 0
```

**COMM\_EOP (4 bytes):**

```
ff ff ff ff
```

#### 4.5.4.4 Downstream Partner Sends Out CMD\_REMOTE\_CO\_DONE

COMM\_BOP (4 bytes):

```
00 00 00 0
```

COMM\_COMMAND (4 bytes):

```
50 2 00 0
```

COMM\_TO (50 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

COMM\_FROM (50 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

COMM\_REPLICA (96 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 0
```

COMM\_CXTION (98 bytes):

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes):

```
10 00 00 00 9f b1 95 83 0a d4 fc 23 d1 08 1d 2a b5 f4
65 39
```

COMM\_LAST\_JOIN\_TIME (8 bytes):



06 74 70 89 f2 8f c6 01

COMM\_BLOCK\_SIZE (8 bytes):

00 00 00 00 00 00 00 00

COMM\_FILE\_SIZE (8 bytes):

00 00 00 00 00 00 00 00

COMM\_FILE\_OFFSET (8 bytes):

00 00 00 00 00 00 00 00

COMM\_GVSN (28 bytes):

18 00 00 00 20 60 ce 47 12 90 c6 01 76 65 78 79 63 b8

COMM\_CO\_GUID (20 bytes):

10 00 00 00 e1 44 39 86 52 b6 5b 4b bf 40 17 82 9d 17  
4e 55

COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):

1d 00 00 00

COMM\_REMOTE\_CO (796 bytes):

18 03 00 00 01 00 00 00 0a 04 00 01 01 00 00 00 16 00  
00 00 00 00 00 00 01 00 00 00 10 00 00 00 00 00 00  
1d 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 20 60 ce 47 12 90 c6 01 80 e6 d4 00  
00 00 00 00 80 d5 c9 00 00 00 00 00 80 d5 c9 00 00 00  
00 00 01 00 00 00 01 00 00 00 e1 44 39 86 52 b6 5b 4b  
bf 40 17 82 9d 17 4e 55 76 65 78 79 63 b8 da 41 b1 1a  
41 63 46 eb be b3 d8 70 3b df 92 9e 42 4e a8 9b ee b7  
73 9a f1 4a bb 31 34 1e 35 5b a5 47 ae e5 55 ba 62 c2  
16 d7 bb 31 34 1e 35 5b a5 47 ae e5 55 ba 62 c2 16 d7  
f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57 ec 6f 8e 89  
bf 35 b5 89 c6 01 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  
a0 45 a1 01 00 00 00 00 0a 96 e5 8e f2 8f c6 01 0a 00  
53 00 74 00 6f 00 6e 00 65 00 00 00 74 00 2e 00 74 00  
78 00 74 00 00 00 00 00 00 00 00 00 00 00 00 00 00



```

      0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid      : 0 0 0 0 0 0 0 0
      0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs      : 0 0 0 0
+0x0f4 Spare2Wcs      : 0 0 0 0
+0x0f8 Extension      : a0 45 a1 1
+0x0fc Spare2Bin      : 0 0 0 0
+0x100 EventTime      : a 96 e5 8e f2 8f c6 1
+0x108 FileNameLength : a 0
+0x10a FileName       : 53 0 74 0 6f 0 6e 0
      65 0 0 0 74 0 2e 0
      74 0 78 0 74 0

```

COMM\_CO\_EXTENSION\_2 (72 bytes):

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 d4 6c 7a 42
f0 06 23 4a 34 42 0f 6e d4 5d 42 01 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 7b a0 0a 91 f2 8f c6 01

```

COMM\_EOP (4 bytes):

```
ff ff ff ff
```

## 4.5.5 Remove a Folder

### 4.5.5.1 Upstream Partner Sends Out CMD\_REMOTE\_CO

COMM\_BOP (4 bytes):

```
00 00 00 00
```

COMM\_COMMAND (4 bytes):

```
18 02 00 00
```

COMM\_TO (50 bytes):

```

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

```

COMM\_FROM (50 bytes):

```

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

```

COMM\_REPLICA (96 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

**COMM\_CXTION (98 bytes):**

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

**COMM\_JOIN\_GUID (20 bytes):**

```
10 00 00 00 9f b1 95 83 0a d4 fc 23 d1 08 1d 2a b5 f4
65 39
```

**COMM\_LAST\_JOIN\_TIME (8 bytes):**

```
06 74 70 89 f2 8f c6 01
```

**COMM\_REMOTE\_CO (796 bytes):**

```
18 03 00 00 1e 00 00 00 28 00 00 00 00 00 00 00 14 00
00 00 00 00 00 00 03 00 00 00 10 00 00 00 01 00 00 00
1e 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 21 60 ce 47 12 90 c6 01 f0 da c9 00
00 00 00 00 f0 da c9 00 00 00 00 00 f0 da c9 00 00 00
00 00 01 00 00 00 01 00 00 00 82 04 b0 60 c1 f6 92 4c
ad 20 40 84 68 ae f0 22 76 65 78 79 63 b8 da 41 b1 1a
41 63 46 eb be b3 d8 70 3b df 92 9e 42 4e a8 9b ee b7
73 9a f1 4a e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0 80 4c
22 99 12 28 0f 2e 2e 43 87 21 1e 45 15 7a cf 14 8e 89
bf 35 b5 89 c6 01 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
f8 1d e8 00 00 00 00 00 ac 2d bc 97 f2 8f c6 01 0a 00
53 00 74 00 6f 00 6e 00 65 00 00 00 74 00 2e 00 74 00
78 00 74 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00  
00 00 00 00

Type\_CHANGE\_ORDER\_RECORD\_\*

+0x000 SequenceNumber : 1e 0 0 0  
+0x004 Flags : 28 0 0 0  
(0x28 - CO\_FLAG\_LOCALCO | CO\_FLAG\_LOCATION\_CMD)  
+0x008 IFlags : 0 0 0 0 (0x0)  
+0x00c State : 14 0 0 0  
(Request outbound propagation)  
+0x010 ContentCmd : 0 0 0 0 (0x0)  
+0x014 LocationCmd : 3 0 0 0 (0x3)  
+0x018 FileAttributes : 10 0 0 0  
+0x01c FileVersionNumber : 1 0 0 0  
+0x020 PartnerAckSeqNumber : 1e 0 0 0 0 0 0 0 0  
+0x028 FileSize : 0 0 0 0 0 0 0 0  
+0x030 FileOffset : 0 0 0 0 0 0 0 0  
+0x038 FrsVsn : 21 60 ce 47 12 90 c6 1  
+0x040 FileUsn : f0 da c9 0 0 0 0 0  
+0x048 JrnlUsn : f0 da c9 0 0 0 0 0  
+0x050 JrnlFirstUsn : f0 da c9 0 0 0 0 0  
+0x058 OriginalReplicaNum : 1 0 0 0  
+0x05c NewReplicaNum : 1 0 0 0  
+0x060 ChangeOrderGuid : 82 4 b0 60 c1 f6 92 4c  
ad 20 40 84 68 ae f0 22  
+0x070 OriginatorGuid : 76 65 78 79 63 b8 da 41  
b1 1a 41 63 46 eb be b3  
+0x080 FileGuid : d8 70 3b df 92 9e 42 4e  
a8 9b ee b7 73 9a f1 4a  
+0x090 OldParentGuid : e6 87 d1 e5 aa 12 df 48  
ab c1 d7 94 a e0 80 4c  
+0x0a0 NewParentGuid : e6 87 d1 e5 aa 12 df 48  
ab c1 d7 94 a e0 80 4c  
+0x0b0 CxtionGuid : 22 99 12 28 f 2e 2e 43  
87 21 1e 45 15 7a cf 14  
+0x0c0 AckVersion : 8e 89 bf 35 b5 89 c6 1  
+0x0c8 Spare2Ull : 0 0 0 0 0 0 0 0  
+0x0d0 Spare1Guid : 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
+0x0e0 Spare2Guid : 0 0 0 0 0 0 0 0  
0 0 0 0 0 0 0 0  
+0x0f0 Spare1Wcs : 0 0 0 0  
+0x0f4 Spare2Wcs : 0 0 0 0  
+0x0f8 Extension : f8 1d e8 0  
+0x0fc Spare2Bin : 0 0 0 0  
+0x100 EventTime : ac 2d bc 97 f2 8f c6 1  
+0x108 FileNameLength : a 0  
+0x10a FileName : 53 0 74 0 6f 0 6e 0  
65 0 0 0 74 0 2e 0  
74 0 78 0 74 0

COMM\_CO\_EXTENSION\_2 (72 bytes):

```
48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 00 c8 7b ca 97 f2 8f c6 01
```

COMM\_EOP (4 bytes):

```
ff ff ff ff
```

#### 4.5.5.2 Downstream Partner Sends Out CMD\_REMOTE\_CO\_DONE

COMM\_BOP (4 bytes):

```
00 00 00 00
```

COMM\_COMMAND (4 bytes):

```
50 02 00 00
```

COMM\_TO (50 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

COMM\_FROM (50 bytes):

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

COMM\_REPLICA (96 bytes):

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

COMM\_CXTION (98 bytes):

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
```

```
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

**COMM\_JOIN\_GUID (20 bytes):**

```
10 00 00 00 9f b1 95 83 0a d4 fc 23 d1 08 1d 2a b5 f4
65 39
```

**COMM\_LAST\_JOIN\_TIME (8 bytes):**

```
06 74 70 89 f2 8f c6 01
```

**COMM\_BLOCK\_SIZE (8 bytes):**

```
00 00 00 00 00 00 00 00
```

**COMM\_FILE\_SIZE (8 bytes):**

```
00 00 00 00 00 00 00 00
```

**COMM\_FILE\_OFFSET (8 bytes):**

```
00 00 00 00 00 00 00 00
```

**COMM\_GVSN (28 bytes):**

```
18 00 00 00 21 60 ce 47 12 90 c6 01 76 65 78 79 63 b8
da 41 b1 1a 41 63 46 eb be b3
```

**COMM\_CO\_GUID (20 bytes):**

```
10 00 00 00 82 04 b0 60 c1 f6 92 4c ad 20 40 84 68 ae
f0 22
```

**COMM\_CO\_SEQUENCE\_NUMBER (4 bytes):**

```
1e 00 00 00
```

**COMM\_REMOTE\_CO (796 bytes):**





```

+0x05c NewReplicaNum      : 1 0 0 0
+0x060 ChangeOrderGuid   : 82 4 b0 60 c1 f6 92 4c
                          ad 20 40 84 68 ae f0 22
+0x070 OriginatorGuid    : 76 65 78 79 63 b8 da 41
                          b1 1a 41 63 46 eb be b3
+0x080 FileGuid          : d8 70 3b df 92 9e 42 4e
                          a8 9b ee b7 73 9a f1 4a
+0x090 OldParentGuid     : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
+0x0a0 NewParentGuid     : bb 31 34 1e 35 5b a5 47
                          ae e5 55 ba 62 c2 16 d7
+0x0b0 CxtionGuid        : f0 a3 f7 8 5c e0 94 49
                          9a 60 e7 cd 40 57 ec 6f
+0x0c0 AckVersion        : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull         : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid        : 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid        : 0 0 0 0 0 0 0 0
                          0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs         : 0 0 0 0
+0x0f4 Spare2Wcs         : 0 0 0 0
+0x0f8 Extension         : d0 4c a1 1
+0x0fc Spare2Bin         : 0 0 0 0
+0x100 EventTime         : ac 2d bc 97 f2 8f c6 1
+0x108 FileNameLength    : a 0
+0x10a FileName          : 53 0 74 0 6f 0 6e 0
                          65 0 0 0 74 0 2e 0
                          74 0 78 0 74 0

```

COMM\_CO\_EXTENSION\_2 (72 bytes):

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 00 bb 8e 0b 98 f2 8f c6 01

```

COMM\_EOP (4 bytes):

```
ff ff ff ff
```

## 4.5.6 Rename an Empty Folder

Rename stone to newstone.

### 4.5.6.1 Upstream Partner Sends Out CMD\_REMOTE\_CO

COMM\_BOP (4 bytes).

```
00 00 00 00
```

COMM\_COMMAND (4 bytes).

```
18 2 00 00
```

COMM\_TO (50 bytes).

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

COMM\_FROM (50 bytes).

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

COMM\_REPLICA (96 bytes).

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

COMM\_CXTION (98 bytes).

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes).

```
10 00 00 00 e9 8e c0 d0 ba 9c 1d 36 a 86 6b d8 bc 9a
87 12
```

COMM\_LAST\_JOIN\_TIME (8 bytes).

```
1c 6f 56 8c f3 8f c6 01
```

COMM\_REMOTE\_CO (796 bytes).

```
18 03 00 00 20 00 00 00 24 00 00 01 00 00 00 14 00
00 00 00 20 00 00 0f 00 00 00 10 00 00 00 01 00 00 00
20 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 23 30 57 0b 23 90 c6 01 e8 0c ca 00
00 00 00 00 e8 0c ca 00 00 00 00 00 e8 0c ca 00 00 00
00 00 01 00 00 00 01 00 00 00 b7 45 6b 03 8d 37 f1 49
89 6e 18 99 17 d6 89 f1 76 65 78 79 63 b8 da 41 b1 1a
41 63 46 eb be b3 07 fb b3 2d 6c ed 10 49 9f f0 79 d4
cb 2f 8f cb e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0 80 4c
```



```

+0x090 OldParentGuid      : e6 87 d1 e5 aa 12 df 48
                           ab c1 d7 94 a e0 80 4c
+0x0a0 NewParentGuid     : e6 87 d1 e5 aa 12 df 48
                           ab c1 d7 94 a e0 80 4c
+0x0b0 CxtionGuid        : 1a b 58 16 10 2c ae 4f
                           85 75 1f 3f ca 1e 94 af
+0x0c0 AckVersion        : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull         : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid        : 0 0 0 0 0 0 0 0
                           0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid        : 0 0 0 0 0 0 0 0
                           0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs         : 0 0 0 0
+0x0f4 Spare2Wcs         : 0 0 0 0
+0x0f8 Extension         : f8 1d e8 0
+0x0fc Spare2Bin         : 0 0 0 0
+0x100 EventTime         : 4c 10 8 91 f3 8f c6 1
+0x108 FileNameLength    : 10 0
+0x10a FileName          : 6e 0 65 0 77 0 73 0
                           74 0 6f 0 6e 0 65 0

```

COMM\_CO\_EXTENSION\_2 (72 bytes).

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 95 2f 15 e9
3e fb bb 87 d0 ec a7 73 9f 12 14 cf 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 9c e6 e4 92 f3 8f c6 01

```

COMM\_EOP (4 bytes): ff ff ff ff

#### 4.5.6.2 Downstream Partner Sends Out CMD\_SEND\_STAGE

COMM\_BOP (4 bytes).

```

00 00 00 00

```

COMM\_COMMAND (4 bytes).

```

28 02 00 00

```

COMM\_TO (50 bytes).

```

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

```

COMM\_FROM (50 bytes).

```

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

```

COMM\_REPLICA (96 bytes).

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

COMM\_CXTION (98 bytes).

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes).

```
10 00 00 00 e9 8e c0 d0 ba 9c 1d 36 0a 86 6b d8 bc 9a
87 12
```

COMM\_LAST\_JOIN\_TIME (8 bytes).

```
1c 6f 56 8c f3 8f c6 01
```

COMM\_BLOCK\_SIZE (8 bytes).

```
00 00 00 00 00 00 00 00
```

COMM\_FILE\_SIZE (8 bytes).

```
00 00 00 00 00 00 00 00
```

COMM\_FILE\_OFFSET (8 bytes).

```
00 00 00 00 00 00 00 00
```

COMM\_CO\_GUID (20 bytes):

```
10 00 00 00 b7 45 6b 03 8d 37 f1 49 89 6e 18 99 17 d6
89 f1
```



```

                                (Remote changing order staging
                                File fetch has started)
+0x010 ContentCmd           : 0 20 0 0
                                (0x2000 - REASON_RENAME_NEW_NAME)
+0x014 LocationCmd         : f 0 0 0 (0xf)
+0x018 FileAttributes      : 10 0 0 0
+0x01c FileVersionNumber   : 1 0 0 0
+0x020 PartnerAckSeqNumber : 20 0 0 0 0 0 0 0
+0x028 FileSize           : 0 0 0 0 0 0 0 0
+0x030 FileOffset         : 0 0 0 0 0 0 0 0
+0x038 FrsVsn             : 23 30 57 b 23 90 c6 1
+0x040 FileUsn            : a0 4 d5 0 0 0 0 0
+0x048 JrnlUsn            : e8 c ca 0 0 0 0 0
+0x050 JrnlFirstUsn       : e8 c ca 0 0 0 0 0
+0x058 OriginalReplicaNum  : 1 0 0 0
+0x05c NewReplicaNum       : 1 0 0 0
+0x060 ChangeOrderGuid    : b7 45 6b 3 8d 37 f1 49
                                89 6e 18 99 17 d6 89 f1
+0x070 OriginatorGuid     : 76 65 78 79 63 b8 da 41
                                b1 1a 41 63 46 eb be b3
+0x080 FileGuid           : 7 fb b3 2d 6c ed 10 49
                                9f f0 79 d4 cb 2f 8f cb
+0x090 OldParentGuid      : bb 31 34 1e 35 5b a5 47
                                ae e5 55 ba 62 c2 16 d7
+0x0a0 NewParentGuid      : bb 31 34 1e 35 5b a5 47
                                ae e5 55 ba 62 c2 16 d7
+0x0b0 CxtionGuid         : f0 a3 f7 8 5c e0 94 49
                                9a 60 e7 cd 40 57 ec 6f
+0x0c0 AckVersion         : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull          : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid         : 0 0 0 0 0 0 0 0
                                0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid         : 0 0 0 0 0 0 0 0
                                0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs          : 0 0 0 0
+0x0f4 Spare2Wcs          : 0 0 0 0
+0x0f8 Extension          : 40 4b a1 1
+0x0fc Spare2Bin          : 0 0 0 0
+0x100 EventTime          : 4c 10 8 91 f3 8f c6 1
+0x108 FileNameLength     : 10 0
+0x10a FileName           : 6e 0 65 0 77 0 73 0
                                74 0 6f 0 6e 0 65 0

```

COMM\_CO\_EXTENSION\_2 (72 bytes).

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 95 2f 15 e9
3e fb bb 87 d0 ec a7 73 9f 12 14 cf 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 eb 90 2b 93 f3 8f c6 01

```

COMM\_EOP (4 bytes).

```
ff ff ff ff
```

#### 4.5.6.3 Upstream Partner Sends Out CMD\_RECEIVING\_STAGE

COMM\_BOP (4 bytes).

```
00 00 00 00
```

COMM\_COMMAND (4 bytes).

38 02 00 00

COMM\_TO (50 bytes).

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00

COMM\_FROM (50 bytes).

10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40  
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00  
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00

COMM\_REPLICA (96 bytes).

10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0  
80 4c 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00  
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00  
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00  
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00  
45 00 29 00 00 00

COMM\_CXTION (98 bytes).

10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57  
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00  
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00  
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00  
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00  
30 00 30 00 44 00 00 00

COMM\_JOIN\_GUID (20 bytes).

10 00 00 00 e9 8e c0 d0 ba 9c 1d 36 0a 86 6b d8 bc 9a  
87 12

COMM\_LAST\_JOIN\_TIME (8 bytes).

1c 6f 56 8c f3 8f c6 01

COMM\_BLOCK (1,223 bytes).

c3 04 00 00 00 00 00 00 03 00 00 00 00 00 00 00 04





To interpret the bytes above, Header is dumped as follows.

```
Local var @ 0x3c6fb14 Type _STAGE_HEADER_
+0x000 Major          : 0 0 0 0
+0x004 Minor          : 3 0 0 0
+0x008 DataHigh       : 0 0 0 0
+0x00c DataLow        : 0 4 0 0
+0x010 Compression    : 0 0 0 0 0 0 0 0
+0x018 Attributes     : _FILE_NETWORK_OPEN_INFORMATION
+0x000 CreationTime   : 26 2f aa f6 f2 8f c6 1
+0x008 LastAccessTime : 26 2f aa f6 f2 8f c6 1
+0x010 LastWriteTime  : 26 2f aa f6 f2 8f c6 1
+0x018 ChangeTime     : 4c 10 8 91 f3 8f c6 1
+0x020 AllocationSize : 0 0 0 0 0 0 0 0
+0x028 EndOfFile      : 0 0 0 0 0 0 0 0
+0x030 FileAttributes : 10 0 0 0 0 0 0 0
+0x050 ChangeOrderCommand : _CHANGE_ORDER_RECORD_
+0x000 SequenceNumber : 1 0 0 0
+0x004 Flags          : 24 0 0 1 (0x1000024 -
CO_FLAG_COMPRESSED_STAGE |
CO_FLAG_LOCALCO |
CO_FLAG_CONTENT_CMD)
+0x008 IFlags        : 0 0 0 0 (0x0)
+0x00c State         : 1 0 0 0
(Allocating staging File
space for local Change Order)
+0x010 ContentCmd    : 0 20 0 0
(0x2000 - REASON_RENAME_NEW_NAME)
+0x014 LocationCmd   : f 0 0 0 (0xf)
+0x018 FileAttributes : 10 0 0 0
+0x01c FileVersionNumber : 1 0 0 0
+0x020 PartnerAckSeqNumber : 0 0 0 0 0 0 0 0
+0x028 FileSize      : 0 0 0 0 0 0 0 0
+0x030 FileOffset    : 0 0 0 0 0 0 0 0
+0x038 FrsVsn        : 23 30 57 b 23 90 c6 1
+0x040 FileUsn        : e8 c ca 0 0 0 0 0
+0x048 JrnlUsn        : e8 c ca 0 0 0 0 0
+0x050 JrnlFirstUsn   : e8 c ca 0 0 0 0 0
+0x058 OriginalReplicaNum : 1 0 0 0
+0x05c NewReplicaNum  : 1 0 0 0
+0x060 ChangeOrderGuid : b7 45 6b 3 8d 37 f1 49
89 6e 18 99 17 d6 89 f1
+0x070 OriginatorGuid : 76 65 78 79 63 b8 da 41
b1 1a 41 63 46 eb be b3
+0x080 FileGuid       : 7 fb b3 2d 6c ed 10 49
9f f0 79 d4 cb 2f 8f cb
+0x090 OldParentGuid  : e1 99 a7 ae 27 b4 22 45
81 59 5e 55 e0 93 ba 24
+0x0a0 NewParentGuid  : e1 99 a7 ae 27 b4 22 45
81 59 5e 55 e0 93 ba 24
+0x0b0 CxtionGuid     : 1a b 58 16 10 2c ae 4f
85 75 1f 3f ca 1e 94 af
+0x0c0 AckVersion     : 0 0 0 0 0 0 0 0
+0x0c8 Spare2Ull      : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid     : 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid     : 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs      : 0 0 0 0
+0x0f4 Spare2Wcs      : 0 0 0 0
+0x0f8 Extension       : 0 0 0 0
+0x0fc Spare2Bin       : 0 0 0 0
+0x100 EventTime      : 4c 10 8 91 f3 8f c6 1
+0x108 FileNameLength : 10 0
+0x10a FileName        : 6e 0 65 0 77 0 73 0
74 0 6f 0 6e 0 65 0
+0x368 FileObjId      : _FILE_OBJECTID_BUFFER
+0x000 ObjectId       : 7 fb b3 2d 6c ed 10 49
```

```

          9f f0 79 d4 cb 2f 8f cb
+0x010 BirthVolumeId : 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 ()
+0x020 BirthObjectId : 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 ()
+0x030 DomainId      : 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 ()
+0x3a8 CocExt        : _CO_RECORD_EXTENSION_WIN2K_
+0x000 FieldSize     : 0 0 0 0 ()
+0x004 Major         : 0 0 ()
+0x006 OffsetCount   : 0 0 ()
+0x008 Offset        : 0 0 0 0 ()
+0x00c OffsetLast    : 0 0 0 0 ()
+0x010 DataChecksum  : _DATA_EXTENSION_CHECKSUM_
+0x000 Prefix        : 0 0 0 0 0 0 0 0 ()
+0x008 Data          : 0 0 0 0 0 0 0 0
          0 0 0 0 0 0 0 0 ()
+0x3d0 CompressionGuid : d2 f7 d2 64 95 26 6d 43
          88 30 8d 3c 58 70 1e 15
(64d2f7d2-2695-436d-88308d3c58701e15)
+0x3e0 EncryptedDataHigh : 0 0 0 0 (0)
+0x3e4 EncryptedDataLow  : 0 0 0 0 (0)
+0x3e8 EncryptedDataSize : 0 0 0 0 0 0 0 0 (0)
+0x3f0 ReparseDataPresent : 0 0 0 0 (0)
+0x3f4 ReparsePointDataHigh : 0 0 0 0 (0)
+0x3f8 ReparsePointDataLow : 0 4 0 0 (0x400)

```

**Real data.**

```

00 00 00 00 c0 b0 20 03 00 00 00 02 00 30 0c 01 83 00
40 04 20 01 00 04 84 14 00 58 2a 24 04 78 40 00 1c 01
01 90 00 00 94 05 20 00 2c 20 00 2c 01 05 00 20 05 00
3c 15 00 0c 57 c2 d9 22 0f 80 2d 42 bc dd 3d e9 5b 01
4e 20 02 00 cc 00 09 01 3c 10 14 40 00 a9 00 12 00 01
03 b6 05 4a 0b 01 26 1b 01 b2 00 a0 0a 26 10 96 18 03
27 08 63 25 01 0b 1b 18 01 31 3e a0 01 53 00 08 02 7b
02 17 00 2f ff 01 ac 1f 00 09 17 01 93 00 03 2f 10 0e
17 b7 00 87 02 2f 04 87 12 01 55 03 87 10 05 87 f3 90
09 01 8d 00 07 81 15 80 01 81 81 04 05 00 00 07 fb b3
2d 6c ed 10 00 49 9f f0 79 d4 cb 2f 8f 06 cb 85 0b a5
03

```

**COMM\_BLOCK\_SIZE (8 bytes).**

```
c3 04 00 00 00 00 00 00
```

**COMM\_FILE\_SIZE (8 bytes).**

```
c3 04 00 00 00 00 00 00
```

**COMM\_FILE\_OFFSET (8 bytes).**

```
00 00 00 00 00 00 00 00
```

**COMM\_CO\_GUID (20 bytes).**

```
10 00 00 00 b7 45 6b 03 8d 37 f1 49 89 6e 18 99 17 d6
89 f1
```

COMM\_CO\_SEQUENCE\_NUMBER (4 bytes).

```
00 00 00 00
```

COMM\_EOP (4 bytes).

```
ff ff ff ff
```

#### 4.5.6.4 Downstream Partner Sends Out CMD\_REMOTE\_CO\_DONE

COMM\_BOP (4 bytes).

```
00 00 00 00
```

COMM\_COMMAND (4 bytes).

```
50 02 00 00
```

COMM\_TO (50 bytes).

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 31 00 00 00
```

COMM\_FROM (50 bytes).

```
10 00 00 00 e6 87 d1 e5 aa 12 df 48 ab c1 d7 94 0a e0
80 4c 1a 00 00 00 53 00 48 00 49 00 43 00 4f 00 2d 00
54 00 45 00 4d 00 50 00 2d 00 32 00 00 00
```

COMM\_REPLICA (96 bytes).

```
10 00 00 00 1a b2 f4 54 fd 03 74 43 8e 3b 28 75 e7 40
d9 58 48 00 00 00 44 00 4f 00 4d 00 41 00 49 00 4e 00
20 00 53 00 59 00 53 00 54 00 45 00 4d 00 20 00 56 00
4f 00 4c 00 55 00 4d 00 45 00 20 00 28 00 53 00 59 00
53 00 56 00 4f 00 4c 00 20 00 53 00 48 00 41 00 52 00
45 00 29 00 00 00
```

COMM\_CXTION (98 bytes).

```
10 00 00 00 f0 a3 f7 08 5c e0 94 49 9a 60 e7 cd 40 57
ec 6f 4a 00 00 00 31 00 43 00 33 00 30 00 46 00 43 00
33 00 38 00 2d 00 30 00 37 00 44 00 31 00 2d 00 34 00
30 00 44 00 46 00 2d 00 38 00 37 00 37 00 42 00 2d 00
38 00 43 00 34 00 38 00 42 00 46 00 32 00 31 00 32 00
30 00 30 00 44 00 00 00
```

COMM\_JOIN\_GUID (20 bytes).

```
10 00 00 00 e9 8e c0 d0 ba 9c 1d 36 0a 86 6b d8 bc 9a
87 12
```

COMM\_LAST\_JOIN\_TIME (8 bytes).

```
1c 6f 56 8c f3 8f c6 01
```

COMM\_BLOCK\_SIZE (8 bytes).

```
00 00 00 00 00 00 00 00
```

COMM\_FILE\_SIZE (8 bytes).

```
00 00 00 00 00 00 00 00
```

COMM\_FILE\_OFFSET (8 bytes).

```
00 00 00 00 00 00 00 00
```

COMM\_GVSN (28 bytes).

```
18 00 00 00 23 30 57 0b 23 90 c6 01 76 65 78 79 63 b8
da 41 b1 1a 41 63 46 eb be b3
```

COMM\_CO\_GUID (20 bytes).

```
10 00 00 00 b7 45 6b 03 8d 37 f1 49 89 6e 18 99 17 d6
89 f1
```

COMM\_CO\_SEQUENCE\_NUMBER (4 bytes).

```
20 00 00 0
```

COMM\_REMOTE\_CO (796 bytes).



```

+0x030 FileOffset      : 0 0 0 0 0 0 0 0
+0x038 FrsVsn         : 23 30 57 b 23 90 c6 1
+0x040 FileUsn        : 0 20 d5 0 0 0 0 0
+0x048 JrnlUsn        : e8 c ca 0 0 0 0 0
+0x050 JrnlFirstUsn   : e8 c ca 0 0 0 0 0
+0x058 OriginalReplicaNum : 1 0 0 0
+0x05c NewReplicaNum  : 1 0 0 0
+0x060 ChangeOrderGuid : b7 45 6b 3 8d 37 f1 49
                        89 6e 18 99 17 d6 89 f1
+0x070 OriginatorGuid : 76 65 78 79 63 b8 da 41
                        b1 1a 41 63 46 eb be b3
+0x080 FileGuid       : 7 fb b3 2d 6c ed 10 49
                        9f f0 79 d4 cb 2f 8f cb
+0x090 OldParentGuid  : bb 31 34 1e 35 5b a5 47
                        ae e5 55 ba 62 c2 16 d7
+0x0a0 NewParentGuid  : bb 31 34 1e 35 5b a5 47
                        ae e5 55 ba 62 c2 16 d7
+0x0b0 CxtionGuid     : f0 a3 f7 8 5c e0 94 49
                        9a 60 e7 cd 40 57 ec 6f
+0x0c0 AckVersion     : 8e 89 bf 35 b5 89 c6 1
+0x0c8 Spare2Ull      : 0 0 0 0 0 0 0 0
+0x0d0 Spare1Guid     : 0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0
+0x0e0 Spare2Guid     : 0 0 0 0 0 0 0 0
                        0 0 0 0 0 0 0 0
+0x0f0 Spare1Wcs      : 0 0 0 0
+0x0f4 Spare2Wcs      : 0 0 0 0
+0x0f8 Extension      : 40 4b a1 1
+0x0fc Spare2Bin      : 0 0 0 0
+0x100 EventTime      : 4c 10 8 91 f3 8f c6 1
+0x108 FileNameLength : 10 0 (16 bytes)
+0x10a FileName       : 6e 0 65 0 77 0 73 0
                        74 0 6f 0 6e 0 65 0

```

COMM\_CO\_EXTENSION\_2 (72 bytes).

```

48 00 00 00 01 00 02 00 18 00 00 00 30 00 00 00 00 00
00 00 00 00 00 00 18 00 00 00 01 00 00 00 95 2f 15 e9
3e fb bb 87 d0 ec a7 73 9f 12 14 cf 18 00 00 00 02 00
00 00 00 00 00 00 00 00 00 00 eb 90 2b 93 f3 8f c6 01

```

COMM\_EOP (4 bytes).

```
ff ff ff ff
```

## 5 Security

The following sections specify security considerations for implementers of the File Replication Service Protocol Specification.

### 5.1 Security Considerations for Implementers

FRS uses MD5 hashes of file contents to determine whether files are identical across replicas. MD5 is no longer considered a cryptographically secure hash function; thus, it might be computationally feasible to produce two files with different content that have the same hash value. Consider this when deploying FRS in an environment where files in the replicated folders can be created by untrustworthy sources.

### 5.2 Index of Security Parameters

| Security parameter               | Section |
|----------------------------------|---------|
| Transport security               | 2.1     |
| ACLs on Active Directory objects | 3.1.1   |



## 6 Appendix A: Full IDL

The FRS protocol contains two interfaces, whose IDL definitions are listed in this section. The IDL definitions for these interfaces import the "ms-dtyp.idl" file, as specified in [MS-DTYP] section 5.

### 6.1 Appendix A.1: frsapi.idl

The IDL definition for this interface imports the "ms-dtyp.idl" file, as specified in [MS-DTYP] section 5.

```
import "ms-dtyp.idl";

[
    uuid (D049B186-814F-11D1-9A3C-00C04FC9B232), version(1.1),
    pointer_default (unique)
] interface NtFrsApi

{
    // This method is not used over the wire
    unsigned long Opnum0NotUsedOnWire(
        void
    );

    // This method is not used over the wire
    unsigned long Opnum1NotUsedOnWire(
        void
    );

    // This method is not used over the wire
    unsigned long Opnum2NotUsedOnWire(
        void
    );

    // This method is not used over the wire
    unsigned long Opnum3NotUsedOnWire(
        void
    );

    unsigned long NtFrsApi_Rpc_Set_DsPollingIntervalW(
        [in] handle_t Handle,
        [in] unsigned long UseShortInterval,
        [in] unsigned long LongInterval,
        [in] unsigned long ShortInterval
    );

    unsigned long NtFrsApi_Rpc_Get_DsPollingIntervalW(
        [in] handle_t Handle,
        [out] unsigned long *Interval,
        [out] unsigned long *LongInterval,
        [out] unsigned long *ShortInterval
    );

    // This method is not used over the wire
    unsigned long Opnum6NotUsedOnWire(
        void
    );

    unsigned long NtFrsApi_Rpc_InfoW(
        [in] handle_t Handle,
        [in, range(0,65536)] unsigned long BlobSize,
        [in, out, size_is(BlobSize), unique] unsigned char *Blob
    );

    unsigned long NtFrsApi_Rpc_IsPathReplicated(
        [in] handle_t Handle,
        [in, string, unique] PWCHAR Path,
        [in] unsigned long ReplicaSetTypeOfInterest,
```

```

        [out] unsigned long *Replicated,
        [out] unsigned long *Primary,
        [out] unsigned long *Root,
        [out] GUID *ReplicaSetGuid
    );

    unsigned long NtFrsApi_Rpc_WriterCommand(
        [in] handle_t Handle,
        [in] unsigned long Command
    );

    unsigned long NtFrsApi_Rpc_ForceReplication(
        [in] handle_t Handle,
        [in, unique] GUID *ReplicaSetGuid,
        [in, unique] GUID *CxtionGuid,
        [in, string, unique] PWCHAR ReplicaSetName,
        [in, string, unique] PWCHAR PartnerDnsName
    );
}

```

## 6.2 Appendix A.2: frsrpc.idl

The IDL definition for this interface imports the "ms-dtyp.idl" file, as specified in [MS-DTYP] section 5.

```

import "ms-dtyp.idl";

[
    uuid (F5CC59B4-4264-101A-8C59-08002B2F8426), version(1.1),
    pointer_default (unique)
] interface frsrpc
{
    typedef struct _COMM_PACKET {
        unsigned long Major;
        unsigned long Minor;
        unsigned long CsId;
        unsigned long MemLen;
        [range(0, 256 * 1024)] unsigned long PktLen;
        unsigned long UpkLen;
        [size_is(PktLen)] unsigned char *Pkt;
        [ignore] void *DataName;
        [ignore] void *DataHandle;
    } COMM_PACKET, *PCOMM_PACKET;

    unsigned long
    FrsRpcSendCommPkt(
        [in] handle_t Handle,
        [in] PCOMM_PACKET CommPkt
    );

    unsigned long
    FrsRpcVerifyPromotionParent(
        [in] handle_t Handle,
        [in, string, unique] PWCHAR ParentAccount,
        [in, string, unique] PWCHAR ParentPassword,
        [in, string, unique] PWCHAR ReplicaSetName,
        [in, string, unique] PWCHAR ReplicaSetType,
        [in] unsigned long PartnerAuthLevel,
        [in] unsigned long GuidSize
    );

    unsigned long
    FrsRpcStartPromotionParent(
        [in] handle_t Handle,
        [in, string, unique] PWCHAR ParentAccount,
        [in, string, unique] PWCHAR ParentPassword,
        [in, string, unique] PWCHAR ReplicaSetName,

```

```

[in, string, unique] PWCHAR ReplicaSetType,
[in, string, unique] PWCHAR CxtionName,
[in, string, unique] PWCHAR PartnerName,
[in, string, unique] PWCHAR PartnerPrincName,
[in] unsigned long PartnerAuthLevel,
[in, range(sizeof(GUID),sizeof(GUID))] unsigned long GuidSize,
[in, size_is(GuidSize), unique] unsigned char *CxtionGuid,
[in, size_is(GuidSize), unique] unsigned char *PartnerGuid,
[in, out, size_is(GuidSize), unique] unsigned char *ParentGuid
);

unsigned long FrsNOP();

// This method is not implemented
unsigned long Opnum4NotUsedOnWire(
    void
);

// This method is not implemented
unsigned long Opnum5NotUsedOnWire(
    void
);

// This method is not implemented
unsigned long Opnum6NotUsedOnWire(
    void
);

// This method is not implemented
unsigned long Opnum7NotUsedOnWire(
    void
);

// This method is not implemented
unsigned long Opnum8NotUsedOnWire(
    void
);

// This method is not implemented
unsigned long Opnum9NotUsedOnWire(
    void
);

// This method is not implemented
unsigned long Opnum10NotUsedOnWire(
    void
);
}

```

## 7 Appendix B: Product Behavior

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

- Windows 2000 operating system
- Windows Server 2003 operating system
- Windows Server 2008 operating system
- Windows Server 2008 R2 operating system
- Windows Server 2012 operating system
- Windows Server 2012 R2 operating system
- Windows Server 2016 operating system
- Windows Server operating system

Exceptions, if any, are noted ~~below in this section~~. If ~~a-an update version~~, service pack or ~~Quick-Fix Engineering (QFE)~~Knowledge Base (KB) number appears with ~~thea~~ product ~~version,name, the~~ behavior changed in that ~~service-pack-or-QFE-update~~. The new behavior also applies to subsequent ~~service packs-of-the-product~~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 1.3: Windows FRS uses File Object ID (as specified in [MS-FSCC] section 2.1.3) as the ID of the files and folders inside a replica set. To get the object ID, FRS first checks if the file or folder already has an object ID. If an object ID does not exist, FRS creates a new GUID and save it on the file or folder. If an object ID already exists, FRS can either use it or reset it to a new value if an object ID is already set on a remote machine for the same file.

<2> Section 1.5: The Windows implementation of FRS works with Active Directory, as specified in [MS-ADTS].

<3> Section 1.7.2:

- Windows 2000 Server operating system uses FRS version 0.3.
- Windows 2000 Server operating system Service Pack 3 (SP3) QFE uses FRS version 0.6.
- Windows 2000 Server SP3 QFE+1 uses FRS version 0.7.
- Windows Server 2003 and Windows Server 2008 use FRS version 0.8.
- Windows 2000 Server operating system Service Pack 4 (SP4) and Windows Server 2008 R2 use FRS version 0.9.

**Note** Although Windows 2000 Server is an earlier version of Windows than Windows Server 2003, Windows 2000 Server SP4 was released subsequent to the initial release of Windows Server 2003—thus its later version number.

- Windows Server 2012 uses FRS version 0.9.
- Windows Server 2012 R2 uses FRS version 0.9.

- Windows Server 2016 uses FRS version 0.9.

- Windows Server operating system uses FRS version 0.9.

<4> Section 2.1: Windows FRS runs under the security context of the local system account.

<5> Section 2.2.3.2: The file or folder USN number in the NTFS file system.

<6> Section 2.2.3.2: Latest USN number contributing to this change order.

<7> Section 2.2.3.2: First USN number contributing to this change order.

<8> Section 2.2.3.2: Old local replica set ID number.

<9> Section 2.2.3.2: New local replica set ID number.

<10> Section 2.2.3.4: The value 0 is used for Windows 2000 Server. The value 1 is used for Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, and Windows Server 2012 R2. There is no difference in the content or handling of version 0 and version 1 of CHANGE\_ORDER\_RECORD\_EXTENSION.

<11> Section 2.2.3.6.12: These are the only two values used by Windows.

<12> Section 2.2.3.6.13: The maximal size of data in bytes is 0x10000. Windows sends, and expects to receive, at most 0x10000 bytes of Data.

<13> Section 2.2.3.10: By default, Windows has staging file compression enabled. Staging file compression can be disabled by setting registry key [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Compress Staging Files] to 0.

<14> Section 2.2.3.10: Windows 2000 Server uses a Minor value of NTFRS\_STAGE\_MINOR\_1. Windows 2000 Server SP3 and Windows 2000 operating system Service Pack 4 (SP4) use a Minor value of NTFRS\_STAGE\_MINOR\_2. Windows Server 2003, Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, Windows Server 2016, and Windows Server 2016 operating system use a Minor value of NTFRS\_STAGE\_MINOR\_3.

<15> Section 2.3: Windows FRS creates the objects in the directory after the first reboot during DC promotion. The objects mentioned above are created by administrators through Active Directory tools. In Windows, these objects are created by administrators using the DFS user interface or DCPromo.exe.

<16> Section 2.3: Windows also uses objectGuid and uSNChanged attributes for every object.

<17> Section 2.3: Windows uses objectGuid and distinguishedName to identify this object in its internal memory structure, as specified in [MS-ADSC] section 2.265.

<18> Section 2.3: Windows uses uSNChanged to calculate a checksum on all Active Directory objects' uSNChanged value returned by an Active Directory poll. It then waits until two polling cycles return the same check sum before using the DS data. This avoids using a set of Active Directory objects that is still changing, as specified in [MS-ADSC] section 2.265.

<19> Section 2.3.1.2: If no value is set for this attribute, the Windows implementation of FRS uses a value of "\*.tmp,\*.bak,~\*", which is also the default value for this attribute in Active Directory.

When the filter changes, FRS applies the new filter to all files and folders that are subsequently modified or created.

FRS will not replicate old files that were excluded before but are allowed with the new filter. FRS will replicate these files only when there are some changes made to the old files.

Similarly, files that were replicated before the filter changed will not be deleted by FRS, but they will be excluded thereafter.

For example, if a filter is changed from "\*.tmp, \*.bak" to "\*.tmp, \*.doc", then FRS will not replicate any previously created \*.bak files, but any changes made to existing \*.bak file or any new \*.bak file will be replicated. Similarly, any previously replicated \*.doc file will not be removed from any member. However, any new or changed \*.doc file will not be replicated out by FRS after the filter change.

FRS does not replicate NTFS mount points, nor does it replicate files and folders that have been encrypted by the Encrypting File System (EFS).

<20> Section 2.3.1.2: Windows does not specify a default value for this attribute in Active Directory.

<21> Section 2.3.1.2: Windows does not set this value by default.

<22> Section 2.3.1.2: This attribute is only available on Windows 2000 Server and Windows Server 2003.

<23> Section 2.3.1.6: Windows sets the default value of the attribute to FRS working path where FRS will store its implementation-specific data.

<24> Section 2.3.1.6: Windows sets the default value of the attribute to FRS working path where FRS will store its implementation-specific data.

<25> Section 2.3.1.6: Windows sets the default priority for DFS replica set replication to 7.

<26> Section 2.3.1.8: Windows sets the default value of the attribute to FRS working path where FRS will store its implementation-specific data.

<27> Section 2.3.2.2: The default replica set name for a SYSVOL object is "Domain System Volume (SYSVOL share)"; however, any name can be used, and the name can be changed during promotion. Windows creates the following two attributes in Active Directory, but never reads or uses them after creation:

- frsReplicaSetGUID: Binary {00000000-0000-0000-0000-000000000000}
- frsVersionGuid: Binary {00000000-0000-0000-0000-000000000000}

<28> Section 2.3.2.4: Windows sets the default priority, as indicated by the priority bits in the NTDS Connection object's Options attribute, for SYSVOL replica set replication to 0.

<29> Section 2.4: Windows responds to requests for FRS performance counter statistics through the Windows Remote Registry Protocol.

<30> Section 2.4: Windows collects these statistics by querying the implementation-specific internal data structures inside the FRS process memory space.

<31> Section 3.1.1.5: Windows uses the NTFS file reference number as the unique key to identify the file in NTFS.

<32> Section 3.1.1.7: Change orders remain in the outbound log for seven days. This value is saved as minutes inside the registry using 4 bytes.

<33> Section 3.1.1.8: In Windows Server 2008, Windows Server 2008 R2 operating system, Windows Server 2012, Windows Server 2012 R2, [Windows Server 2016](#), and Windows Server ~~2016~~, [operating system](#) read-only domain controllers (RODCs) are supported. To support RODCs in Windows Server 2008, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2, [Windows Server 2016](#), and Windows Server ~~2016~~, [operating system](#) no outbound connections are created for the RODC Member objects. Because no outbound connections exist for RODC members, no changes on RODCs can ever be replicated out.

<34> Section 3.1.1.9: The only type of reparse point that is replicated as a reparse point by Windows implementations of FRS is IO\_REPARSE\_TAG\_SYMLINK. Reparse point types IO\_REPARSE\_TAG\_SIS and IO\_REPARSE\_TAG\_HSM are replicated as normal files rather than as reparse points. No other reparse point types (IO\_REPARSE\_TAG\_MOUNT\_POINT, IO\_REPARSE\_TAG\_DFS, and so on) are replicated, neither as reparse points nor as normal files.

<35> Section 3.1.1.10.2: The default value is 3,000, which is stored in the following registry entry: HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Maximum CO Retry Count.

<36> Section 3.1.1.10.2: The default value is 10,080 (which is 7 days expressed in minutes), stored in the following registry entry: HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Maximum CO Retry Timeout In Minutes.

<37> Section 3.1.2: For domain controllers, the default short and long polling intervals are 5 minutes each. Domain controllers always use the short interval, regardless of the long interval setting.

For member servers, the default short polling interval is 5 minutes, and the long polling interval is 60 minutes.

<38> Section 3.1.2: If there have been no FRS configuration changes in Active Directory after the completion of eight short polling intervals, FRS automatically begins polling according to the long polling interval. If one of the following events occurs, the polling cycle is reset:

- Replica Member is added or removed (see section 3.1.1.3).
- Connection is added or removed.
- Schedule is changed.
- File or folder filter is changed.

<39> Section 3.1.3: In Windows implementations, change orders remain in the outbound log for seven days by default. This value is saved as minutes inside the registry entry [HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Outlog Change History In Minutes].

<40> Section 3.1.3: Windows-based servers use the registry entry RPC TCP/IP Port Assignment under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\] to restrict replication traffic to a specific port.

<41> Section 3.1.5.1: Windows modifies the following registry entries when removing a machine from the replica set: [HKLM\System\CurrentControlSet\Services\NtFrs\Parameters\Replica Sets\<ReplicaSetGuid>] is deleted and [HKLM\System\CurrentControlSet\Services\NtFrs\Parameters\Cumulative Replica Sets\<ReplicaSetGuid>] is deleted. Also, values are added to [HKLM\SYSTEM\CurrentControlSet\Control\BackupRestore\FilesNotToBackup] (or [HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\FilesNotToBackup] if the former does not exist) and [HKLM\SYSTEM\CurrentControlSet\Control\BackupRestore\KeysNotToRestore] to ensure that NtBackup does not back up the files of the deleted replica and does not restore the deleted registry keys of deleted replica.

<42> Section 3.1.5.1: Encrypting File System (EFS)-encrypted files and folders are not replicated by the Windows implementation of FRS.

<43> Section 3.1.5.1: Windows Server 2008 and Windows Server 2008 R2, with domain functional level set to DS\_BEHAVIOR\_2000 or DS\_BEHAVIOR\_2003, use FRS to replicate the SYSVOL replica sets. Windows 2000 and Windows Server 2003 will always use FRS to replicate SYSVOL replica sets.

<44> Section 3.1.5.1: Windows FRS also creates the DFSR SYSVOL replica set object in the DS, and then starts the DFSR service and allows it to replicate the SYSVOL replica set, as specified in [MS-DFSRH]. If the domain functional level is DS\_BEHAVIOR\_WIN2008, DS\_BEHAVIOR\_WIN2008R2, DS\_BEHAVIOR\_WIN2012, DS\_BEHAVIOR\_WIN2012R2, or DS\_BEHAVIOR\_WIN2016 (as specified in [MS-ADTS] section 6.1.4.3), FRS removes the SYSVOL replica set object in the Directory Service (DS) and stops replicating the SYSVOL replica set. DFSR, not FRS, is used to replicate the SYSVOL replica set when the domain functional level is DS\_BEHAVIOR\_WIN2008, DS\_BEHAVIOR\_WIN2008R2, DS\_BEHAVIOR\_WIN2012, DS\_BEHAVIOR\_WIN2012R2, or DS\_BEHAVIOR\_WIN2016.

<45> Section 3.2.3: Windows enables or disables the user or process to change the Active Directory polling interval, according to the value of the registry subkey under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Set Ds Polling Interval\Access, checks are [Enabled or Disabled]].

If the registry subkey is not present, Windows considers **IsSetDsPollingIntervalAccessCheckEnabled** to have a value of "Enabled".

<46> Section 3.2.3: Windows allows the user or process to change the access to read/write, according to the value of the registry subkey under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Set Ds Polling Interval\Access, checks require [Full Control or Read]].

If the registry subkey is not present, Windows considers **SetDsPollingIntervalAccessRequired** to have a value of "Read".

<47> Section 3.2.3: Windows enables or disables the user or process to get the information about the Active Directory polling interval, according to the value of the registry subkey under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Get Ds Polling Interval\Access, checks are [Enabled or Disabled]].

If the registry subkey is not present, Windows considers **IsGetDsPollingIntervalAccessCheckEnabled** to have a value of "Enabled".

<48> Section 3.2.3: Windows allows the user or process to change the access to read/write, according to the value of the registry subkey under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Get Ds Polling Interval\Access, checks require [Full Control or Read]].

If the registry subkey is not present, Windows considers **GetDsPollingIntervalAccessRequired** to have a value of "Read".

<49> Section 3.2.3: Windows enables or disables the user or process to get the internal information about FRS, according to the value of the registry subkey under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Get Internal Information\Access, checks are [Enabled or Disabled]].

If the registry subkey is not present, Windows considers **IsGetInfoWAccessCheckEnabled** to have a value of "Enabled".

<50> Section 3.2.3: Windows allows the user or process to change the access to read/write, according to the value of the registry subkey under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Get Internal Information\Access, checks require [Full Control or Read]].

If the registry subkey is not present, Windows considers **GetInfoWAccessRequired** to have a value of "Read".

<51> Section 3.2.3: Windows enables or disables the FRS writer to issue commands. according to the value of the registry subkey under



[HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Writer Commands\Access, checks are [Enabled or Disabled]].

If the registry subkey is not present, Windows considers **IsWriterCommandCheckEnabled** to have a value of "Enabled".

<52> Section 3.2.3: Windows allows the user or process to change the access to read/write, according to the value of the registry subkey under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Writer Commands\Access, checks require [Full Control or Read]].

If the registry subkey is not present, Windows considers **WriterCommandAccessRequired** to have a value of "Read".

<53> Section 3.2.3: Windows enables or disables the FRS writer to issue commands. according to the value of the registry subkey under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Is Path Replicated\Access, checks are [Enabled or Disabled]].

If the registry subkey is not present, Windows considers **IsPathReplicatedCheckEnabled** to have a value of "Enabled".

<54> Section 3.2.3: Windows allows the user or process to change the access to read/write, according to the value of the registry subkey under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Is Path Replicated\Access, checks require [Full Control or Read]].

If the registry subkey is not present, Windows considers **ReplicatedPathAccessRequired** to have a value of "Read".

<55> Section 3.2.3: Windows enables or disables the FRS writer to issue commands, according to the value of the registry subkey under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Force Replication\Access, checks are [Enabled or Disabled]].

If the registry subkey is not present, Windows considers **IsForceReplicationCheckEnabled** to have a value of "Enabled".

<56> Section 3.2.3: Windows allows the user or process to change the access to read/write, according to the value of the registry subkey under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\Force Replication\Access, checks require [Full Control or Read]].

If the registry subkey is not present, Windows considers **ForceReplicationAccessRequired** to have a value of "Read".

<57> Section 3.2.4: Windows enables or disables the access check on each FRSAPI message according to the value of the registry subkey that is associated with the message under [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\FunctionName\Access checks are [Enabled or Disabled]].

If the access check is enabled for an FRS message, Windows restricts user access to this FRS message by impersonating the caller with an RPC impersonation level of `RPC_C_IMPL_LEVEL_IMPERSONATE`, as specified in [MS-RPCE] section 2.2.1.1.9, and opening the key [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\Access Checks\FunctionName\Access checks require [Full Control or Read]] with the access specified in the key.

FunctionName is one of the following:

- Set Ds Polling Interval

- Get Ds Polling Interval
- Get Internal Information
- Is Path Replicated
- Writer Commands
- Force Replication

<58> Section 3.2.4.1: Windows implementation persists the long interval in a registry key to use it across service restarts. This value persisted in registry is in the range of 0 to 0x00008BCF inclusive. Any value higher is truncated to 0x00008BCF.

<59> Section 3.2.4.1: Windows implementation persists the short interval in a registry key to use it across service restarts. This value persisted in registry is in the range of 0 to 0x00008BCF inclusive. Any value higher is truncated to 0x00008BCF.

<60> Section 3.2.4.3: The following information is returned inside the BLOB by this method in Windows FRS:

- Version information
- Active replica sets
- DS objects related to the local machine
- Memory usage
- IDTable
- Outbound log
- Inbound log

<61> Section 3.2.4.3: Windows FRS uses NTFRSAPI\_INFO data structures (as specified in section 2.2.2.1) in *Blob*. This field is transmitted across the wire as a BLOB, and is not marshaled by RPC. Data in the Windows BLOB includes memory usage, outbound log, and so on).

<62> Section 3.2.4.3: Windows supplies the following information in this call:

1. FRS version information
2. FRS internal information on replica sets
3. FRS internal information on DS
4. FRS internal information on memory usage
5. FRS internal information on Jet DB tables
  - IDTable
  - Outbound log table
  - Inbound log table

<63> Section 3.2.4.4: The path is in the following formats (there is no Windows-based client tool that uses this method):

- [DriveLetter]:\[FolderPath]
- \??\Volume{Volume GUID}\[FolderPath]

- \\.\Volume{Volume GUID}\[FolderPath]
- \\?\Volume{Volume GUID}\[FolderPath]

<64> Section 3.3.4.1: Windows-based servers send the NetBIOS name whenever the upstream partner fails to find the FQDN of the downstream partner.

<65> Section 3.3.4.1: In Windows implementations, the FRS server does not validate the parameter and requires the client to send the correct data, as specified in section 2.2.3.6.3.

<66> Section 3.3.4.1: In Windows implementations, the FRS server does not validate the parameter and requires the client to send the correct data, as specified in section 2.2.3.6.4.

<67> Section 3.3.4.1: Windows-based servers send the NetBIOS name whenever the upstream partner fails to find the FQDN.

<68> Section 3.3.4.1: In Windows implementations, the FRS server does not validate the parameter and requires the client to send the correct data, as specified in section 2.2.3.6.4.

<69> Section 3.3.4.1: In Windows implementations, the FRS server does not validate the parameter and requires the client to send the correct data, as specified in section 2.2.3.6.5.

<70> Section 3.3.4.1: In Windows implementations, the FRS server does not validate the parameter and requires the client to send the correct data, as specified in section 2.2.3.6.6.

<71> Section 3.3.4.1: In Windows the volume allocation size for the staging file is used.

<72> Section 3.3.4.1: In Windows this field specifies the time stamp of the update sequence number (USN) change.

<73> Section 3.3.4.1.1: In Windows, the FRS also uses CO\_LOCATION\_MOVEIN, CO\_LOCATION\_MOVEIN2, and CO\_LOCATION\_MOVEDIR.

<74> Section 3.3.4.1.4: In Windows, the FRS also uses CO\_LOCATION\_MOVEIN, CO\_LOCATION\_MOVEIN2, and CO\_LOCATION\_MOVEDIR.

<75> Section 3.3.4.1.5: In Windows, the FRS also uses CO\_LOCATION\_MOVEOUT.

<76> Section 3.3.4.2: Windows sets PartnerAuthLevel to 0x0.

<77> Section 3.3.4.2: Newly created DCs create volatile connections to improve the speed of initial sync.

<78> Section 3.3.4.4.1: For non-connection establishment communications: If the client is an upstream partner, the timeout is at least 30 seconds. If the client is a downstream partner, the timeout is at least 60 seconds.

<79> Section 3.3.4.4.1: Windows-based servers use the NetBIOS name whenever the upstream partner fails to find the FQDN of the downstream partner.

<80> Section 3.3.4.4.1: Windows-based servers send the NetBIOS name whenever the upstream partner fails to find the FQDN.

<81> Section 3.3.4.4.4: Default Maximum Partner Clock Skew is 30 minutes and is configurable through the Windows registry.

<82> Section 3.3.4.4.4.1.1: Volume allocation size for the staging file is used.

<83> Section 3.3.4.4.4.1.1: The time stamp of the update sequence number (USN) change.

<84> Section 3.3.4.4.6: In Windows implementations, the registry key [HKLM\SYSTEM\CurrentControlSet\Services\NtFrs\Parameters\ Suppress Identical Updates to Files] can be created or updated to 0 to force identical updates to replicate.

<85> Section 3.3.4.4.6: The Windows downstream partner sends 65,536 bytes of staging file data in the CMD\_RECEIVING packet.

<86> Section 3.3.4.4.7: In Windows, the CocExt structure is only filled for downstream partners that are running Windows 2000 Server. Windows 2000 Server partners will have a FRS Minor version of NTFRS\_COMM\_MINOR\_3 or below.

<87> Section 3.3.4.6: If the server does not reply to the client in 10 seconds, the client tries again. The client doubles the interval every retry and stops retrying after 1 hour.

## 8 Change Tracking

~~No table of This section identifies changes is available. The that were made to this document is either new or has had no changes since its 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).

| <b>Section</b>                        | <b>Description</b>   | <b>Revision class</b> |
|---------------------------------------|--|-----------------------|
| <u>7 Appendix B: Product Behavior</u> | <u>Added Windows Server to the list of applicable products and product behavior notes.</u> | <u>Major</u>          |

## 9 Index

### A

Abstract data model  
FRSAPI interface (section 3.1.1 67, section 3.2.1 78)  
FRSRPC interface (section 3.1.1 67, section 3.3.1 88)  
PERFFRS interface 67  
Applicability 17

### C

Capability negotiation 17  
Change order handling sequence example 122  
Change tracking 245  
CHANGE\_ORDER\_COMMAND structure 21  
CHANGE\_ORDER\_RECORD\_EXTENSION structure 28  
CO\_RECORD\_EXTENSION\_WIN2K structure 28  
COMM\_BLOCK structure 40  
COMM\_BLOCK\_SIZE structure 40  
COMM\_BOP structure 34  
COMM\_CO\_EXT\_WIN2K structure 43  
COMM\_CO\_EXTENSION\_2 structure 43  
COMM\_CO\_GUID structure 42  
COMM\_CO\_SEQUENCE\_NUMBER structure 42  
COMM\_COMMAND structure 34  
COMM\_COMPRESSION\_GUID structure 39  
COMM\_CXTION structure 37  
COMM\_EOP structure 44  
COMM\_FILE\_OFFSET structure 41  
COMM\_FILE\_SIZE structure 41  
COMM\_FROM structure 36  
COMM\_GVSN structure 41  
COMM\_JOIN\_GUID structure 37  
COMM\_JOIN\_TIME structure 39  
COMM\_LAST\_JOIN\_TIME structure 38  
Comm\_packet example 125  
COMM\_PACKET structure 29  
COMM\_REMOTE\_CO structure 43  
COMM\_REPLICA structure 36  
COMM\_REPLICA\_VERSION\_GUID structure 39  
COMM\_TO structure 35  
COMM\_VVECTOR structure 38  
Common data types 19  
Connection establishment sequence example 121

### D

Data model - abstract  
FRSAPI interface (section 3.1.1 67, section 3.2.1 78)  
FRSRPC interface (section 3.1.1 67, section 3.3.1 88)  
PERFFRS interface 67  
Data types 19  
common - overview 19  
DATA\_EXTENSION\_CHECKSUM structure 44  
DATA\_EXTENSION\_PREFIX structure 44  
DATA\_EXTENSION\_RETRY\_TIMEOUT structure 45  
Directory service schema elements 48

### E

Elements - directory service schema 48  
Examples

- change order handling sequence 122
- comm\_packet 125
- connection establishment sequence 121
- normal sync 157
- overview 120
- sysvol initial sync 127

## F

- Fields - vendor-extensible 18
- FRSAPI interface
  - abstract data model (section 3.1.1 67, section 3.2.1 78)
  - initialization (section 3.1.3 73, section 3.2.3 78)
  - local events (section 3.1.6 77, section 3.2.6 88)
  - message processing 73
  - overview 77
  - sequencing rules 73
  - timer events (section 3.1.5 74, section 3.2.5 88)
  - timers (section 3.1.2 73, section 3.2.2 78)
- FrsNOP method 97
- FRSRPC interface
  - abstract data model (section 3.1.1 67, section 3.3.1 88)
  - initialization (section 3.1.3 73, section 3.3.3 89)
  - local events (section 3.1.6 77, section 3.3.6 119)
  - message processing (section 3.1.4 73, section 3.3.4 89)
  - overview 88
  - sequencing rules (section 3.1.4 73, section 3.3.4 89)
  - timer events (section 3.1.5 74, section 3.3.5 119)
  - timers (section 3.1.2 73, section 3.3.2 88)
- FrsRpcSendCommPkt method 98
- FrsRpcStartPromotionParent method 95
- FrsRpcVerifyPromotionParent method 117
- Full IDL 233

## G

- Glossary 8
- GVSN structure 21

## I

- IDL 233
- Implementer - security considerations 232
- Index of security parameters 232
- Informative references 14
- Initialization
  - FRSAPI interface (section 3.1.3 73, section 3.2.3 78)
  - FRSRPC interface (section 3.1.3 73, section 3.3.3 89)
  - PERFFRS interface 73
- Introduction 8

## L

- Local events
  - FRSAPI interface (section 3.1.6 77, section 3.2.6 88)
  - FRSRPC interface (section 3.1.6 77, section 3.3.6 119)
  - PERFFRS interface 77

## M

- Message processing
  - FRSAPI interface 73
  - FRSRPC interface (section 3.1.4 73, section 3.3.4 89)
  - PERFFRS interface 73

## Messages

- common data types 19
- data types 19
- performance counters 60
- transport 19

## N

- Normal sync example 157
- Normative references 13
- NTFRSAPI\_INFO structure 19
- NtFrsApi\_Rpc\_ForceReplication method 87
- NtFrsApi\_Rpc\_Get\_DsPollingIntervalW method 81
- NtFrsApi\_Rpc\_InfoW method 82
- NtFrsApi\_Rpc\_IsPathReplicated method 83
- NtFrsApi\_Rpc\_Set\_DsPollingIntervalW method 80
- NtFrsApi\_Rpc\_WriterCommand method 85

## O

- Overview 14
- Overview (synopsis) 14

## P

- Parameters - security index 232
- PCHANGE\_ORDER\_COMMAND 21
- PCHANGE\_ORDER\_RECORD\_EXTENSION 28
- PCO\_RECORD\_EXTENSION\_WIN2K 28
- PCOMM\_BLOCK 40
- PCOMM\_BLOCK\_SIZE 40
- PCOMM\_BOP 34
- PCOMM\_CO\_EXT\_WIN2K 43
- PCOMM\_CO\_EXTENSION\_2 43
- PCOMM\_CO\_GUID 42
- PCOMM\_CO\_SEQUENCE\_NUMBER 42
- PCOMM\_COMMAND 34
- PCOMM\_COMPRESSION\_GUID 39
- PCOMM\_CXTION 37
- PCOMM\_EOP 44
- PCOMM\_FILE\_OFFSET 41
- PCOMM\_FILE\_SIZE 41
- PCOMM\_FROM 36
- PCOMM\_GVSN 41
- PCOMM\_JOIN\_GUID 37
- PCOMM\_JOIN\_TIME 39
- PCOMM\_LAST\_JOIN\_TIME 38
- PCOMM\_PACKET 29
- PCOMM\_REMOTE\_CO 43
- PCOMM\_REPLICA 36
- PCOMM\_REPLICA\_VERSION\_GUID 39
- PCOMM\_TO 35
- PCOMM\_VVECTOR 38
- PDATA\_EXTENSION\_CHECKSUM 44
- PDATA\_EXTENSION\_PREFIX 44
- PDATA\_EXTENSION\_RETRY\_TIMEOUT 45
- PERFFRS interface
  - abstract data model 67
  - initialization 73
  - local events 77
  - message processing 73
  - overview 119
  - sequencing rules 73
  - timer events 74
  - timers 73



- Performance counters 60
- PGVSN 21
- PNTFRSAPI\_INFO 19
- Preconditions 17
- Prerequisites 17
- Product behavior 236
- Protocol Details
  - overview 67
- PSTAGE\_HEADER 45

## **R**

- References 13
  - informative 14
  - normative 13
- Relationship to other protocols 16

## **S**

- Schema elements - directory service 48
- Security
  - implementer considerations 232
  - overview 232
  - parameter index 232
- Sequencing rules
  - FRSAPI interface 73
  - FRSRPC interface (section 3.1.4 73, section 3.3.4 89)
  - PERFFRS interface 73
- STAGE\_HEADER structure 45
- Standards assignments 18
- Sysvol initial sync example 127

## **T**

- Timer events
  - FRSAPI interface (section 3.1.5 74, section 3.2.5 88)
  - FRSRPC interface (section 3.1.5 74, section 3.3.5 119)
  - PERFFRS interface 74
- Timers
  - FRSAPI interface (section 3.1.2 73, section 3.2.2 78)
  - FRSRPC interface (section 3.1.2 73, section 3.3.2 88)
  - PERFFRS interface 73
- Tracking changes 245
- Transport 19

## **V**

- Vendor-extensible fields 18
- Versioning 17