[MS-FSRFC]:
Remote File Copy Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Copyright © 2010 Microsoft Corporation.

Revision Summary

Date	Revision History	Revision Class	Comments
02/19/2010	1.0	Major	Initial Availability
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.

Copyright © 2010 Microsoft Corporation.

Table of Contents

Copyright © 2010 Microsoft Corporation.

Copyright © 2010 Microsoft Corporation.

1 Introduction

This document specifies the Remote File Copy Protocol used to copy index related files from a file sender to a file receiver.

1.1 Glossary

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

big-endian

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

backup indexer node
master indexer node
mebibyte (MiB)
query matching node
query processing
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

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-FSIXDS] Microsoft Corporation, "Index Data Structures", February 2010.

[MS-FSRFCO] Microsoft Corporation, "Remote File Copy Orchestration Protocol Specification", February 2010.

[RFC20] Cerf, V., "ASCII Format for Network Interchange", RFC 20, October 1969, http://www.ietf.org/rfc/rfc20.txt

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

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981, http://www.ietf.org/rfc/rfc0793.txt

1.2.2 Informative References

[MS-GLOS] Microsoft Corporation, "Windows Protocols Master Glossary", March 2007.

[MS-OFCGLOS] Microsoft Corporation, "Microsoft Office Master Glossary", June 2008.

Copyright © 2010 Microsoft Corporation.

1.3 Protocol Overview (Synopsis)

This protocol enables files and directory structures to be copied from a file sender, the protocol client, to a file receiver, the protocol server. This is accomplished by sending a set of messages in a predefined sequence between the protocol client and the protocol server. The protocol client determines the order in which each file within the directory structure is copied.

Files larger than 5,242,880 bytes, that is 5 **mebibytes** or megabinary bytes (MiB), are split into one or more chunks of 5 MiB plus a trailing chunk of a size less than or equal to 5 MiB, and sent sequentially to the file receiver.

1.4 Relationship to Other Protocols

This protocol relies on TCP/IP, as described in [RFC793], as its transport protocol for passing the messages between the protocol client and the protocol server.

The following figure shows the underlying messaging and transport stack that the protocol uses:

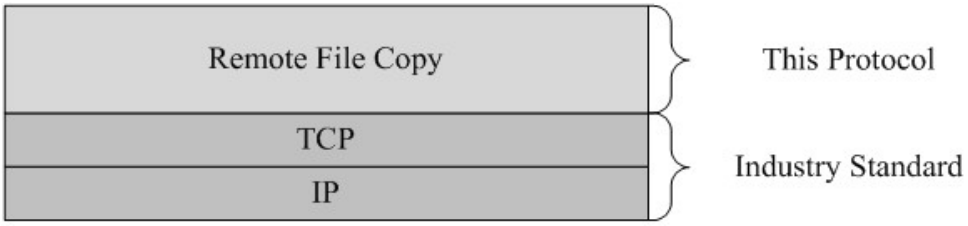


Figure 1: This protocol in relation to other protocols

No other protocol depends directly on this protocol.

1.5 Prerequisites/Preconditions

A TCP/IP connection, as described in [RFC793], between the protocol client and the protocol server needs to exist before this protocol can be used. The protocol client and protocol server are initialized to either transfer a single file or a directory structure. The Remote File Copy Orchestrating protocol, as described in [MS-FSRFCO], is used for this purpose.

The protocol client and protocol server are expected to know a shared signature string, as described in section 2.2.2.

1.6 Applicability Statement

This protocol is applicable for copying directories and single files between **search service application** nodes, typically between a **master indexer node**, that is, the file sender, and file receivers including **backup indexer nodes**, **query matching nodes**, and **query processing** nodes.

This protocol does not support multiple endpoints. The TCP/IP connection, as described in [RFC793], between the file sender and file receiver is closed after either a single file or a directory structure has been copied, as described in [MS-FSRFCO] section 3.1.4.6

Copyright © 2010 Microsoft Corporation.

1.7 Versioning and Capability Negotiation

Capability Negotiation: The protocol client and protocol server are expected to know a shared signature string, as described in section 2.2.2, and be configured to copy either a single file or a directory structure.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

Copyright © 2010 Microsoft Corporation.

2 Messages

2.1 Transport

Messages MUST be transported by using TCP/IP, as specified in [RFC793].

2.2 Message Syntax

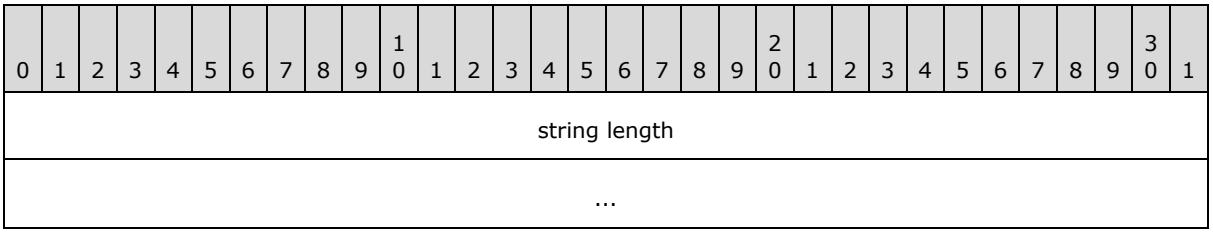
The type of an incoming message MUST be determined by the protocol server based on the sequence of the messages sent from the protocol client and the content of the **copy type** state, as specified in section 3.1.1.

The data types used in the following sections MUST be as specified in the following table.

Type	Specification
int64	A 64 bit signed integer with **big-endian** data representation.
string	A string encoded in ASCII, as specified in [RFC20].
byte	A signed byte.

2.2.1 string length

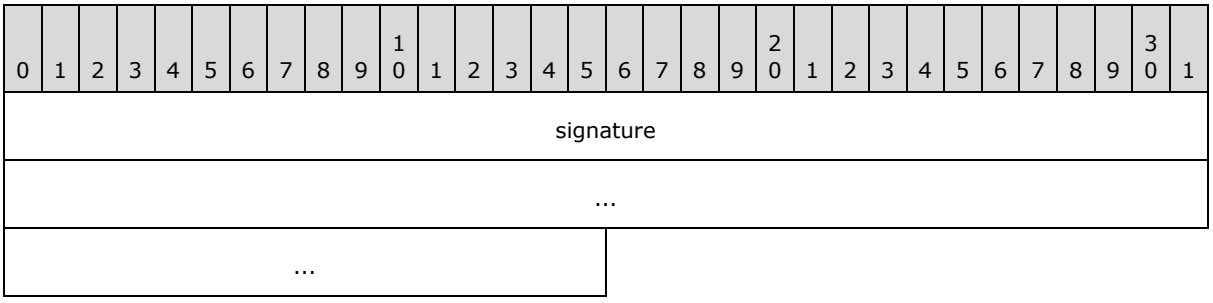
The **string length** message contains the length of a string.



string length (8 bytes): An **int64** that MUST be equal to or greater than zero.

2.2.2 signature string

The **signature string** message contains the unique signature string for the socket protocol. The signature string MUST be known to both the protocol client and the protocol server.

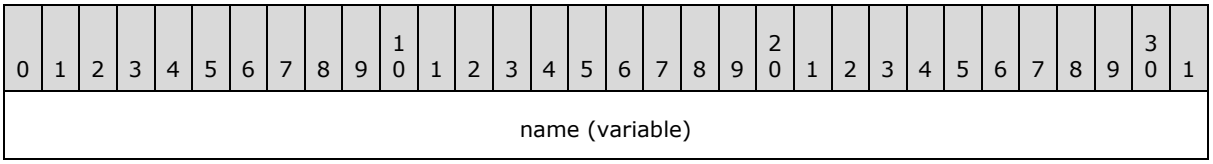


signature (10 bytes): A **string** that MUST contain the signature string, "RTS_FT_V_9".

Copyright © 2010 Microsoft Corporation.

2.2.3 name string

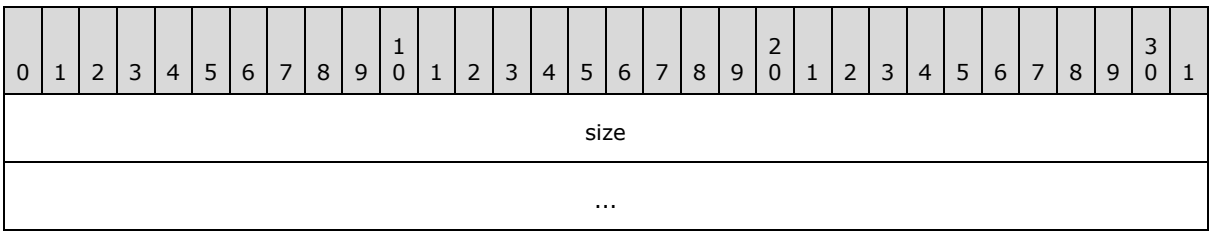
The **name string** message contains a relative file or path name to the file or directory being copied. The length is variable, and the content can be empty. The file or path name MUST be relative to the content of the **base directory** state.



name: A **string** that MUST be empty or a valid relative file system directory or file name.

2.2.4 size

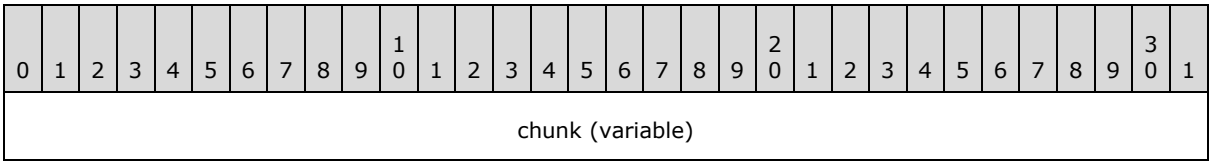
The **size** message contains the size in bytes of the directory or file to be transferred.



size (8 bytes): An **int64** that MUST be equal to or greater than zero.

2.2.5 chunk

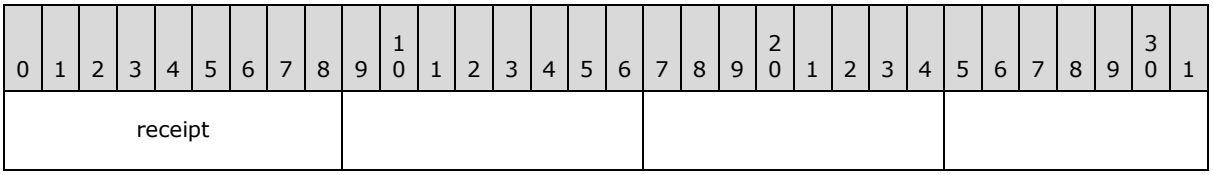
The **chunk** message contains the data or part of the data of the file to be transferred.



chunk: A **byte** sequence which MUST have length between 0 and 5 MiB.

2.2.6 receipt

The **receipt** message contains status information from the file receiver. If the content is 1, the file receiver detected no errors, and no special actions are required. If the content is 0, the file receiver detected errors, and the file sender MUST close down the TCP/IP socket and stop the file sending sequence.



receipt (1 byte): A **byte** that MUST be either 0 or 1.

Copyright © 2010 Microsoft Corporation.

3 Protocol Details

3.1 Common Details

The first sequence of messages MUST be a signature message sequence, as specified in section 3.1.5.1, used by the protocol server to verify and acknowledge the protocol client.

If the content of the **copy type** state is "file", the second sequence of messages MUST be a file information message sequence, as specified in section 3.1.5.3, followed by a file data message sequence, as specified in section 3.1.5.4.

If the content of the **copy type** state is "directory", the second sequence of messages MUST be a directory information message sequence, as specified in section 3.1.5.2, followed by a file information message sequence and a file data message sequence for each file within the directory. The protocol client determines the order in which each file within the directory structure is copied.

The content of a file larger than 5 MiB MUST be sent sequentially within the file data message sequence in one or more chunks of 5 MiB plus a trailing chunk of a size less than or equal to 5 MiB.

The last message sent MUST be a **receipt** message, as specified in section 2.2.6, sent from the protocol server to the protocol client. If the content of the **copy type** state is "file", the protocol server sends a second **receipt** message, which MUST be ignored by the protocol client.

The internal format of files being copied is specified in [MS-FSIXDS]. The copy sequence is shown in the following figure.

Copyright © 2010 Microsoft Corporation.

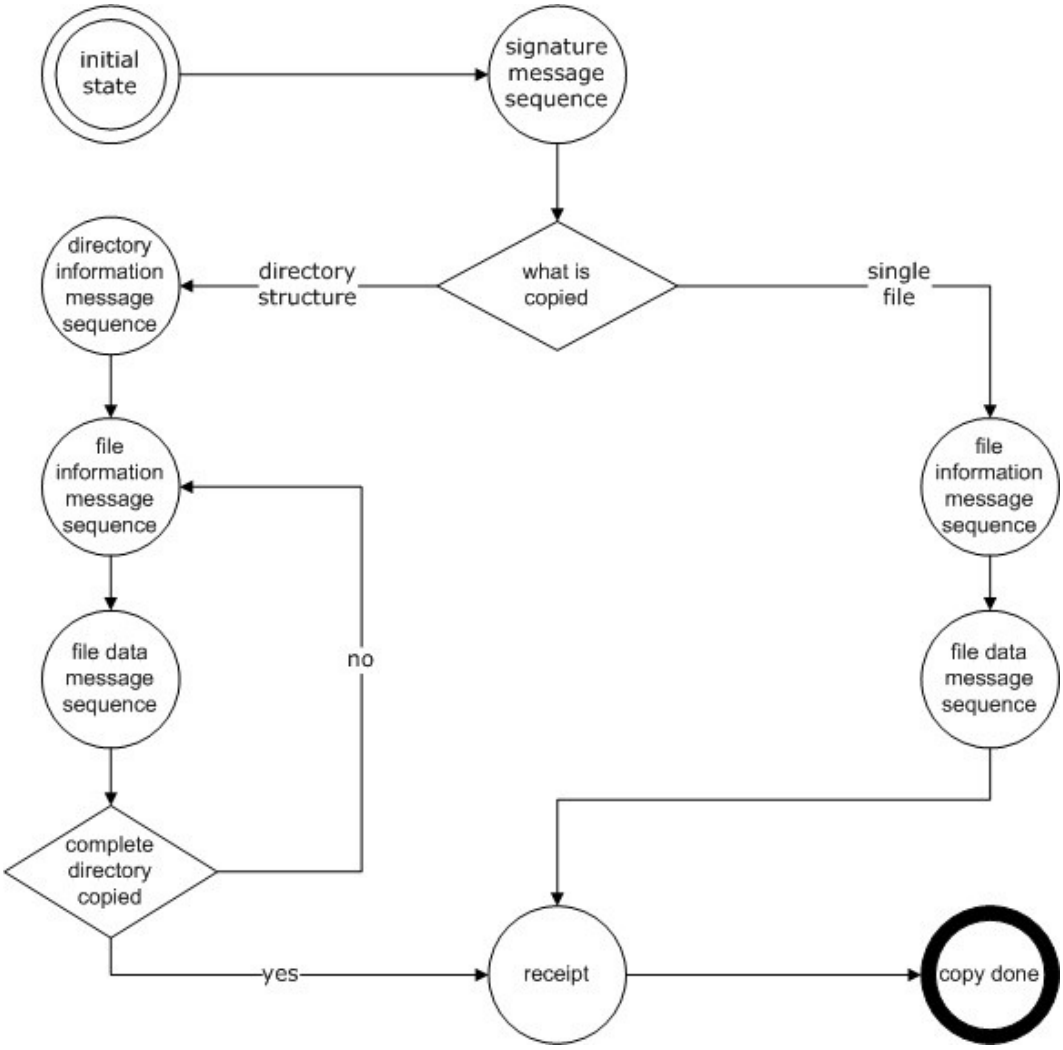


Figure 2: Data flow of directory structure and single file copy

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 client and protocol server MUST maintain the following states:

copy type: A state that MUST be "file" if a single file is being copied, or "directory" if a directory structure is being copied. This state MUST be set by the **start** method specified in [MS-FSRFCO] section 3.1.4.5.

base directory: A state containing the base directory name of the file or directory being copied. This state MUST be set by the **start** method specified in [MS-FSRFCO] section 3.1.4.5.

Copyright © 2010 Microsoft Corporation.

3.1.2 Timers

None.

3.1.3 Initialization

A TCP/IP connection, as specified in [RFC793], between the protocol client and the protocol server MUST exist before this protocol can be used. The protocol client and protocol server MUST be configured to either transfer a single file or a directory structure, and the **copy type** state updated to reflect this. The **base directory** state on the protocol server MUST be configured. The Remote File Copy Orchestrating protocol, as specified in [MS-FSRFCO], is used for this purpose.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

This section specifies message sequences sent and received by the protocol client and the protocol server.

3.1.5.1 Signature message sequence

The signature message sequence contains a signature string, used by the protocol client to verify the protocol server. It MUST adhere to the following sequence.

1. A **string length** message, as specified in section 2.2.1, containing the length of the signature string, as specified in section 2.2.2.
2. A **signature string** message, as specified in section 2.2.2.
3. A **receipt** message, as specified in section 2.2.6, sent from the protocol server to the protocol client.
4. Done.

The signature message sequence is shown in the following figure.

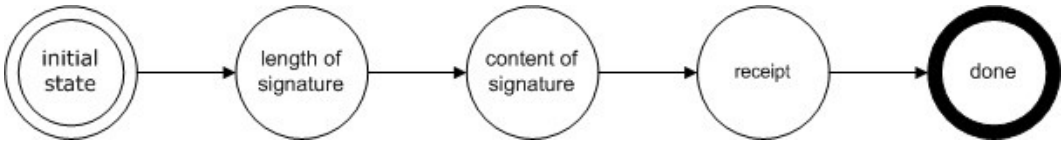


Figure 3: Signature message sequence

3.1.5.2 Directory information message sequence

The directory information message sequence contains information about a directory being copied, which MUST be relative to the content of the **base directory** state. It MUST be followed by a file information message sequence, as specified in section 3.1.5.3, for each file within the directory structure. It MUST adhere to the following sequence.

1. A **string length** message, as specified in section 2.2.1, containing the length of the relative directory name.
2. If the length of the directory name is zero, go to step 4, otherwise go to step 3.

Copyright © 2010 Microsoft Corporation.

3. A **name string** message, as specified in section 2.2.3, containing the relative directory name to be copied, which can be empty. The directory name MUST be relative to the content of the **base directory** state.

4. A directory **size** message, as specified in section 2.2.4, containing the combined size, in bytes, of all the files within the directory structure.

5. A number **size** message, as specified in section 2.2.4, containing the number of files within the directory structure.

6. Done.

The directory information message sequence is shown in the following figure.

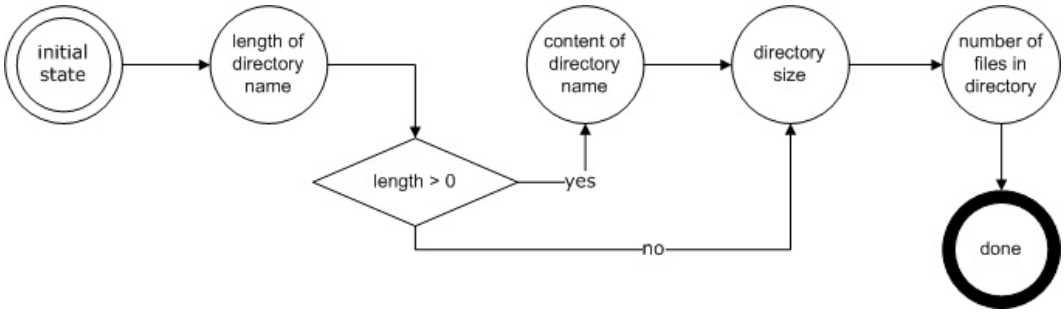


Figure 4: Directory information message sequence

3.1.5.3 File information message sequence

The file information message sequence contains information about a file being copied. It MUST adhere to the following sequence.

1. A **string length** message, as specified in section 2.2.1, containing the length of the file name.

2. If the length of the file name is zero, go to step 4, otherwise go to step 3.

3. A **name string** message, as specified in section 2.2.3, containing the relative file name to be copied. The file name MUST be relative to the content of the **base directory** state.

4. A **size** message, as specified in section 2.2.4, containing the file size, in bytes.

5. Done.

The file information message sequence is shown in the following figure.

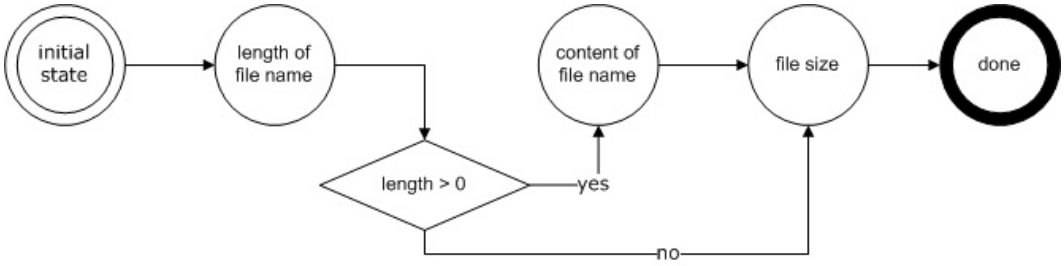


Figure 5: File information message sequence

Copyright © 2010 Microsoft Corporation.

3.1.5.4 File data message sequence

The file data message sequence contains the content of a file being copied. Files larger than 5 MiB are sent in sequential chunks. It MUST adhere to the following sequence.

1. If the size of the file data not yet sent is smaller than or equal to 5 MiB, go to step 4.
2. A **chunk** message, as specified in section 2.2.5, containing the first 5 MiB of the file not yet sent.
3. Go to step 1.
4. If the size of the file data not yet sent is larger than zero, go to step 5, otherwise go to step 6.
5. A **chunk** message containing the not yet sent file data.
6. Done.

The file data message sequence is shown in the following figure.

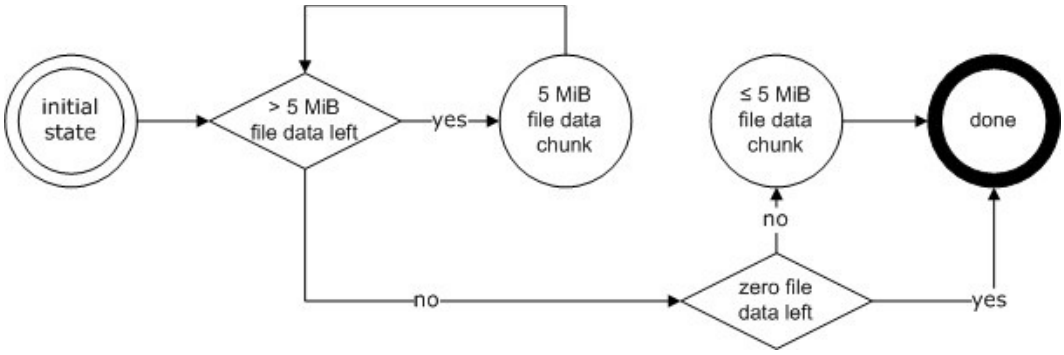


Figure 6: File data message sequence

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Server Details

A file receiver acts as protocol server, receiving files from a file sender, the protocol client.

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 receiver MUST maintain the common states as specified in section 3.1.1.

The file receiver MUST maintain the following states for a directory being copied:

Copyright © 2010 Microsoft Corporation.

directory size: A state containing the combined size, in bytes, of all the files within the directory structure.

number of files: A state containing the number of files within the directory structure.

The file receiver MUST maintain the following states for a file being copied, either as a single file or as a file within a directory structure:

file name: A state containing the name of the file received from the file sender.

file size: A state containing the size, in bytes, of the file received from the file sender.

3.2.2 Timers

The **socket timeout** timer measures the time spent on a write or read to the TCP/IP socket, as specified in [RFC793], between the protocol client and protocol server. The implementation-specific maximum value is approximately 10 minutes.

3.2.3 Initialization

See section 3.1.3.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

The type of each message received by the protocol server is based on a predetermined receiving sequence, as specified in section 3.1.

3.2.5.1 Receiving a signature message sequence

The signature message sequence is specified in section 3.1.5.1.

The protocol server MUST send a **receipt** message with the value 1, as specified in section 2.2.6, if the received signature string is "RTS_FT_V_9", as specified in section 2.2.2.

The protocol server MUST send a **receipt** message with the value 0, as specified in section 2.2.6, if the received signature string is not "RTS_FT_V_9", as specified in section 2.2.2. If so, it MUST also stop the file copying process.

3.2.5.2 Receiving a directory information message sequence

The directory information message sequence is specified in section 3.1.5.2.

The protocol server MUST store the content of the directory **size** message in the **directory size** state, and the content of the number **size** message in the **number of files** state.

When a directory is copied, the protocol client determines the order in which each file within the directory structure is copied.

3.2.5.3 Receiving a file information message sequence

The file information message sequence is specified in section 3.1.5.3.

Copyright © 2010 Microsoft Corporation.

The protocol server MUST store the content of the **name string** message in the **file name** state, and the content of the **size** message in the **file size** state

3.2.5.4 Receiving a file data message sequence

The file data message sequence is specified in section 3.1.5.4.

If the **file size** state is greater than zero, the protocol server MUST store the received file data to a file with the name equal to the relative **file name** state within the base directory contained in the **base directory** state.

If a directory structure is being sent, the protocol server MUST continue to receive file information message sequences and file data message sequences until the number of files received equals the **number of files** state.

3.2.5.5 Sending a receipt message

The receipt message is specified in section 2.2.6.

If the content of the **copy type** state is "file", the protocol server MUST send a **receipt** message with the value 0 if the size of the received file differs from the content of the **file size** state, otherwise a **receipt** message with the value 1. The protocol server sends a second **receipt** message with the value 1, which MUST be ignored by the protocol client.

If the content of the **copy type** state is "directory", the protocol MUST send a **receipt** message with the value 0 if the size of the received directory differs from the content of the **directory size** state, otherwise a **receipt** message with the value 1.

3.2.6 Timer Events

The **socket timeout** event stops the file copying and closes down the TCP/IP connection.

3.2.7 Other Local Events

None.

3.3 Client Details

A file sender acts as protocol client, sending a single file or a directory structure to a file receiver, the protocol server.

3.3.1 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 MUST maintain the common states as specified in section 3.1.1 and the following state.

remaining file content: A state containing the remaining content of a file being sent.

Copyright © 2010 Microsoft Corporation.

3.3.2 Timers

The **socket timeout** timer measures the time spent on a write or read to the TCP/IP socket between the protocol client and protocol server. The implementation-specific maximum value is approximately 10 minutes.

3.3.3 Initialization

See section 3.1.3.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

The type of each message sent by the protocol client is based on a predetermined sending sequence, as specified in section 3.1.

3.3.5.1 Sending a signature message sequence

The signature message sequence is specified in section 3.1.5.1.

If the protocol server's **receipt** message is 0, the protocol client MUST stop the file transfer process.

3.3.5.2 Sending a directory information message sequence

The directory information message sequence is specified in section 3.1.5.2.

When a directory is copied, the protocol client determines the order in which each file within the directory structure is copied.

3.3.5.3 Sending a file information message sequence

The file information message sequence is specified in section 3.1.5.3.

When a directory is copied, the protocol client determines the order in which each file within the directory structure is copied.

3.3.5.4 Sending a file data message sequence

The file data message sequence is specified in section 3.1.5.4.

If a directory structure is being sent, the protocol server MUST continue to send file information message sequences and file data message sequences until all files within the directory structure have been sent.

3.3.5.5 Receiving a receipt message

The receipt message is specified in section 2.2.6.

If the content of the **copy type** state is "file", the protocol client MUST read a **receipt** message from the protocol server after the file data message sequence has been sent. The protocol server sends a second **receipt** message, which MUST be ignored by the protocol client.

Copyright © 2010 Microsoft Corporation.

If the content of the **copy type** state is "directory", the protocol client MUST read a **receipt** message from the protocol server after the file data message sequence of the last file has been sent.

3.3.6 Timer Events

The **socket timeout** event stops the file copying and closes down the TCP/IP connection.

3.3.7 Other Local Events

None

Copyright © 2010 Microsoft Corporation.

4 Protocol Examples

4.1 Copy a single file

The example in this section copies a single file named "toobad", of which the content is "abc".

Data	Description
Sent from protocol client:	
00 00 00 00 00 00 00 0A	Length of signature string, 10.
52 54 53 5F 46 54 5F 56 5F 39	Signature string, "RTS_FT_V_9".
Sent from protocol server:	
01	Receipt with success value 1.
Sent from protocol client:	
00 00 00 00 00 00 00 06	Length of file name, 6.
74 6F 6F 62 61 64	File name string, "toobad".
00 00 00 00 00 00 00 03	Length of file, 3 bytes.
61 62 63	File data, "abc". Only one chunk sent, as length is less than 5 MiB.
Sent from protocol server:	
01	Receipt with success value 1.
01	Receipt with success value 1 (message ignored by protocol client)

4.2 Copy a directory structure

The example in this section copies a directory named "toobad" which contains the two files "abc" and "def", and a directory "too". The directory "too" contains the file "ghi". The content of all files is "test".

```
toobad – abc
         def
         too – ghi
```

Data	Description
Sent from protocol client:	
00 00 00 00 00 00 00 0A	Length of signature string, 10.
52 54 53 5F 46 54 5F 56 5F 39	Signature string, "RTS_FT_V_9".
Sent from protocol server:	
01	Receipt with success value 1.
Sent from protocol client:	

Copyright © 2010 Microsoft Corporation.

Data	Description
00 00 00 00 00 00 00 06	Length of directory name, 6.
74 6F 6F 62 61 64	Directory name string, "toobad"
00 00 00 00 00 00 00 0C	Size of files in directory, 12 bytes
00 00 00 00 00 00 00 03	Number of files in directory, 3
00 00 00 00 00 00 00 0A	Length of file name, 10.
74 6F 6F 62 61 64 5C 61 62 63	File name string, "toobad\abc".
00 00 00 00 00 00 00 04	Length of file, 4.
74 65 73 74	File data, "test". Only one chunk sent, as length is less than 5 MiB.
00 00 00 00 00 00 00 0A	Length of file name, 10.
74 6F 6F 62 61 64 5C 64 65 66	File name string, "toobad\def".
00 00 00 00 00 00 00 04	Length of file, 4.
74 65 73 74	File data, "test". Only one chunk sent, as length is less than 5 MiB.
00 00 00 00 00 00 00 0E	Length of file name, 14.
74 6F 6F 62 61 64 5C 74 6F 6F 5C 67 68 69	File name string, "toobad\too\ghi".
00 00 00 00 00 00 00 04	Length of file, 4.
74 65 73 74	File data, "test". Only one chunk sent, as length is less than 5 MiB.
Sent from protocol server:	
01	Receipt with success value 1.

Copyright © 2010 Microsoft Corporation.

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

Copyright © 2010 Microsoft Corporation.

6 Appendix A: 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.

Copyright © 2010 Microsoft Corporation.

7 Change Tracking

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

Copyright © 2010 Microsoft Corporation.

8 Index

Copyright © 2010 Microsoft Corporation.

Copyright © 2010 Microsoft Corporation.