Remote File Copy Orchestration Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.

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

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

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

- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

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

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

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

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision History</th>
<th>Revision Class</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>07/13/2009</td>
<td>0.1</td>
<td>Major</td>
<td>Initial Availability</td>
</tr>
<tr>
<td>02/19/2010</td>
<td>1.0</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>03/31/2010</td>
<td>1.01</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>04/30/2010</td>
<td>1.02</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>06/07/2010</td>
<td>1.03</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>06/29/2010</td>
<td>1.04</td>
<td>Editorial</td>
<td>Changed language and formatting in the technical content.</td>
</tr>
<tr>
<td>07/23/2010</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>09/27/2010</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>11/15/2010</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>12/17/2010</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>03/18/2011</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>06/10/2011</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>01/20/2012</td>
<td>1.5</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>04/11/2012</td>
<td>1.5</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>07/16/2012</td>
<td>1.5</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
</tbody>
</table>

[MS-FSRFCO] — v20120630
Remote File Copy Orchestration Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
# Table of Contents

1 **Introduction** .............................................................................................................. 5  
   1.1 Glossary .................................................................................................................. 5  
   1.2 References .............................................................................................................. 5  
      1.2.1 Normative References ...................................................................................... 5  
      1.2.2 Informative References .................................................................................... 5  
   1.3 Protocol Overview (Synopsis) .................................................................................. 6  
   1.4 Relationship to Other Protocols ............................................................................ 7  
   1.5 Prerequisites/Preconditions ..................................................................................... 7  
   1.6 Applicability Statement .......................................................................................... 7  
   1.7 Versioning and Capability Negotiation .................................................................... 8  
   1.8 Vendor-Extensible Fields ......................................................................................... 8  
   1.9 Standards Assignments ............................................................................................ 8  

2 **Messages** ................................................................................................................... 9  
   2.1 Transport .................................................................................................................. 9  
   2.2 Common Data Types ................................................................................................ 9  

3 **Protocol Details** ........................................................................................................ 10  
   3.1 rtsearch::file_receiver Server Details ................................................................. 11  
      3.1.1 Abstract Data Model ....................................................................................... 11  
      3.1.2 Timers ............................................................................................................. 12  
      3.1.3 Initialization .................................................................................................... 12  
      3.1.4 Message Processing Events and Sequencing Rules ..................................... 12  
         3.1.4.1 get_data_dir ............................................................................................... 13  
         3.1.4.2 data_needed ............................................................................................... 13  
         3.1.4.3 remove_directory ...................................................................................... 14  
         3.1.4.4 remove_file ............................................................................................... 14  
         3.1.4.5 start ........................................................................................................... 14  
         3.1.4.6 close ......................................................................................................... 15  
         3.1.4.7 abort ........................................................................................................ 15  
      3.1.5 Timer Events ...................................................................................................... 15  
      3.1.6 Other Local Events ......................................................................................... 16  
   3.2 rtsearch::file_receiver Client Details ..................................................................... 16  
      3.2.1 Abstract Data Model ....................................................................................... 16  
      3.2.2 Timers ............................................................................................................. 16  
      3.2.3 Initialization .................................................................................................... 17  
      3.2.4 Message Processing Events and Sequencing Rules ..................................... 17  
         3.2.4.1 get_data_dir ............................................................................................... 17  
         3.2.4.2 data_needed ............................................................................................... 17  
         3.2.4.3 remove_directory ...................................................................................... 17  
         3.2.4.4 remove_file ............................................................................................... 17  
         3.2.4.5 start ........................................................................................................... 18  
         3.2.4.6 close ......................................................................................................... 18  
         3.2.4.7 abort ........................................................................................................ 18  
      3.2.5 Timer Events ...................................................................................................... 18  
      3.2.6 Other Local Events ......................................................................................... 18  

4 **Protocol Examples** .................................................................................................... 19  
   4.1 Calculate the Combined Subscription Value .......................................................... 19  
   4.2 Use the data_needed Method ................................................................................. 19
1 Introduction

This document specifies the Remote File Copy Orchestration Protocol, which is used to orchestrate copying of index related directories and files from a file sender to a file receiver.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

- fully qualified domain name (FQDN)
- Hypertext Transfer Protocol (HTTP)

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

- abstract object reference (AOR)
- backup indexer node
- base port
- client proxy
- FAST Search Interface Definition Language (FSIDL)
- host name
- HTTP POST
- index partition
- master indexer node
- name server
- query matching node
- search service application

The following terms are specific to this document:

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

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

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

[MS-FSICFG] Microsoft Corporation, "Indexer Configuration File Format".

1.3 Protocol Overview (Synopsis)

This protocol enables the configuration and administration of a subscription based framework to facilitate directory and file copying from a file sender, acting as protocol client, to a file receiver, acting as protocol server.

A file receiver registers a subscription with the file sender using the Index Publication and Activation Protocol, as described in [MS-FSIPA], or the Indexer Fault Tolerance Protocol, as described in [MS-FSIFT]. The subscription value is mapped to a set of files and directories on the file sender.

When a new file or directory structure that is subscribed to is available on the file sender, the protocol client asks the file receiver if a copy is needed. If a new copy is needed, the protocol client invokes a method on the protocol server, informing it to start a Remote File Copy Protocol server. The Remote File Copy Protocol is then used to copy the file or directory from the protocol client to the protocol server, as described in [MS-FSRFC]. The formats of the files copied are described in the Index Data Structures Specification, [MS-FSIXDS].

After the copy, the Remote File Copy Protocol server on the file receiver is stopped, and the copy process is done. This sequence is shown in the following figure, where the reference short names of the protocol documents specifying the methods used are shown in brackets.
1.4 Relationship to Other Protocols

This protocol uses the Middleware Protocol, an HTTP-based protocol, as described in [MS-FSMW]. The following figure shows the underlying messaging and transport stack that the protocol uses:

![Diagram showing the relationship to other protocols]

1.5 Prerequisites/Preconditions

The protocol client and protocol server are expected to know the location and connection information of the shared *name server*.

1.6 Applicability Statement

This protocol is applicable for configuring and administrating a framework for directory and file copying between a file sender and file receiver, using the Remote File Copy Protocol, as described in [MS-FSRFC].

---

[MS-FSRFCO] — v20120630
Remote File Copy Orchestration Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
1.7 Versioning and Capability Negotiation

Capability negotiation: The Middleware Protocol is connectionless, but the correct interface version is to be specified in every message passed using the Middleware Protocol. See section 3.1.3 for the version number.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.
2 Messages

2.1 Transport

The messages in this protocol MUST be sent as HTTP POST messages, as specified in [MS-FSMW], the Middleware Protocol.

2.2 Common Data Types

FAST Search Interface Definition Language (FSIDL) data types are encoded as specified in [MS-FSMW] section 2. The full FSIDL for this protocol is specified in section 6.1.
3 Protocol Details

This protocol enables the configuration and administration of a subscription based framework to facilitate directory and file copying from a file sender to a file receiver.

Multiple file receivers can register subscriptions with the file sender. In the **search service application** the file sender is the **master indexer node**, and file receivers are **backup indexer nodes** and **query matching nodes**. A subscription is mapped to a set of files and directories on the file sender.

A file receiver MUST register a subscription with the file sender using the **search_master::connect_receiver** message, as specified in [MS-FSIPA], or **column_master::connect_receiver** message, as specified in [MS-FSIFT]. See section 3.1.3 for details.

When a new version of a file or directory mapped to a subscription is available on the file sender, it MUST send a **data_needed** message, as specified in section 3.1.4.2, to all registered file receivers, determining if a file receiver subscribes to the file or directory, and if it needs a new copy.

If a new copy of a file or directory is needed, the file sender MUST send a **get_data_dir** message, as specified in section 3.1.4.1. The returned value is used to construct the name of the target file name or target directory name on the file receiver. If a directory is copied, the file sender MUST also construct a temporary directory name. Directory files MUST be copied to this temporary directory on the file receiver and renamed to the target directory name after the copy is done, making the directory copy an atomic event.

If a single file is being copied, the target file MUST be removed on the file receiver before the copying starts by sending a **remove_file** message, as specified in section 3.1.4.4. If a directory is being copied, both the target directory and temporary directory MUST be removed on the file receiver before the copying starts by sending **remove_directory** messages, as specified in section 3.1.4.3.

The file copying protocol used, the Remote File Copy Protocol, is specified in [MS-FSRFC]. The protocol client starts a Remote File Copy Protocol client on the file sender. A Remote File Copy Protocol server MUST be started on the file receiver by the **start** message, as specified in section 3.1.4.5. The internal formats of files copied are specified in [MS-FSIXDS]. If the **start** message is unable to start the Remote File Copy Protocol server, or a communication error is detected by the file sender, an **abort** message, as specified in section 3.1.4.7, MUST be sent.

When file copying is done, the Remote File Copy Protocol server MUST be closed by the **close** message, as specified in section 3.1.4.6.

A time sequence diagram of copying a single file is shown in the following figure, where the reference short names of the protocol documents specifying the methods used are shown in brackets. A query matching node, "file receiver 1", subscribes to subscription A and subscription B. A backup indexer node, "file receiver 2", subscribes to subscription B. When a file mapped to subscription A becomes available on the file sender, it is copied to "file receiver 1", but not "file receiver 2".
Figure 3: Copy sequence for a single file

3.1 rtsearch::file_receiver Server Details

A file receiver, acting as protocol server, receives files from a file sender, acting as protocol client, using the Remote File Copy Protocol, as specified in [MS-FSRCO].

3.1.1 Abstract Data Model

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

The protocol server MUST maintain the following state:
**subscriptions**: A state containing the combined subscription values for the file receiver, calculated by adding individual subscription values together, as specified in the **subscription mapping** state in section 3.2.1.

### 3.1.2 Timers

None.

### 3.1.3 Initialization

The file receiver, acting as protocol server, MUST store a combination of one or more subscription values, as specified in section 3.2.1, in the **subscriptions** state. A file receiver MUST use the subscription values as specified in the following table.

<table>
<thead>
<tr>
<th>Type of file receiver</th>
<th>Value</th>
<th>Subscription names</th>
</tr>
</thead>
<tbody>
<tr>
<td>Query matching node</td>
<td>3</td>
<td>index and dictionary</td>
</tr>
<tr>
<td>Backup indexer node</td>
<td>31</td>
<td>index, dictionary, state, generation, and counter</td>
</tr>
</tbody>
</table>

The file receiver, acting as protocol server, MUST use the `search_master::connect_receiver` method, as specified in [MS-FSIPA] section 3.3.4.2, or the `column_master::connect_receiver` method, as specified in [MS-FSIFT] section 3.1.4.6, to register a `file_receiver` server object with the file sender, acting as protocol client. The registering file receiver MUST be added to the **file receivers** state, as specified in section 3.2.1, by the file sender. The server object MUST be initialized using an **abstract object reference (AOR)**, as specified in [MS-FSMW], containing the following values:

- **object_id**: This MUST be an integer that is unique for each server object.
- **host**: A string specifying the **host name** of the server hosting the server object.
- **port**: This MUST be an integer that contains the port number of the server object on the protocol server. The value is **base port** plus 390.
- **interface_type**: This MUST be a string that contains the value "rtsearch::file_receiver".
- **interface_version**: This MUST be a string that contains the value "1.1".

### 3.1.4 Message Processing Events and Sequencing Rules

This interface includes the methods described in the following table.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_data_dir</td>
<td>This method returns the name of a directory structure on the file receiver that contains the target name of the file or directory to be copied.</td>
</tr>
<tr>
<td>data_needed</td>
<td>This method determines if the file receiver subscribes to a specific subscription value, and if a new copy of the file or directory is needed.</td>
</tr>
<tr>
<td>remove_directory</td>
<td>This method removes the specified directory.</td>
</tr>
<tr>
<td>remove_file</td>
<td>This method removes the specified file.</td>
</tr>
<tr>
<td>start</td>
<td>This method starts a Remote File Copy Protocol server on the file receiver, enabling the file sender to copy files and directories to the file receiver, as specified in [MS-</td>
</tr>
<tr>
<td>Method</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
</tr>
<tr>
<td>close</td>
<td>This method stops the Remote File Copy Protocol server on the file receiver.</td>
</tr>
<tr>
<td>abort</td>
<td>This method aborts the Remote File Copy Protocol server on the file receiver.</td>
</tr>
</tbody>
</table>

### 3.1.4.1 get_data_dir

The `get_data_dir` method returns the name of a directory structure on the file receiver that contains the target name of the file or directory copied, as follows.

```c
string get_data_dir(in long file_dir_idx)
```

**file_dir_idx:** An integer which MUST be ignored.

**Return values:** A string that MUST contain a valid directory name, which MUST be the directory name contained in `indexDir` in the configuration file specified in [MS-FSICFG].

**Exceptions:** No exceptions are raised beyond those raised by the underlying Middleware Protocol as specified in [MS-FSMW].

### 3.1.4.2 data_needed

The `data_needed` method determines if the file receiver subscribes to a specific subscription value, and if a new copy of the file or directory is needed, as follows.

```c
boolean data_needed(in long datatype,
                   in string stamp,
                   in string sub_dir,
                   in long file_dir_idx)
```

**datatype:** The subscription value to compare, which MUST be an integer with value as specified in the `subscription mapping` state, as specified in section 3.2.1.

**stamp:** A string that contains the stamp of the file or directory to be copied from the file sender. A stamp is an entity containing a 10 digit timestamp, as specified in section 3.2.1, indicating when an index related directory was created and consistent. A stamp file named "stamp.txt" is located in the root of that directory. When a directory is copied to a file receiver, the stamp file is copied by the file sender as part of the directory structure. When a single file is copied to a file receiver, the file sender copies a separate stamp file.

**sub_dir:** A string that contains a relative subdirectory name following the directory name returned by `get_data_dir`, as specified in section 3.1.4.1, where the stamp file is located.

**file_dir_idx:** An integer which MUST be ignored.

**Return values:** If the bitwise AND operator between the content of the `subscriptions` state and `datatype` returns a Boolean `true`, and the stamp file on the file receiver is missing or is different from `stamp`, a Boolean `true` MUST be returned. Otherwise, a Boolean `false` MUST be returned.

**Exceptions:** No exceptions are raised beyond those raised by the underlying Middleware Protocol as specified in [MS-FSMW].
3.1.4.3 remove_directory

The remove_directory method removes the specified directory, as follows.

boolean remove_directory(in string directory)

directory: A string that contains the absolute path of the directory to be removed.

Return values: A Boolean that MUST be true if directory was successfully removed or did not exist; otherwise, false.

Exceptions: No exceptions are raised beyond those raised by the underlying Middleware Protocol as specified in [MS-FSMW].

3.1.4.4 remove_file

The remove_file method removes the specified file, as follows.

boolean remove_file(in string file)

file: A string that contains the absolute path of the file to be removed.

Return values: A Boolean that MUST be true if file was successfully removed or did not exist; otherwise, false.

Exceptions: No exceptions are raised beyond those raised by the underlying Middleware Protocol as specified in [MS-FSMW].

3.1.4.5 start

When receiving the start method, the protocol server MUST try to initialize and start a Remote File Copy Protocol server on the file receiver, enabling the file sender to copy files and directories to the file receiver, as specified in [MS-FSRFC], as follows.

boolean start(in string hostname,
              in long port,
              in string dest_dir,
              in string inter_dir,
              in boolean file_receiver)

hostname: A string that contains the fully qualified domain name (FQDN) of the file receiver.

port: An integer that contains the port number of the file receiver, which MUST be an integer equal to or greater than zero and within the legal port range of the system.

dest_dir: A string that MUST contain the absolute directory name of the root directory of the target. Subsequent transfers using the Remote File Copy Protocol, as specified in [MS-FSRFC], MUST use paths relative to this target root directory.

inter_dir: A string that MUST contain the absolute directory name of the temporary root directory of the target, used when directories are copied. If a file is copied, it MUST be an empty string. Subsequent directory transfers using the Remote File Copy Protocol, as specified in [MS-FSRFC], MUST use paths relative to this temporary target root directory. Once the directory has been transferred, it is copied to the root directory specified in dest_dir.
**file_receiver**: A Boolean that MUST be true if a single file is to be copied, or false if a directory is to be copied.

**Return values**: A Boolean that MUST be true if the Remote File Copy Protocol server was started successfully on the file receiver; otherwise, false.

**Exceptions**: No exceptions are raised beyond those raised by the underlying Middleware Protocol as specified in [MS-FSMW].

This method MUST NOT be called unless the remove_directory method, as specified in section 3.1.4.3, or remove_file method, as specified in section 3.1.4.4, have been called and returned a Boolean true, depending on whether a directory or a single file is respectively to be copied.

### 3.1.4.6 close

When receiving the close method, the protocol server MUST try to stop the Remote File Copy Protocol server on the file receiver, as follows.

```java
boolean close(in long transfer_port)
```

**transfer_port**: An integer that contains the port number of the file receiver to close, which MUST be an integer equal to or greater than zero and within the legal port range of the system.

**Return values**: A Boolean that MUST be true if the Remote File Copy Protocol server was closed successfully; otherwise, false.

**Exceptions**: No exceptions are raised beyond those raised by the underlying Middleware Protocol as specified in [MS-FSMW].

This method MUST NOT stop the Remote File Copy Protocol server if a file or directory transfer is in progress, but waits until it is either finished, or has failed.

### 3.1.4.7 abort

When receiving the abort method, the protocol server MUST abort the Remote File Copy Protocol server on the file receiver, as follows.

```java
void abort(in long transfer_port)
```

**transfer_port**: An integer that contains the port number of the file receiver to abort, which MUST be an integer equal to or greater than zero and within the legal port range of the system.

**Return values**: None.

**Exceptions**: No exceptions are raised beyond those raised by the underlying Middleware protocol as specified in [MS-FSMW].

This method MUST stop the Remote File Copy Protocol server without waiting for an ongoing file or directory transfer to finish.

### 3.1.5 Timer Events

None.
3.1.6 Other Local Events

None.

3.2 rtsearch::file_receiver Client Details

A file sender, acting as protocol client, sends files to a file receiver, acting as protocol server, using
the Remote File Copy Protocol, as specified in [MS-FSRFC].

3.2.1 Abstract Data Model

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

The file sender, acting as protocol client, MUST maintain the following states:

file receivers: A state that contains information about file receivers which have registered with the
file sender, as described in section 3.1.3. This state MUST contain a client proxy and a port
number for each file receiver.

subscription mapping: A state that contains a mapping between files and directories, and a
subscription value, as specified in the following table. Multiple subscription values are combined by
adding them together. Files and directories mapping to a subscription value are copied by the file
sender to file receivers subscribing to the subscription values. Descriptions and format specifications
for the files copied are found in [MS-FSIXDS].

All files and directories in the following table are local to the implementation-specific index directory,
where PP indicates an index partition identifier, TTTT indicates a 19-digit timestamp, TT indicates
a 10 digit timestamp, FQDN indicates the FQDN of the node that generated the directory, and NN
indicates an implementation specific counter. A timestamp is a numeric representation of the
number of seconds or nanoseconds since 1970-01-01T00:00:00 UTC, specified with 10 digits or 19
digits, respectively. Possible values are described in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Bit value</th>
<th>Name</th>
<th>Subscription content</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00001</td>
<td>index</td>
<td>PP\index_TTTT\index_data</td>
<td>directory</td>
</tr>
<tr>
<td>2</td>
<td>00010</td>
<td>dictionary</td>
<td>FQDN.normalized.TT</td>
<td>directory</td>
</tr>
<tr>
<td>4</td>
<td>00100</td>
<td>state</td>
<td>state</td>
<td>directory</td>
</tr>
<tr>
<td>8</td>
<td>01000</td>
<td>generation</td>
<td>PP\index_TTTT\NN\stamp.txt</td>
<td>file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PP\index_TTTT\NN\urlmap_sorted.txt</td>
<td>file</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PP\index_TTTT\NN\exclusionlisted.txt</td>
<td>file</td>
</tr>
<tr>
<td>16</td>
<td>10000</td>
<td>counter</td>
<td>PP\activated_counter</td>
<td>directory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PP\activated_indexed_counter</td>
<td>directory</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PP\index_counter</td>
<td>directory</td>
</tr>
</tbody>
</table>

3.2.2 Timers

None.
3.2.3 Initialization

The file sender, acting as protocol client, MUST use the file receivers state, as specified in section 3.2.1, to communicate with file receivers.

3.2.4 Message Processing Events and Sequencing Rules

This interface includes the methods described in the following table.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get_data_dir</td>
<td>This method returns the name of a directory structure on the file receiver that contains the target name of the file or directory to be copied.</td>
</tr>
<tr>
<td>data_needed</td>
<td>This method determines if the file receiver subscribes to a specific subscription value, and if a new copy of the file or directory is needed.</td>
</tr>
<tr>
<td>remove_directory</td>
<td>This method removes the specified directory.</td>
</tr>
<tr>
<td>remove_file</td>
<td>This method removes the specified file.</td>
</tr>
<tr>
<td>start</td>
<td>This method starts a Remote File Copy Protocol server on the file receiver, enabling the file sender to copy files and directories to the file receiver, as specified in [MS-FSRFC].</td>
</tr>
<tr>
<td>close</td>
<td>This method stops the Remote File Copy Protocol server on the file receiver.</td>
</tr>
<tr>
<td>abort</td>
<td>This method aborts the Remote File Copy Protocol server on the file receiver.</td>
</tr>
</tbody>
</table>

3.2.4.1 get_data_dir

The get_data_dir method is specified in section 3.1.4.1. The protocol client MUST use the returned value to construct the target file or directory name of the file or directory to be copied.

3.2.4.2 data_needed

The data_needed method is specified in section 3.1.4.2. The protocol client MUST end the copy sequence if the return value is not a Boolean true.

3.2.4.3 remove_directory

The remove_directory method is specified in section 3.1.4.3. If a directory is to be copied, the protocol client MUST call this method for both the target directory and the temporary directory on the protocol server, and only proceed to call the start method, as specified in section 3.2.4.5, if the remove_directory method returns a Boolean true. This to ensure that the target directory on the protocol server is empty before copying starts.

3.2.4.4 remove_file

The remove_file method is specified in section 3.1.4.4. If a single file is to be copied, the protocol client MUST call this method for the target file on the protocol server, and only proceed to call the start method, as specified in section 3.2.4.5, if the remove_file method returns a Boolean true. This to ensure that the target file on the protocol server does not exist before copying starts.
3.2.4.5 start

The **start** method is specified in section 3.1.4.5. This method MUST NOT be called unless the **remove_directory** method, as specified in section 3.2.4.3, or the **remove_file** method, as specified in section 3.2.4.4, has been called and returns a Boolean **true**, depending on whether a directory or a single file is respectively to be copied.

This method MUST store the **port** parameter in the **file receivers** state.

3.2.4.6 close

The **close** method is specified in section 3.1.4.6. This method MUST be called after the file or directory structure has been copied successfully, as specified in [MS-FSRFC].

3.2.4.7 abort

The **abort** method is specified in section 3.1.4.7. This method MUST be called if the file or directory structure copy failed, as specified in [MS-FSRFC].

3.2.5 Timer Events

None.

3.2.6 Other Local Events

None.
4 Protocol Examples

4.1 Calculate the Combined Subscription Value

This example describes how to calculate the combined subscription value of the subscriptions state on a file receiver, which subscribes to three entries from the subscription mapping state, as specified in section 3.2.1; index, state, and counter.

The calculation of the combined subscription value is done by adding the individual subscription values, and is shown in the following table.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Value</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>index</td>
<td>1</td>
<td>00001</td>
</tr>
<tr>
<td>state</td>
<td>4</td>
<td>00100</td>
</tr>
<tr>
<td>counter</td>
<td>16</td>
<td>10000</td>
</tr>
<tr>
<td>subscription</td>
<td>21</td>
<td>10101</td>
</tr>
</tbody>
</table>

When the files mapping to the state entry have changed, the file receiver sends a data_needed message, as specified in section 3.1.4.2, with the value 4 as subscription parameter. The file receiver uses the bitwise AND operator between the received parameter and the stored subscriptions state, as shown in the following table.

<table>
<thead>
<tr>
<th>Entry</th>
<th>Value</th>
<th>Binary</th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter</td>
<td>4</td>
<td>00100</td>
</tr>
<tr>
<td>subscription</td>
<td>21</td>
<td>10101</td>
</tr>
<tr>
<td>subscribes</td>
<td>1</td>
<td>00100</td>
</tr>
</tbody>
</table>

In this example, the calculated value is 1, and the data_needed method will return a Boolean true.

4.2 Use the data_needed Method

This example describes how to use the data_needed method of the file_receiver interface, as specified in section 3.1.4.2, enabling a file sender to ask a query matching node file receiver subscribing to a dictionary, as specified in section 3.2.1, if a new version of the dictionary needs to be copied.

First the file receiver creates a server object implementing the file_receiver interface, and registers it with the file sender, which adds it to the file receivers state. The method for registering is search_master::connect_receiver, as specified in [MS-FSIPA] section 3.3.4.2.

The file sender is now ready to call the data_needed method on the file_receiver client proxy.

4.2.1 is_data_needed Code

4.2.1.1 File-Receiver Initialization

```plaintext
SET subscription_type TO "2"
```
SET server_object_instance TO INSTANCE OF file_receiver SERVER OBJECT
SET server_object_host TO "myserver.mydomain.com"
SET server_object_port TO "1234"
SET server_object_interface_type TO "rtsearch::file_receiver"
SET server_object_interface_version TO "1.1"
SET server_object_aor TO server_object_host, server_object_port, server_object_interface_type, server_object_interface_version
CALL search_master.connect_receiver WITH server_object_instance AND server_object_aor

4.2.1.2 File-Sender Initialization
ADD server_object_aor TO file_receivers_state

4.2.1.3 File-Sender Message
FIND file_receiver_client_proxy FROM file_receivers_state
SET subscription_type_to_check TO "2"
SET stamp_file_value_to_check TO "1255960136"
CALL file_receiver_client_proxy.data_needed WITH subscription_type_to_check AND stamp_file_value_to_check RETURNING is_data_needed

4.2.1.4 File-Receiver Response
RETURN BOOLEAN(subscription_type AND subscription_type_to_check)AND stamp_file_value_to_check NOT EQUAL stamp_file_local_value
5 Security

5.1 Security Considerations for Implementers

Security is resolved in the Middleware Protocol, as described in [MS-FSMW].

5.2 Index of Security Parameters

None.
6 Appendix A: Full FSIDL

For ease of implementation the full FSIDL used in this protocol is provided in the following section.

6.1 FSIDL

module interfaces {
    module rtsearch {
        interface file_receiver {
            #pragma version file_receiver 1.1
            string get_data_dir(in long file_dir_idx);
            boolean data_needed(in long datatype,
                                 in string stamp,
                                 in string sub_dir,
                                 in long file_dir_idx);
            boolean remove_directory(in string directory);
            boolean remove_file(in string file);
            boolean start(in string hostname,
                           in long port,
                           in string dest_dir,
                           in string inter_dir,
                           in boolean file_receiver);
            boolean close(in long transfer_port);
            void abort(in long transfer_port);
        }
    }
};
}
7 Appendix B: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® FAST™ Search Server 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.
8 Change Tracking

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

A

abort method (section 3.1.4.7 15, section 3.2.4.7 18)
Abstract data model
  client 16
  server 11
Applicability 7

C

Calculate the combined subscription value example 19
Capability negotiation 8
Change tracking 24
Client
  abort method 18
  abstract data model 16
  close method 18
  data_needed method 17
  get_data_dir method 17
  initialization 17
  local events 18
  message processing 17
  overview (section 3 10, section 3.2 16)
  remove_directory method 17
  remove_file method 17
  rtsearch::file_receiver interface 16
  sequencing rules 17
  start method 18
  timer events 18
  timers 16
close method (section 3.1.4.6 15, section 3.2.4.6 18)
Common data types 9

D

Data model - abstract
  client 16
  server 11
Data types
  common - overview 9
  data_needed method (section 3.1.4.2 13, section 3.2.4.2 17)

E

Events
  local - client 18
  local - server 16
  timer - client 18
  timer - server 15
Examples
  calculate the combined subscription value 19
  use the data_needed method 19

F

Fields - vendor-extensible 8
FSIDL 22
Full FSIDL 22

G

get_data_dir method (section 3.1.4.1 13, section 3.2.4.1 17)
Glossary 5

I

Implementer - security considerations 21
Index of security parameters 21
Informative references 6
Initialization
  client 17
  server 12
Interfaces - client
  rtsearch::file_receiver 16
Interfaces - server
  rtsearch::file_receiver 11
Introduction 5

L

Local events
  client 18
  server 16

M

Message processing
  client 17
  server 12
Messages
  common data types 9
  transport 9
Methods
  abort (section 3.1.4.7 15, section 3.2.4.7 18)
  close (section 3.1.4.6 15, section 3.2.4.6 18)
  data_needed (section 3.1.4.2 13, section 3.2.4.2 17)
  get_data_dir (section 3.1.4.1 13, section 3.2.4.1 17)
  remove_directory (section 3.1.4.3 14, section 3.2.4.3 17)
  remove_file (section 3.1.4.4 14, section 3.2.4.4 17)
  start (section 3.1.4.5 14, section 3.2.4.5 18)

N

Normative references 5

O

Overview (synopsis) 6