[MS-FSCMT]:
Crawler Multinode Transport Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
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.

Table of Contents

1 Introduction

This document specifies the Crawler Multinode Transport Protocol, which defines the messages exchanged in a multinode **Web crawler** environment.

1.1 Glossary

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

big-endian
Domain Name System (DNS)
Hypertext Transfer Protocol (HTTP)
Transmission Control Protocol (TCP)
UTF-8
XML

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

adaptive crawl
connection
crawl collection
crawl queue
crawl refresh cycle
crawl routing
crawl rule
crawl site
duplicate server
file
forward link
HTML (HyperText Markup Language)
multinode scheduler
node identifier
node scheduler
owner URI
RSS channel
start URI
URI (Uniform Resource Identifier)
vector clock
Web crawler
XML attribute
XML element

The following terms are specific to this document:

duplicate: A search result that is identified as having identical or near identical content.

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.

[HTML] World Wide Web Consortium, "HTML 4.01 Specification", December 1999, http://www.w3.org/TR/html4/

[IEEE754] Institute of Electrical and Electronics Engineers, "Standard for Binary Floating-Point Arithmetic", IEEE 754-1985, October 1985, http://ieeexplore.ieee.org/servlet/opac?punumber=2355

[MS-FSCADM] Microsoft Corporation, "Crawler Administration and Status Protocol Specification", November 2009.

[MS-FSCCFG] Microsoft Corporation, "Crawler Configuration File Format Specification", November 2009.

[MS-FSCF] Microsoft Corporation, "Content Feeding Protocol Specification", November 2009.

[MS-FSWCU] Microsoft Corporation, "WebAnalyzer/Crawler Utility Structure Specification", November 2009.

[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

[RFC2396] Berners-Lee, T., Fielding, R., and Masinter, L., "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998, http://www.ietf.org/rfc/rfc2396.txt

[RFC2616] Fielding, R., Gettys, J., Mogul, J., et al., "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999, http://www.ietf.org/rfc/rfc2616.txt

[RFC3490] Flatstrom, P., "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003, http://www.ietf.org/rfc/rfc3490.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.

1.3 Overview

The Crawler Multinode Transport Protocol specifies the communication messages that are exchanged in a multinode Web crawler setup. This protocol allows a Web crawler to be distributed over multiple nodes, that are each responsible for a subset of a Web crawl, and thus allow a Crawler to scale in network, CPU and I/O resources by sharing the load between the nodes.

A multinode Web crawler setup consists of four services, each responsible for a separate part of a Web crawl, as shown in the following figure.

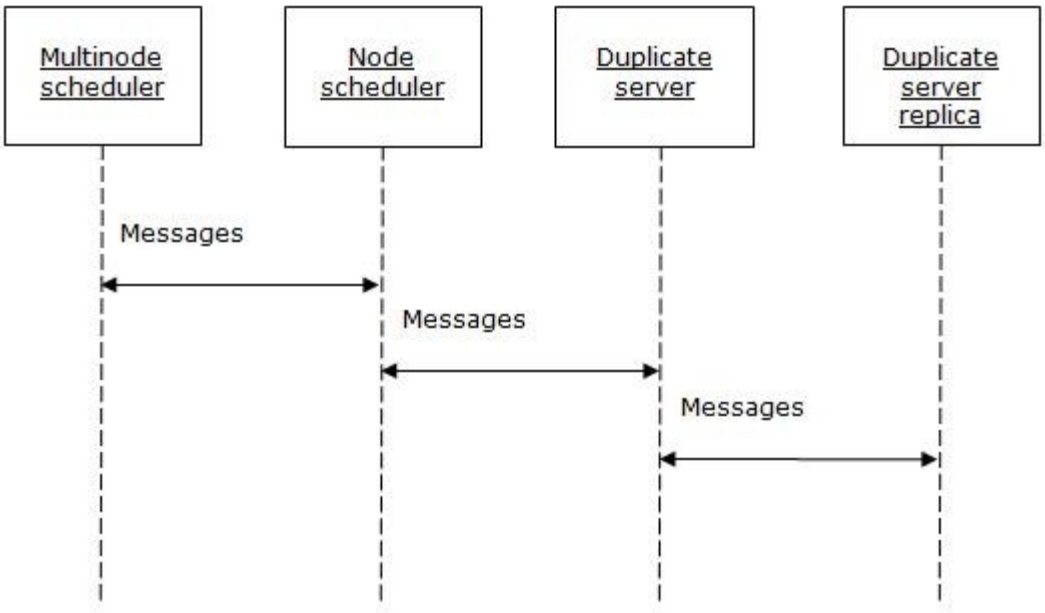


Figure 1: Communication overview for the Crawler Multinode Transport Protocol

The **multinode scheduler** orchestrates the Web crawl. It is responsible for **Domain Name System (DNS)** resolution and for distributing **crawl sites** among the participating **node schedulers**. The multinode scheduler also maintains the global state across all nodes of the node scheduler, including the configured **crawl collections**, statistics, and the IP addresses that are being crawled.

A multinode Web crawler setup can contain multiple node schedulers. Each node scheduler is responsible for the actual crawl of the subset of crawl sites that the multinode scheduler assigns to it. A node scheduler also maintains the local state of each crawl collection as well as the crawl statistics for the assigned crawl sites.

A **duplicate server** provides **duplicate** detection across all participating node schedulers. A duplicate server maintains a global database of all detected checksums and responds to participating node scheduler requests to add or remove checksums. A multinode Web crawler setup can contain multiple duplicate servers.

A duplicate server replica maintains an identical replica of the checksum database of a duplicate server. A duplicate server replica is primarily used to add fault tolerance to the multinode Web crawler setup. A multinode Web crawler setup can contain at most one duplicate server replica for each duplicate server.

1.4 Relationship to Other Protocols

The Crawler Multinode Transport Protocol uses **Transmission Control Protocol (TCP)** as shown in the following layering diagram:

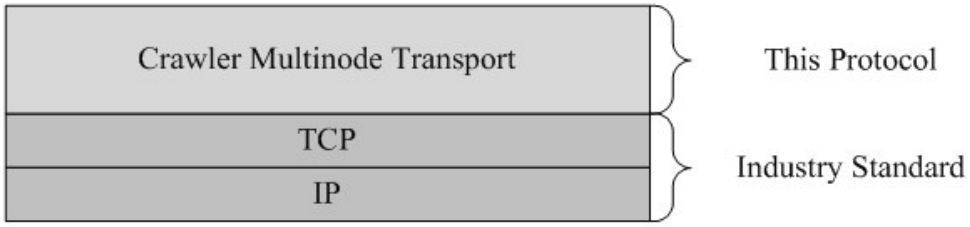


Figure 2: This protocol in relation to other protocols

Multiple messages are sent over the same TCP **connection (2)**.

1.5 Prerequisites/Preconditions

It is assumed that the node schedulers have information about the host name and the port of the multinode scheduler and of every duplicate server.

It is assumed that every duplicate server has information about the host name and the port of any duplicate server replica to which it is required to replicate its database.

1.6 Applicability Statement

This protocol is designed to transfer messages among the participating services in a multinode Web crawler setup.

This protocol is intended for use within private networks and is not appropriate for use on public networks.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

All messages are transferred over a TCP/IP connection (2). The standard protocol behavior calls for one TCP/IP connection (2) between each pair of participating components in the multinode Web crawler environment.

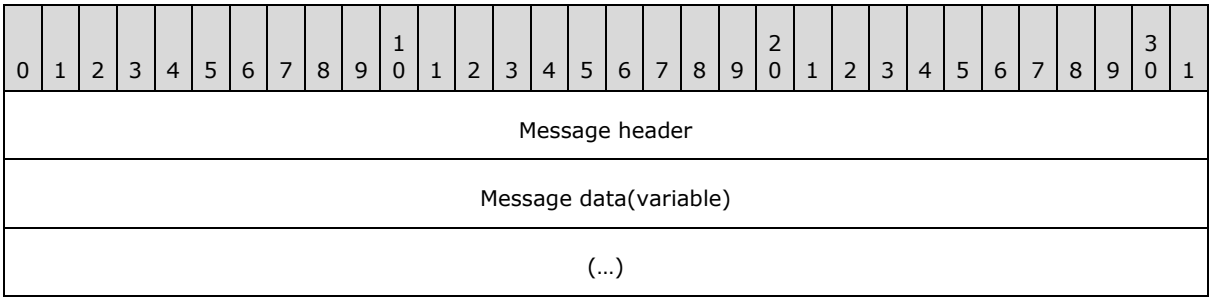
The numerical data format conventions are described in the following table.

Data type	Description
uint32_b	A **big-endian** 32-bit unsigned integer.
integer	A 32-bit signed integer.
long	A 64-bit signed long integer.
float	A 64-bit signed floating point number, as specified in [IEEE754].
string	A string that, unless otherwise stated, is encoded as **UTF-8**.
None	No value, as specified in [MS-FSWCU] section 2.1.4.

2.2 Message Syntax

2.2.1 Message Format

All messages MUST conform to the format that is shown in the following table.



Message header (uint32_b): The length, in bytes, of the message, excluding this message header field.

Message data (variable-length byte stream): A byte stream that is serialized as specified in [MS-FSWCU], that represents a tuple of length 2, and that contains the following objects in the order provided:

1. A dictionary that contains the message data, which consists of a command and its related data. For more details, see section 2.2.1.1.

2. An integer that MUST have a value of 0 or 1. A value of 0 specifies a normal-priority message, which MUST be appended to any communication queues. A value of 1 specifies a high-priority message, which MUST be delivered as quickly as possible, before normal-priority messages, to the recipient and which MUST be prepended to any communication queues.

2.2.1.1 Message Data Dictionary

The deserialized message data specifies a dictionary that contains a set of key/value pairs. The following table lists the two-letter strings that the dictionary keys MUST be specified as, the names that this protocol specification uses to refer to the keys, and the values that correspond to the keys.

Key	Key name	Value
"cm"	PKT_CMD	An **integer** specifying the type of packet command that the message includes. For more details, see sections 2.2.3, 2.2.4, and 2.2.5.
"ur"	PKT_URI	A **URI** that is specified as one of the following: A **crawl queue** entry structure. For more details, see section 2.2.2.2. A **string** that specifies the URI.
"vc"	PKT_VCLOCK	An **integer** or **long** value that specifies a **vector clock**. A vector clock is a number that MUST be incremented by 1 for each message that is sent over a connection (2). The vector clock can be used to detect the ordering of messages that are sent over a connection (2). If a received message contains a vector clock that is less than the current vector clock, and the content of the message requires sequencing, the protocol client MAY<1> ignore the message. For values greater than 2^{31}, the **long** data type is used; otherwise, **integer** is used.
"si"	PKT_SITE_ADDRINFO	An **addrinfo** structure as specified in section 2.2.2.1.
"dn"	PKT_DSPECNAME	A **string** specifying a crawl collection that the message applies to.
"st"	PKT_SITE	A **string** that specifies a crawl site.
"ds"	PKT_DSPEC	A structure that specifies a crawl configuration. For more details, see section 2.2.2.3.
"ep"	PKT_EPOCH	An **integer** that specifies the current version of a crawl configuration. The value MUST be incremented by 1 for every CMD_ADM_CONF_UPDATE and every CMD_ADM_CONF_ADD message that is sent.
"cs"	PKT_CSUM	A **string** of length 32 specifying a unique fingerprint of the Web document.
"id"	PKT_ID	A **string** that specifies a symbolic **node identifier**.
"ou"	PKT_OWNURI	A **string** that specifies an URI.
"wy"	PKT_WHY	A **string** that specifies the reason for the error returned by the PPDUP_ERROR message, as specified in section 2.2.5.2.
"mi"	PKT_NODE_ID	A **string** that specifies a symbolic node identifier.
"pd"	PKT_DATA	A structure that specifies the data of the message command. The format depends on the command that is specified by the PKT_CMD field.
"pi"	PKT_PPDUP_ID	A **string** that specifies a symbolic node identifier.
"db"	PKT_DBNAME	A **string** that specifies the name of a database.
"xp"	PKT_XMLRPCPRT	An **integer** that specifies a port number.

2.2.2 Complex Data Types

This section defines the complex data types that are used by the messages.

2.2.2.1 addrinfo Structure

The **addrinfo** structure MUST consist of an ordered tuple of length 5 that contains the results of a host name to IP address lookup, as described in the following table.

Index	Name	Description
0	Socket family	An **integer** that specifies the address socket family.
1	Socket type	An **integer** that specifies the address socket type.
2	Protocol	An **integer** that specifies the address protocol.
3	Canonical name	A **string** that specifies the canonical name of the address.
4	Address	A tuple of length 2 that contains the following entries: 1. A **string** that specifies an IP address. 2. An **integer** that specifies the port number. The value MUST be 0.

2.2.2.2 Crawl Queue Entry

This section specifies the data structure of a crawl queue entry, which is used for distributing URIs among node schedulers. A crawl queue entry consists of a URI to be crawled, meta information about the URI, and flags governing how the URI is to be crawled.

A crawl queue entry MUST be a tuple of length 9 as specified by the following table.

Index	Name	Description
0	Structure identifier	A **string** that identifies the crawl queue entry. The value of this **string** MUST be "WQE".
1	URI	A URI structure that specifies the URI to be crawled. For more details, see section 2.2.2.2.4.
2	Referring URI	A URI structure that specifies the URI of the Web document that this URI was extracted from. For more details, see section 2.2.2.2.4. If the URI was not extracted from a referring Web document, the referring URI structure MUST be based on the empty URI, "".
3	URI flags	An **integer** that specifies the properties of the URI. For more details, see section 2.2.2.2.1.
4	Frame flags	An **integer** that specifies whether the URI contains an **HTML** <FRAME> tag or was found within an HTML <FRAME> tag. For more details, see section 2.2.2.2.2.
5	Redirect trail	A list of all the redirections that lead to this URI. The list contains tuples of length 2 in the following format: A **string** that specifies the URI that redirected to this URI. An **integer** that specifies the type of redirection. The valid values, which

Index	Name	Description
		comprise a subset of the valid URI flags as specified in section 2.2.2.2.1, are: ▪ URIF_REDIRECT_300 ▪ URIF_REDIRECT_301 ▪ URIF_REDIRECT_302 ▪ URIF_META_REFRESH
6	Queue flag	An **integer** that specifies how this URI is queued. For more details, see section 2.2.2.2.3.
7	**Crawl refresh cycle**	An **integer** that specifies the crawl refresh cycle number, which identifies where this URI was last crawled.
8	Extra	A dictionary of **string** values that specify additional information associated with the URI. This information typically consists of metadata extracted from **RSS channels** or site maps.

2.2.2.2.1 URI Flag Values

The value of the **integer** that contains the URI flags MUST comprise a bitmask of one or more of the values that are described in the following table.

Bit value	Flag name	Description
0x00000000	URIF_NOFLAGS	Specifies that this URI has no flags attached.
0x00000001	URIF_STARTURI	Specifies that this is a **start URI**.
0x00000002	URIF_FORCE	Specifies that the crawler MUST force a crawl of this URI, regardless of whether this URI was already crawled during the current crawl cycle.
0x00000004	URIF_FORCESITE	Specifies that the crawler MUST force a crawl of this site, regardless of whether this site was already crawled during the current crawl cycle.
0x00000008	URIF_FORCE_REFEED	Specifies that the crawler MUST resubmit the document for content indexing, regardless of whether the document changed since its earlier submission.
0x00000010	URIF_JAVASCRIPT	Specifies that this URI points to a JavaScript document.
0x00000020	URIF_REDIRECT_301	Specifies that this URI was found through a 301 **Hypertext Transfer Protocol (HTTP)** response code as specified in [RFC2616].
0x00000040	URIF_REDIRECT_302	Specifies that this URI was found through a 302 HTTP response code as specified in [RFC2616].
0x00000080	URIF_DEL_IF_EXCL	Specifies that the crawler delete the document if it is excluded by the crawl configuration.
0x00000100	URIF_PROMOTE	Specifies that this URI was added to the crawl queue by a

Bit value	Flag name	Description
		PPDUP_PROMOTE_REQ message command as specified in section 2.2.5.8.
0x00000200	URIF_META_REFRESH	Specifies that the **forward link** was found in an HTML <META/> refresh tag as specified in [HTML].
0x00000400	URIF_STARTURI_FEED	Specifies that this URI is a start URI.
0x00001000	URIF_RESCHEDULED	Specifies that this URI was already crawled and rescheduled by the **adaptive crawling** refresh mode as specified in [MS-FSCCFG] section 2.2.3.
0x00002000	URIF_UNSCREENED	Specifies that this URI was not checked against the configuration **crawl rules** of the crawl collection.
0x00004000	URIF_META_NOINDEX	Specifies that this URI contains an HTML document with a NoIndex HTML <META/> tag as specified in [HTML].
0x00008000	URIF_META_NOFOLLOW	Specifies that this URI contains an HTML document with a NoFollow HTML <META/> tag as specified in [HTML].
0x00010000	URIF_SITEURI	Specifies that this URI MUST be used to indicate the crawling of the documents that were previously queued in the crawl queue for the crawl site. The URI itself MUST NOT be queued.
0x00020000	Not applicable	Reserved.
0x00040000	URIF_NO_CRAWL	Specifies that this URI MUST NOT be crawled. This flag MUST be used only in conjunction with the URIF_UPDATE_CARGO flag.
0x00080000	URIF_REDIRECT_300	Specifies that this URI was found through a 300 HTTP response code as specified in [RFC2616].
0x00100000	URIF_MULTIPLECHOICE	Specifies that this URI was found through a 300 HTTP response code with multiple choices as specified in [RFC2616]
0x00200000	URIF_FEED_HIGHPRI	Specifies that the location of the URI MUST be given a high priority when it is submitted for content indexing.
0x00400000	URIF_RSS	Specifies that this URI points to an RSS channel.
0x00800000	URIF_CLEANSITE	Specifies that this URI was added to the crawl queue by a clean crawl site operation. This operation takes place at the start of every crawl refresh cycle to verify the URIs that were not crawled for a number of crawl refresh cycles.
0x01000000	URIF_NOFOLLOW	Specifies that forward links from this URI MUST NOT be extracted or followed.
0x02000000	URIF_LOADLEVEL	Specifies that the node scheduler MUST use existing information about the number of hops that it took to find this URI from the start URIs.
0x04000000	URIF_IGNORE_RULES	Specifies that this URI MUST bypass all crawl rules.
0x08000000	URIF_RSS_CHILD	Specifies that this URI was extracted from an RSS channel.
0x10000000	URIF_SITEMAP	Specifies that this URI points to a site map.

Bit value	Flag name	Description
0x20000000	URIF_IGNORE_NETLOC	Specifies that this URI MUST bypass all host name crawl rules.
0x40000000	URIF_UPDATE_CARGO	Specifies that the crawl queue entry contains a dictionary that contains updated RSS channel or site map data.

2.2.2.2.2 Frame Flag Values

The value of the **integer** that contains the frame flags MUST be either 0 or one of the values that is described in the following table.

Value	Flag name	Description
1	URI_STATE_FRAME_PARENT	Specifies that the document that the URI points to contains a frameset as specified in [HTML].
2	URI_STATE_FRAME_CHILD	Specifies that the document that the URI points to consists of a frame document that was extracted from a frameset document.

2.2.2.2.3 Queue Flag Values

The value of the **integer** that contains the queue flags MUST comprise a bitmask of one or more of the values that are described in the following table.

Bit value	Flag name	Description
0x01	QF_PREPEND	Specifies that the crawl queue entry MUST be enqueued to the front of the crawl queue.
0x02	QF_FORCE	Specifies that this queue entry MUST bypass the **wqfilter**, **smfilter**, and **mufilter** filters as specified in [MS-FSCCFG] section 2.2.3.
0x04	QF_URGENT	Specifies that this crawl queue entry is urgent. If this bit is set, the node scheduler MUST crawl the queue entry as soon as possible.
0x08	Not applicable	Reserved.
0x10	Not applicable	Reserved.

2.2.2.2.4 URI Structure

The following table specifies the individual entries in the URI structure.

Index	Name	Description
0	URI	A **string** that specifies the URI.
1	Scheme	A **string** that specifies the scheme part of the URI.
2	Netloc	A **string** that specifies the network location part of the URI.
3	Host name	A **string** that specifies the host name part of the URI.

Index	Name	Description
4	Port	An **integer** that specifies the port part of the URI.
5	Path	A **string** that specifies the path part of the URI.
6	Query	A **string** that specifies the query part of the URI.
7	Parameters	A **string** that specifies the parameters part of the URI.
8	Fragments	A **string** that specifies the fragments part of the URI.
9	Credentials	A **string** that specifies the credentials part of the URI.
10	Default scheme	A **string** that specifies a default scheme for the URI.
11	Skip fragments	An **integer** that specifies whether to skip the fragments part: A value of 1 means skip. A value of 0 means do not skip.
12	Parsed	An **integer** that specifies whether the URI has been parsed in this structure as specified in [RFC2396]: A value of 1 means that the URI has been parsed. In this case, Entries 1–9 MUST contain the individually parsed portions of the URI. A value of 0 means that the URI has not been parsed. In this case, Entries 1–3 and 5–9 MUST each contain the default empty string value and Entry 4 MUST be set to **None**.
13	Host name IDNA	A string that specifies the IDNA-encoded host name of the URI as specified in [RFC3490].

2.2.2.3 Crawl Configuration

This section specifies how to convert the **XML** crawl configuration format as specified in [MS-FSCCFG] to the internal representation that is necessary for this protocol.

2.2.2.3.1 XML Element Mappings

The crawl configuration structure consists of a dictionary of all the crawl configurations. This dictionary MUST be keyed according to the **name XML attribute** that is found in the **DomainSpecification XML element** as specified in [MS-FSCCFG] section 2.2.2. The value of each of the entries MUST be a dictionary that contains the parameters of the respective crawl collection. These parameters MUST be mapped from each XML element to a data type according to the following table.

XML element	Data type	Key	Description
attrib	Specified by the **type** XML attribute. For more details, see section 2.2.2.3.2.	The **name** XML attribute of the **attrib** XML element.	This element contains a crawl configuration parameter.
section	Dictionary	The **name** XML attribute of the	This element MUST consist of a dictionary that contains all the parameters of the section.

XML element	Data type	Key	Description
		section XML element.	
SubDomain	Dictionary	"subdomains"	All the **SubDomain** XML elements MUST be stored in a dictionary with the key "subdomains". This dictionary MUST be keyed according to the **name** XML attribute of the **SubDomain** XML elements. Each entry MUST contain a dictionary with the parameters of the subcollection.
Login	Dictionary	"logins"	All the **Login** XML elements MUST be stored in a dictionary with the key "logins". This dictionary MUST be keyed according to the **name** XML attribute of the **Login** XML elements. Each entry MUST contain a dictionary with the parameters of the login session.
Node	Dictionary	"nodes"	All the **Node** XML elements MUST be stored in a dictionary with the key "nodes". This dictionary MUST be keyed according to the **name** XML attribute of the **Node** XML elements. Each entry MUST contain a dictionary with the parameters of a node scheduler.

2.2.2.3.2 attrib Element Mapping

An **attrib** XML element has a **type** XML attribute that defines the data type of the **attrib** XML element. The following table specifies how the **type** XML attribute MUST be mapped to a data type.

Value of type XML attribute	Mapped data type
"integer"	**integer**
"string"	**string**
"list-string"	**array**
"boolean"	**integer**, where 1 indicates True and 0 indicates False
"real"	**float**

For an **attrib** XML element with list-string as the **type**, the **array** data type MUST contain **string** values, where each **string** contains the value of a member XML element.

2.2.2.3.3 link_extraction Section Mappings

The **attrib** XML element names of the **link_extraction** section MUST be mapped to integer keys according to the following table.

attrib XML element name	Dictionary key
a	0x1

attrib XML element name	Dictionary key
action	0x2
area	0x4
card	0x400000
comment	0x10
embed	0x100000
frame	0x20
go	0x40
img	0x80
layer	0x100
link	0x200
meta	0x400
meta_refresh	0x8000
object	0x4000
script	0x800
script_java	0x1000
style	0x2000

2.2.3 Node Scheduler Messages

The following table lists all of the messages that are handled by the node scheduler. The message value MUST be specified as an **integer**.

Value	Message name	Description
10	CMD_UM_KEEPALIVE	For more details, see section 2.2.3.1.
11	CMD_UM_KEEPALIVE_ACK	For more details, see section 2.2.3.2.
25	CMD_UM_URI	For more details, see section 2.2.3.3.
26	CMD_UM_URI_URG	For more details, see section 2.2.3.4.
3	CMD_ADM_CONF_SUSPEND	For more details, see section 2.2.3.5.
4	CMD_ADM_CONF_RESUME	For more details, see section 2.2.3.6.
23	CMD_ADM_CONF_FEEDING_SUSPEND	For more details, see section 2.2.3.7.
7	CMD_UM_ADD_ROUTE	For more details, see section 2.2.3.8.
8	CMD_UM_ASK_ROUTE	For more details, see section 2.2.3.9.
5	CMD_ADM_CONF_REFRESH	For more details, see section 2.2.3.10.

Value	Message name	Description
1	CMD_ADM_CONF_ADD	For more details, see section 2.2.3.11.
0	CMD_ADM_CONF_UPDATE	For more details, see section 2.2.3.12.
2	CMD_ADM_CONF_REMOVE	For more details, see section 2.2.3.13.
6	CMD_ADM_PREEMPT_SITE	For more details, see section 2.2.3.14.
14	CMD_UM_REPROCESS_SITE	For more details, see section 2.2.3.15.
18	CMD_UM_DELETE_SITE	For more details, see section 2.2.3.16.
19	CMD_UM_DELETE_URIS	For more details, see section 2.2.3.17.
15	CMD_UM_QUARANTINE_SITE	For more details, see section 2.2.3.18.
21	CMD_UM_QUARANTINE_SITE_REQUEUE	For more details, see section 2.2.3.19.
13	CMD_UM_DNS_REPLY	For more details, see section 2.2.3.20.
20	CMD_UM_CONF_STATE	For more details, see section 2.2.3.21.
16	CMD_UM_START_CRAWL_IP	For more details, see section 2.2.3.22.
50	PPDUP_ADD_OK	For more details, see section 2.2.5.1.
51	PPDUP_ERROR	For more details, see section 2.2.5.2.
54	PPDUP_REMOVE_OK	For more details, see section 2.2.5.5.
55	PPDUP_KEEPALIVE	For more details, see section 2.2.5.6.
56	PPDUP_KEEPALIVE_ACK	For more details, see section 2.2.5.7.
57	PPDUP_PROMOTE_REQ	For more details, see section 2.2.5.8.
60	PPDUP_CONFIG_ACK	For more details, see section 2.2.5.11.

2.2.3.1 CMD_UM_KEEPALIVE

The CMD_UM_KEEPALIVE message contains a command that is used to keep the connection (2) to the multinode scheduler active and to check the availability and responsiveness of the connection (2).

This message MUST contain the key/value pair that is listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_KEEPALIVE

2.2.3.2 CMD_UM_KEEPALIVE_ACK

The CMD_UM_KEEPALIVE_ACK message contains a command that functions as the response to a previously sent CMD_UM_KEEPALIVE message as specified in section 2.2.4.2.

This message MUST contain the key/value pair that is listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_KEEPALIVE_ACK

2.2.3.3 CMD_UM_URI

The CMD_UM_URI message contains a command that specifies a crawl queue entry.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_URI
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies
PKT_DATA	A crawl queue entry structure as specified in section 2.2.2.2

2.2.3.4 CMD_UM_URI_URG

The CMD_UM_URI_URG message contains a command that specifies a URI crawl queue entry.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_URI_URG
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies
PKT_DATA	A crawl queue entry structure as specified in section 2.2.2.2

2.2.3.5 CMD_ADM_CONF_SUSPEND

The CMD_ADM_CONF_SUSPEND message contains a command to temporarily suspend the crawling of the specified crawl collection.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_ADM_CONF_SUSPEND
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	An **integer** that indicates the state that the node scheduler MUST enter. The value MUST be one of the following: 4 to indicate that the crawl collection is being suspended based on user input 5 to indicate that the crawl collection is being suspended because the disk is close to being full 8 to indicate that the crawl collection is being suspended because of the variable delay option

2.2.3.6 CMD_ADM_CONF_RESUME

The CMD_ADM_CONF_RESUME message contains a command to resume the crawling of the specified crawl collection.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_ADM_CONF_RESUME
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies

2.2.3.7 CMD_ADM_CONF_FEEDING_SUSPEND

The CMD_ADM_CONF_FEEDING_SUSPEND message contains a command to temporarily suspend or resume the content submission of Web documents for content indexing of the specified content-submission destinations as specified in [MS-FSCCFG] section 2.2.4.9.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_ADM_CONF_FEEDING_SUSPEND
PKT_DSPECNAME	A **string** that specifies the crawler collection to which the message applies.
PKT_DATA	A tuple of length 2 that contains the following entries: An integer that specifies the new content submission state: 1 to suspend or 0 to resume. The content submission destinations that the new state applies to. The valid values are: **None** to specify that the new state applies to all the existing content submission destinations for the specified collection. An array of **string** values that specify the named content-submission destinations that this operation applies to. Each **string** MUST specify either the name of a content submission destination as specified in [MS-FSCCFG] or the **string** "default:<content collection>", where <content collection> MUST specify a valid content collection name.

2.2.3.8 CMD_UM_ADD_ROUTE

The CMD_UM_ADD_ROUTE message contains a command to add a new route from the multinode scheduler to a node scheduler.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_ADD_ROUTE
PKT_DSPECNAME	A **string** that specifies the crawler collection to which the message applies.
PKT_DATA	A tuple of length 5 that contains the following entries: The node identifier of the node scheduler that the specified crawl site MUST be routed

Key name	Value
	to. A **string** that specifies the **crawl routing** attribute, which is the part of the host name that is used for the crawl routing decision. A **string** that specifies the crawl site. An **addrinfo** structure that contains the address information for the crawl site. If the specified crawl collection is configured to crawl through a proxy as specified in [MS-FSCCFG], the value of the **addrinfo** structure is **None**. An **integer** that specifies the time in seconds before the address information mapping MUST be resolved again.

2.2.3.9 CMD_UM_ASK_ROUTE

The CMD_UM_ASK_ROUTE message contains a command requesting that all stored routing information be sent to the multinode scheduler.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_ASK_ROUTE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.

2.2.3.10 CMD_ADM_CONF_REFRESH

The CMD_ADM_CONF_REFRESH message contains a command for the node scheduler to initiate a new crawl refresh cycle for the specified crawl or subcrawl collection based on the content of PKT_DATA.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_ADM_CONF_REFRESH
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the command message applies.
PKT_DATA	A tuple of length 2 that contains the following entries: A **string** that specifies a subcollection to refresh: To refresh the entire crawl collection, specify the value **None**. To refresh a subcollection, specify a valid subcollection for the crawl collection. An **integer** bitmask that specifies the part of the crawl collection to be refreshed: A bit value of 0x1 specifies that the collection configuration MUST be loaded from disk. A bit value of 0x2 specifies that the entire crawl collection MUST be refreshed. A bit value of 0x4 specifies that the subcollection MUST be refreshed. A bit value of 0x8 specifies that only start URIs MUST be requeued. A bit value of 0x16 specifies that the crawl work queues MUST be truncated and the crawl restarted.

2.2.3.11 CMD_ADM_CONF_ADD

The CMD_ADM_CONF_ADD message contains a command to add a new crawl collection to the node scheduler.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_ADM_CONF_ADD
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DSPEC	A crawl configuration structure as specified in section 2.2.2.3.

2.2.3.12 CMD_ADM_CONF_UPDATE

The CMD_ADM_CONF_UPDATE message contains a command to update the crawl configuration of an existing crawl collection in the node scheduler.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_ADM_CONF_UPDATE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DSPEC	A crawl configuration structure as defined in section 2.2.2.3.

2.2.3.13 CMD_ADM_CONF_REMOVE

The CMD_ADM_CONF_REMOVE message contains a command to remove an existing crawl collection from the node scheduler.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_ADM_CONF_REMOVE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.

2.2.3.14 CMD_ADM_PREEMPT_SITE

The CMD_ADM_PREEMPT_SITE message contains a command to temporarily stop the crawling of the specified crawl site.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_ADM_PREEMPT_SITE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.

Key name	Value
PKT_DATA	A tuple of length 2 that contains the following entries: A **string** that specifies the crawl site to be preempted. An **integer** that specifies whether the site is to be requeued to the crawl queue after being preempted. This integer can be one of the following: A value of 1 to requeue. The crawl site MUST be queued to the front of the crawl queue. A value of 0.

2.2.3.15 CMD_UM_REPROCESS_SITE

The CMD_UM_REPROCESS_SITE message contains a command to resubmit all the documents that match the contents of PKT_DATA to document processing process for reindexing.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_REPROCESS_SITE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A tuple of length 4 that contains the following entries: 1. A **string** that specifies the crawl site to be reprocessed. 2. A **string** that specifies a URI prefix matching the URIs to be reprocessed. 3. A **string** that specifies the content submission destinations to which the new state applies. The valid values are: 4. **None** to specify that the new state applies to all the existing content-submission destinations for the specified collection. 5. A **string** that specifies the named content-submission destinations to which this operation applies. The string MUST specify either the name of a content submission destination as specified in [MS-FSCCFG] or the string "default:<content collection>", where <content collection> MUST specify a valid content collection name. 6. An **integer** that specifies whether the prefix is an exact match: 7. A value of 1 specifies an exact match. 8. A value of 0 specifies a prefix match.

2.2.3.16 CMD_UM_DELETE_SITE

The CMD_UM_DELETE_SITE message contains a command to delete the specified crawl site from both local data storage and the content index.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_DELETE_SITE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A **string** that specifies the crawl site to delete.

2.2.3.17 CMD_UM_DELETE_URIS

The CMD_UM_DELETE_URIS contains a command for the node scheduler to delete the specified URIs from both local storage and the content index.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_DELETE_URIS
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A tuple of length 2 that contains the following entries: 1. A dictionary that specifies arrays of URIs that are keyed according to their respective crawl sites. 2. A dictionary that contains the **string** key "errcode", with the **string** "USR" to be output to the node scheduler fetch log.

2.2.3.18 CMD_UM_QUARANTINE_SITE

The CMD_UM_QUARANTINE_SITE message contains a command to temporarily block a crawl site from being crawled.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_QUARANTINE_SITE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A tuple of length 2 that contains the following entries: 1. A **string** that specifies the crawl site to be quarantined. 2. An **integer** that specifies the time, in seconds, for which the crawl site MUST remain quarantined.

2.2.3.19 CMD_UM_QUARANTINE_SITE_REQUEUE

The CMD_UM_QUARANTINE_SITE_REQUEUE message contains a command to temporarily block a crawl site from being crawled.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_QUARANTINE_SITE_REQUEUE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the command message applies.
PKT_DATA	A tuple of length 2 that contains the following entries: 1. A **string** that specifies the crawl site to be quarantined. 2. An **integer** that specifies the time, in seconds, for which the crawl site MUST remain quarantined.

2.2.3.20 CMD_UM_DNS_REPLY

The CMD_UM_DNS_REPLY message contains a command with the response to a previously sent CMD_UM_DNS_REQUEST command as specified in section 2.2.4.4.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_DNS_REPLY
PKT_DSPECNAME	A value that MUST be set to an empty **string**.
PKT_DATA	A tuple of length 2 that contains the following entries: 1. A **string** that specifies the host name that was resolved. 2. A tuple of length 3 that contains the following entries: 3. An **addrinfo** structure, or **None** if the host name lookup was unsuccessful. 4. An **integer** that specifies the time at which the host name resolution was finished. 5. An **integer** that specifies time to live, in seconds, for this host name resolution.

2.2.3.21 CMD_UM_CONF_STATE

The CMD_UM_CONF_STATE message contains a command to set a new state for the crawl collection.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_CONF_STATE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	An **integer** that specifies the new state of the node scheduler.

The following table lists the valid state values for the PKT_DATA key.

Value	Meaning
1	The crawl collection is active and can queue new URIs.
2	The crawl collection is temporarily blocked from queuing new URIs.
3	The crawl collection is in the process of being removed.
4	The crawl collection was suspended by the user.
5	The crawl collection was suspended because the disk is almost full.
6	The crawl collection initiated its nightly compaction.
7	The crawl collection is currently compacting.
8	The crawl collection is suspended by the variable delay option.

2.2.3.22 CMD_UM_START_CRAWL_IP

The CMD_UM_START_CRAWL_IP message contains a command that functions as a response to a previous CMD_UM_START_CRAWL_IP request command as specified in section 2.2.4.11.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_START_CRAWL_IP
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A tuple of length 4 that contains the following entries: 1. A **string** that specifies the IP address or, if the crawl collection is configured to crawl through a proxy, the host name of the crawl site. 2. A **string** that specifies the crawl site. 3. An **integer** that specifies the crawl store identifier, which identifies the storage cluster (as specified in [MS-FSCCFG] section 2.2.4.6) to which the crawl site belongs. 4. An **integer** that specifies whether crawling the specified IP address or crawl site is allowed. The options are: 5. A value of 1 to specify that crawling is allowed. 6. A value of 0 to specify that crawling is not allowed. In this case, the node scheduler MUST temporarily block the IP address or crawl site from being crawled for 30 minutes.

2.2.4 Multinode Scheduler Messages

The following table lists all of the messages that are handled by the multinode scheduler. The message value MUST be specified as an **integer**.

Value	Message name	Description
24	CMD_UM_INIT	For more details, see section 2.2.4.1.
10	CMD_UM_KEEPALIVE	For more details, see section 2.2.4.2.
11	CMD_UM_KEEPALIVE_ACK	For more details, see section 2.2.4.3.
12	CMD_UM_DNS_REQUEST	For more details, see section 2.2.4.4.
25	CMD_UM_URI	For more details, see section 2.2.4.5.
27	CMD_UM_STAT	For more details, see section 2.2.4.6.
26	CMD_UM_URI_URG	For more details, see section 2.2.4.7.
28	CMD_UM_LOG	For more details, see section 2.2.4.8.
7	CMD_UM_ADD_ROUTE	For more details, see section 2.2.4.9.
9	CMD_UM_LAST_ROUTE	For more details, see section 2.2.4.10.
16	CMD_UM_START_CRAWL_IP	For more details, see section 2.2.4.11.
17	CMD_UM_STOP_CRAWL_IP	For more details, see section 2.2.4.12.
20	CMD_UM_CONF_STATE	For more details, see section 2.2.4.13.
22	CMD_UM_DELETE_OK	For more details, see section 2.2.4.14.

2.2.4.1 CMD_UM_INIT

The CMD_UM_INIT message contains a command to initialize the connection (2) between a node scheduler and a multinode scheduler.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_INIT
PKT_NODE_ID	A **string** that specifies the node identifier of the connected node scheduler.
PKT_XMLRPCPRT	A tuple of length 2 that contains the following entries: 1. A **string** that specifies the host name of the node scheduler. 2. An **integer** that specifies the port at which the administrative interface of the node scheduler listens. For more details, see [MS-FSCADM].

2.2.4.2 CMD_UM_KEEPALIVE

The CMD_UM_KEEPALIVE message contains a command that is used to keep the connection (2) between the node scheduler and the multinode scheduler active and to check the availability and responsiveness of that connection (2).

This message MUST contain the key/value pair that is listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_KEEPALIVE

2.2.4.3 CMD_UM_KEEPALIVE_ACK

The CMD_UM_KEEPALIVE_ACK message contains a command that functions as the response to a previously sent CMD_UM_KEEPALIVE message as specified in section 2.2.3.1.

This message MUST contain the key/value pair that is listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_KEEPALIVE_ACK

2.2.4.4 CMD_UM_DNS_REQUEST

The CMD_UM_DNS_REQUEST message contains a command for the multinode scheduler to resolve the specified host name.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_DNS_REQUEST
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_SITE	A **string** that specifies the host name to resolve.

2.2.4.5 CMD_UM_URI

The CMD_UM_URI message contains a command that specifies a crawl queue entry for a specified crawl collection.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_URI
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A crawl queue entry structure.

2.2.4.6 CMD_UM_STAT

The CMD_UM_STAT message contains a command that contains an update to the statistics structure for a specified crawl collection.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_STAT
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A statistics structure as specified in [MS-FSCADM] section 2.2.5.

2.2.4.7 CMD_UM_URI_URG

The CMD_UM_URI_URG message contains a command that specifies a high-priority crawl queue entry structure for a specified crawl collection.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_URI_URG
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A crawl queue entry structure as specified in section 2.2.2.2.

2.2.4.8 CMD_UM_LOG

The CMD_UM_LOG message contains a command that specifies a message to be logged to the specified log of the specified crawl collection.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_LOG
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A tuple of length 2 that contains the following entries: 1. A **string** that specifies the log **file**. 2. A **string** that specifies the message to be logged.

The following table specifies the valid log files.

Log-file value	Log-file name	Description
"si"	CF_SITELOGFILE	Contains crawl site log messages.
"ur"	CF_SCREENEDLOGFILE	Contains screened log messages.
"sc"	CF_SCHEDULELOGFILE	Contains schedule log messages.
"dk"	CF_LOGFILE	Contains fetch log messages.

2.2.4.9 CMD_UM_ADD_ROUTE

The CMD_UM_ADD_ROUTE message contains a command that specifies a new crawl route to be added. This command functions as a response to a previously sent CMD_UM_ASK_ROUTE message as specified in section 2.2.3.9.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_ADD_ROUTE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A variable-length array (with a maximum of 256 entries) in which each entry is a tuple of length 2 that contains the following entries: A **string** that specifies the crawl routing attribute, which defines the part of the host name that is used for the crawl routing decision. Crawl sites that match this attribute MUST be routed to the specified node scheduler. A **string** that specifies the node identifier of the node scheduler.

2.2.4.10 CMD_UM_LAST_ROUTE

The CMD_UM_LAST_ROUTE message contains a command that both functions as the response to a previously sent CMD_UM_ASK_ROUTE command as specified in section 2.2.3.9 and indicates that the node scheduler has sent all of its stored routing information.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_LAST_ROUTE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.

2.2.4.11 CMD_UM_START_CRAWL_IP

The CMD_UM_START_CRAWL_IP message contains a command that functions as a request for permission to crawl the specified crawl site.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_START_CRAWL_IP
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A tuple of length 3 that contains the following entries: 1. A **string** that specifies the IP address of the crawl site that permission is being requested for. If the crawl is performed through a proxy, the **string** MUST specify the crawl site instead of an IP address. 2. A **string** that specifies the crawl site that permission is being requested

Key name	Value
	for. 3. An **integer** that specifies the crawl store identifier, which identifies the storage cluster (as specified in [MS-FSCCFG] section 2.2.4.6) to which the crawl site belongs.

2.2.4.12 CMD_UM_STOP_CRAWL_IP

The CMD_UM_STOP_CRAWL_IP message contains a command specifying that the node scheduler has stopped crawling the specified crawl site.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_STOP_CRAWL_IP
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_DATA	A tuple of length 2 that contains the following entries: 1. A **string** that specifies the IP address of the crawl site that the node scheduler stopped crawling. If the crawl occurred through a proxy, the **string** MUST specify the crawl site instead of an IP address. 2. A **string** specifying the crawl site that the node scheduler stopped crawling.

2.2.4.13 CMD_UM_CONF_STATE

The CMD_UM_CONF_STATE message contains a command that functions as a request for the multinode scheduler to return the current state of the specified crawl collection.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_CONF_STATE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.

2.2.4.14 CMD_UM_DELETE_OK

The CMD_UM_DELETE_OK message contains a command that functions as the response to a previously sent CMD_ADM_CONF_REMOVE command as specified in section 2.2.3.13.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_DELETE_OK

Key name	Value
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.

2.2.5 Duplicate Server Messages

Duplicate server messages are issued from a node scheduler to a duplicate server and vice versa, as well as from one duplicate server to another.

The following table lists all of the messages that are handled by the duplicate server. The message value MUST be specified as an **integer**.

Value	Message name	Description
50	PPDUP_ADD_OK	For more details, see section 2.2.5.1.
51	PPDUP_ERROR	For more details, see section 2.2.5.2.
52	PPDUP_ADD	For more details, see section 2.2.5.3.
53	PPDUP_REMOVE	For more details, see section 2.2.5.4.
54	PPDUP_REMOVE_OK	For more details, see section 2.2.5.5.
55	PPDUP_KEEPALIVE	For more details, see section 2.2.5.6.
56	PPDUP_KEEPALIVE_ACK	For more details, see section 2.2.5.7.
57	PPDUP_PROMOTE_REQ	For more details, see section 2.2.5.8.
58	PPDUP_CONF_REMOVE	For more details, see section 2.2.5.9.
59	PPDUP_SET_CONFIG	For more details, see section 2.2.5.10.
60	PPDUP_CONFIG_ACK	For more details, see section 2.2.5.11.
61	PPDUP_CONF_REMOVE2	For more details, see section 2.2.5.12.

2.2.5.1 PPDUP_ADD_OK

The PPDUP_ADD_OK message contains a command specifying a new URI that was added to a duplicate server or duplicate server replica. This message functions as a response to a previously sent PPDUP_ADD command as specified in section 2.2.5.3.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_ADD_OK
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_ID	A **string** that specifies the node identifier of the node scheduler from which the message originated.
PKT_URI	A **string** that specifies the URI that was added to the duplicate server.

This message can also contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_OWNURI	A **string** specifying the **owner URI** of the checksum that was specified in the PPDUP_ADD message. This key MUST be included under certain conditions. For more details, see section 3.3.5.4.
PKT_PPDUP_ID	A **string** specifying the node identifier of the duplicate server from which this message was received. This key MUST be included under certain conditions. For more details, see section 3.4.5.3.
PKT_DBNAME	A **string** specifying the path to the duplicate database of the crawl collection. This key MUST be included under certain conditions. For more details, see section 3.4.5.3.

2.2.5.2 PPDUP_ERROR

The PPDUP_ERROR message functions as a response to an unsuccessful command.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_ERROR
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_WHY	A **string** that specifies the cause of the error.

2.2.5.3 PPDUP_ADD

The PPDUP_ADD message contains a command that specifies a new URI to be added for a crawl collection. For details about when and how to use this message, see sections 3.1.5.23, 3.3.5.4, and 3.4.5.3.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_ADD
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_ID	A **string** that specifies the node identifier of the node scheduler from which this message originated.
PKT_URI	A **string** that specifies the URI to add to the duplicate server or duplicate server replica.
PKT_CSUM	A **string** that specifies the checksum, which MUST be 16 bytes, of the document to add.

This message can also contain the key/value pairs that are listed in the following table. These keys MUST be included under certain conditions. For more details, see section 3.3.5.4.

Key name	Value
PKT_PPDUP_ID	A **string** that specifies the node identifier of the duplicate server from which this command was received.
PK_DBNAME	A **string** that specifies the path to the duplicate database of the crawl collection.

2.2.5.4 PPDUP_REMOVE

The PPDUP_REMOVE message contains a command that specifies a URI to remove from a crawl collection.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_REMOVE
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_ID	A **string** that specifies the node identifier of the node scheduler from which this message originated.
PKT_URI	A **string** that specifies the owner URI to remove.
PKT_CSUM	A **string** that specifies the checksum of the document to remove.

This message can also contain the key/value pairs that are listed in the following table. These keys MUST be included under certain conditions. For more details, see section 3.3.5.5.

Key name	Value
PKT_PPDUP_ID	A **string** that specifies the node identifier of the duplicate server from which this command was received.
PK_DBNAME	A **string** that specifies the path to the duplicate database of the crawl collection.

2.2.5.5 PPDUP_REMOVE_OK

The PPDUP_REMOVE_OK message functions as a response to a previously sent PPDUP_REMOVE message as specified in section 2.2.5.4.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_REMOVE_OK
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_URI	A **string** that specifies the URI of the document that was removed.

This message can also contain the key/value pairs that are listed in the following table. These keys MUST be included under certain conditions. For more details, see section 3.4.5.4.

Key name	Value
PKT_PPDUP_ID	A **string** that specifies the node identifier of the duplicate server from which this command was received.
PK_DBNAME	A **string** that specifies the path to the duplicate database of the crawl collection.

2.2.5.6 PPDUP_KEEPALIVE

The PPDUP_KEEPALIVE message contains a command that is used to keep the connections (2) to the node scheduler and duplicate server replica active.

This message MUST contain the key/value pair that is listed in the following table.

Key name	Value
PKT_CMD	PPDUP_KEEPALIVE

2.2.5.7 PPDUP_KEEPALIVE_ACK

The PPDUP_KEEPALIVE_ACK message functions as a response to a PPDUP_KEEPALIVE message.

This message MUST contain the key/value pair that is listed in the following table.

Key name	Value
PKT_CMD	PPDUP_KEEPALIVE_ACK

2.2.5.8 PPDUP_PROMOTE_REQ

The PPDUP_PROMOTE_REQ message is used to notify all the node schedulers to initiate a recrawl of the URIs that match the specified document checksum.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_PROMOTE_REQ
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.
PKT_CSUM	A **string** that specifies the checksum of the documents to recrawl.

2.2.5.9 PPDUP_CONF_REMOVE

The PPDUP_CONF_REMOVE message contains a command to remove an existing crawl collection from a duplicate server or duplicate server replica.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_CONF_REMOVE

Key name	Value
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.

This message can also contain the key/value pair that is listed in the following table.

Key name	Value
PKT_PPDUP_ID	A **string** that specifies the node identifier of the duplicate server from which this command was received. This key MUST be included under certain conditions. For more details, see section 3.3.5.2.

2.2.5.10 PPDUP_SET_CONFIG

The PPDUP_SET_CONFIG message contains a command to add a new crawl collection to a duplicate server or duplicate server replica.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_SET_CONFIG
PKT_DATA	A dictionary that specifies the crawl collection configurations as specified in section 2.2.2.3.
PKT_VCLOCK	An **integer** that specifies the vector clock of either the node scheduler or duplicate server, depending on which service the message was sent from.

This message can also contain the key/value pair that is listed in the following table.

Key name	Value
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies. This key MUST be included under certain conditions. For more details, see section 3.1.5.11.

2.2.5.11 PPDUP_CONFIG_ACK

The PPDUP_CONFIG_ACK message functions as a response to the PPDUP_CONFIG message.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_CONFIG_ACK
PKT_VCLOCK	An **integer** that specifies the vector clock of either the node scheduler or duplicate server, depending on which service the message was sent from.

2.2.5.12 PPDUP_CONF_REMOVE2

The PPDUP_CONF_REMOVE2 message contains a command to remove an existing crawl collection from a duplicate server.

This message MUST contain the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_CONF_REMOVE2
PKT_DSPECNAME	A **string** that specifies the crawl collection to which the message applies.

3 Protocol Details

3.1 Node Scheduler Details

The node scheduler communicates and exchanges messages with a multinode scheduler and, if configured, to duplicate servers.

The node scheduler receives new crawl collection configurations from the multinode scheduler and sets up the required data structures. After a crawl collection is set up, the node scheduler receives new crawl queue entries from the multinode scheduler that is queued to the local crawl queue. Crawl queue entries are then extracted from the crawl queue. If the crawl site of the crawl queue entry is not active, the node scheduler communicates with the multinode scheduler to check whether it is permitted to crawl the IP address that is associated with the crawl site. If a crawl operation is approved, or if the crawl site is already active, the documents are downloaded from the Web.

After the Web documents are downloaded, they are processed for forward links. Forward links to crawl sites that are not registered to the node scheduler are sent back to the multinode scheduler as crawl queue entries, while crawl sites that are registered to the node scheduler are queued to the local crawl queue.

After the documents are downloaded they are sent to the content-indexing process for indexing. For more details, see [MS-FSCF] section 2.2.39.

If a duplicate server is configured for the crawl collection, the node scheduler issues messages to the duplicate server to check for duplicate documents before sending them to the content-indexing process.

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.

Crawl configuration: A tracking of all the active crawl collections and their respective crawl configurations, as specified in section 2.2.2.3.

Crawl site IP addresses: A list of all the IP addresses that are actively being crawled for each crawl collection.

Crawl statistics: The crawl statistics, as specified in [MS-FSCADM] section 2.2.5, for each active crawl collection.

Crawl store: The document data and metadata that is associated with each crawled URI for each crawl collection.

Crawl queue: A queue of URIs to be crawled for each crawl collection.

DNS database: A database of resolved IP addresses and their expiration times.

Duplicate server status structure: A structure that contains tracking information about the duplicate servers that are currently connected and their status.

Routing database: A database of all the registered routing decisions that have been received from the multinode scheduler for each crawl collection.

Quarantine list: A list of all the currently quarantined crawl sites for each crawl collection and their expiration times.

3.1.2 Timers

The Duplicate Server Keep-Alive timer issues PPDUP_KEEPALIVE messages to all known duplicate servers to keep the connection alive and detect connection failures. The default value is 120 seconds.

The Multinode Scheduler Keep-Alive timer issues a CMD_UM_KEEPALIVE message to the multinode scheduler to keep the connection alive and detect connection failures. The default value is 120 seconds.

The Quarantine timer inspects the **Quarantine list** to check if the quarantine of any of the registered sites have expired. The default value is 5 seconds.

The Statistics timer sends CMD_UM_STAT messages with updated **Crawl statistics** to the multinode scheduler. The default value is 60 seconds.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 Receiving a CMD_UM_KEEPALIVE Message

After receiving a CMD_UM_KEEPALIVE message, a node scheduler MUST send a CMD_UM_KEEPALIVE_ACK reply message, as illustrated by the following figure.

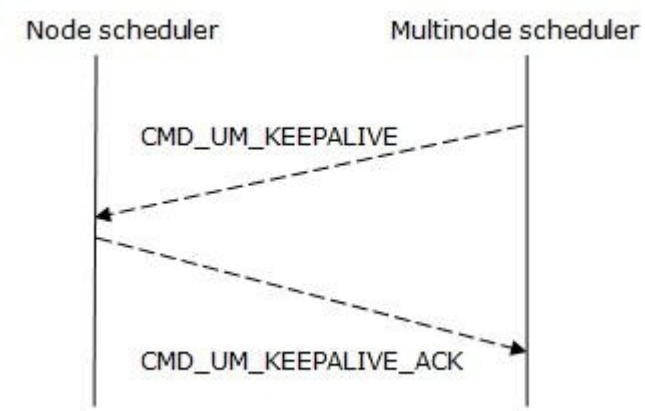


Figure 3: Sequencing rule for a CMD_UM_KEEPALIVE message

3.1.5.2 Receiving a CMD_UM_URI Message

After receiving a CMD_UM_URI message, a node scheduler MUST enqueue the crawl queue entry that the message contains to the back of the crawl queue.

3.1.5.3 Receiving a CMD_UM_URI_URG Message

After receiving a CMD_UM_URI_URG message, a node scheduler MUST enqueue the crawl queue entry that the message contains to the front of the crawl queue.

3.1.5.4 Receiving a CMD_ADM_CONF_SUSPEND Message

After receiving a CMD_ADM_CONF_SUSPEND message, a node scheduler MUST stop crawling all of the crawl sites that are handled by the specified crawl collection. Crawling the crawl collection MUST NOT resume until a CMD_ADM_CONF_RESUME message is received. For more information, see section 3.1.5.5.

3.1.5.5 Receiving a CMD_ADM_CONF_RESUME Message

After receiving a CMD_ADM_CONF_RESUME message, a node scheduler MUST resume crawling the specified crawl collection.

3.1.5.6 Receiving a CMD_ADM_CONF_FEEDING_SUSPEND Message

After receiving a CMD_ADM_CONF_FEEDING_SUSPEND message, a node scheduler MUST either suspend or resume the content feeding of crawled content to the indexing process, based on the content of the message. For more details, see section 2.2.3.7.

3.1.5.7 Receiving a CMD_UM_ADD_ROUTE Message

After receiving a CMD_UM_ADD_ROUTE message, a node scheduler MUST process the message as follows:

- The node scheduler MUST store the route information that is specified in the message in the routing database.
- If the node identifier that is specified in the message matches the node identifier of the node scheduler, the node scheduler MUST store the **addrinfo** structure specified in the message in the local DNS database. (The message contains the crawl site that was routed, the node identifier of the node it was routed to, and its associated **addrinfo** structure. For more details, see section 2.2.3.8.)

3.1.5.8 Receiving a CMD_UM_ASK_ROUTE Message

After receiving a CMD_UM_ASK_ROUTE message, a node scheduler MUST process the message as follows:

1. The node scheduler MUST send a CMD_UM_ADD_ROUTE reply message for every route that the node scheduler registers in its routing database.
2. After all the CMD_UM_ADD_ROUTE messages are sent, the node scheduler MUST send a CMD_UM_LAST_ROUTE message.

The following figure illustrates this sequence of events:

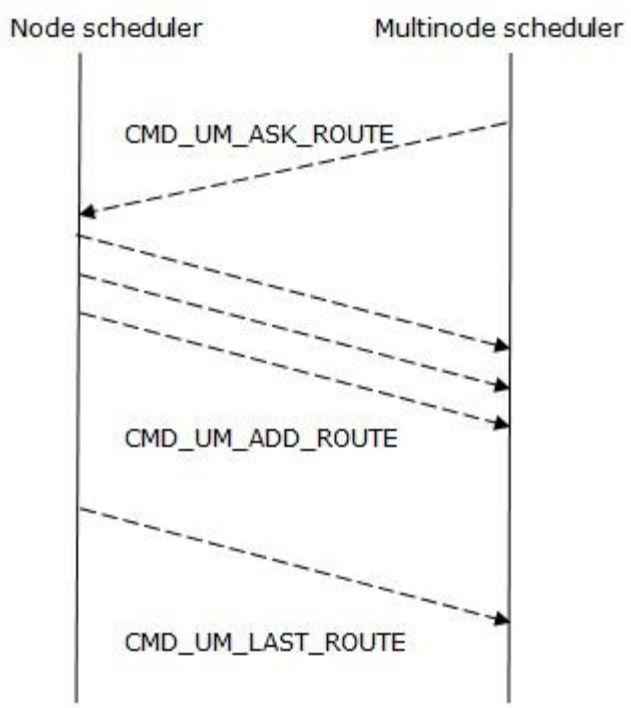


Figure 4: Receiving a CMD_UM_ASK_ROUTE message from the multinode scheduler

3.1.5.9 Receiving a CMD_ADM_CONF_REFRESH Message

After receiving a CMD_ADM_CONF_REFRESH message, a node scheduler MUST initiate a refresh of the specified crawl or subcrawl collection. For more details, see section 2.2.3.10.

3.1.5.10 Receiving a CMD_ADM_CONF_ADD Message

After receiving a CMD_ADM_CONF_ADD message, a node scheduler MUST process the message as follows:

- If the specified crawl collection already exists, the node scheduler MUST treat this command as a CMD_ADM_CONF_UPDATE command, as specified in section 2.2.3.11.
- Otherwise, the node scheduler MUST set up all the data structures that are necessary to initiate a crawl of the specified crawl collection, and add the crawl collection to the crawl configuration structure.

3.1.5.11 Receiving a CMD_ADM_CONF_UPDATE Message

After receiving a CMD_ADM_CONF_UPDATE message, a node scheduler MUST determine whether the specified crawl collection exists. If it does not exist, the node scheduler MUST ignore the update. Otherwise:

- Based on the updated crawl configuration, the node scheduler MUST reinitialize the crawl rules and other configurable data structures, and update the crawl configuration structure.
- If the crawl configuration specifies duplicate servers, the node scheduler MUST send a PPDUP_SET_CONFIG message to each of those servers. The value associated with the PKT_DATA

key in this message MUST contain only the **ppdup** section of each crawl collection as specified in [MS-FSCCFG] section 2.2.4.8. The node scheduler MUST then change the status of each duplicate server to reflect the fact that the duplicate server was updated with the latest configuration. The PKT_DSPECNAME key MUST be included, and its associated value MUST specify the crawl collection from the original CMD_ADM_CONF_UPDATE message.

- The node scheduler MUST requeue all the start URIs that are specified by the crawl configuration to the crawl queue.

3.1.5.12 Receiving a CMD_ADM_CONF_REMOVE Message

After receiving a CMD_ADM_CONF_REMOVE message, a node scheduler MUST process the message as follows:

1. The node scheduler MUST stop the crawling of all the active crawl sites for the specified crawl collection.
2. After the crawling stops, the node scheduler MUST remove all the existing information regarding the crawl collection from the crawl store and the crawl configuration structure.
3. The node scheduler MUST send a PPDUP_CONF_REMOVE message to each duplicate server that is configured by the crawl collection. To avoid having the duplicate servers spend unnecessary time on processing messages for this crawl collection, the node scheduler MUST add this message to the front of each communication queue so that the messages reach the duplicate servers as soon as possible.
4. The node scheduler MUST send a PPDUP_CONF_REMOVE2 message to each duplicate server. The node scheduler MUST add this message to the end of each communication queue. (When a duplicate server receives this message, it can perform a final cleanup.)
5. The node scheduler MUST send a CMD_UM_DELETE_OK reply to the multinode scheduler.

The following figure illustrates the communication between the multinode scheduler and the node scheduler.

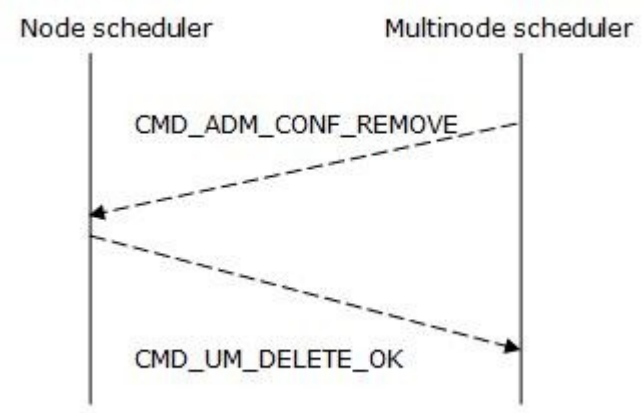


Figure 5: Receiving a CMD_ADM_CONF_REMOVE message from the multinode scheduler

3.1.5.13 Receiving a CMD_ADM_PREEMPT_SITE Message

After receiving a CMD_ADM_PREEMPT_SITE message, a node scheduler MUST determine whether the specified crawl site is currently being crawled. If not, the node scheduler MUST ignore the message. Otherwise:

1. The node scheduler MUST stop the crawling of the specified crawl site.

2. The node scheduler MUST queue the crawl site to the crawl queue that is specified by the message. For more details, see section 2.2.3.14.

3.1.5.14 Receiving a CMD_UM_REPROCESS_SITE Message

After receiving a CMD_UM_REPROCESS_SITE message, a node scheduler MUST resubmit all the crawl documents that match the message specifications to the content-indexing process. For more details, see section 2.2.3.15.

3.1.5.15 Receiving a CMD_UM_DELETE_SITE Message

After receiving a CMD_UM_DELETE_SITE message, a node scheduler MUST delete all the information that is stored for the specified crawl site and send remove operations to the content-indexing process for all the documents that were previously indexed. For more details, see [MS-FSCF] section 2.2.41.

3.1.5.16 Receiving a CMD_UM_DELETE_URIS Message

After receiving a CMD_UM_DELETE_URIS message, a node scheduler MUST delete all the specified URIs from local storage and send remove operations to the content-indexing process for all the documents that were previously indexed. For more details, see [MS-FSCF] section 2.2.41.

3.1.5.17 Receiving a CMD_UM_QUARANTINE_SITE Message

After receiving a CMD_UM_QUARANTINE_SITE message, a node scheduler MUST process the message as follows:

1. If the specified crawl site is currently being crawled, the node scheduler MUST stop the crawling.

2. The node scheduler MUST add the crawl site to the quarantine list and block the crawling of that crawl site for the duration of the specified time period.

3. After the time period has expired, the node scheduler MUST remove the crawl site from the quarantine list and allow it to be crawled again.

3.1.5.18 Receiving a CMD_UM_QUARANTINE_SITE_REQUEUE Message

After receiving a CMD_UM_QUARANTINE_SITE_REQUEUE message, a node scheduler MUST follow the same steps as those for the CMD_UM_QUARANTINE_SITE message as specified in section 3.1.5.17, with a final, additional step to enqueue the crawl site to the front of the crawl queue.

3.1.5.19 Receiving a CMD_UM_DNS_REPLY Message

After receiving a CMD_UM_DNS_REPLY message, a node scheduler MUST store the result of the DNS resolution operation in the DNS database.

3.1.5.20 Receiving a CMD_UM_CONF_STATE Message

After receiving a CMD_UM_CONF_STATE message, a node scheduler MUST change the state of the specified crawl collection to the state that is specified by the message.

3.1.5.21 Receiving a CMD_UM_START_CRAWL_IP Message

After receiving a CMD_UM_START_CRAWL_IP message, a node scheduler MUST process the message as follows:

- If the request to crawl the site at the specified IP address was not approved as specified in section 2.2.3.22, the node scheduler MUST temporarily block the crawling of the crawl site for 30 minutes and enqueue that crawl site to the front of the crawl queue.
- If the request was approved, the node scheduler MUST add the crawl site to the crawl site IP addresses list of crawl sites that are currently crawling the site at the specified IP address. If more than 10 crawl sites are crawling the same IP address, the node scheduler MUST temporarily block the crawling of that crawl site for 30 minutes and enqueue it to the front of the crawl queue.

3.1.5.22 Receiving a PPDUP_CONFIG_ACK Message

After receiving a PPDUP_CONFIG_ACK message, a node scheduler MUST verify that the message is a reply to the most-recent PPDUP_SET_CONFIG message that it sent. To do so, the node scheduler checks that the value associated with the PKT_VCLOCK key in the PPDUP_CONFIG_ACK message is the same as the value of the node scheduler's internal vector clock. For more details, see section 2.2.5.11.

If the PPDUP_CONFIG_ACK message is such a reply, the node scheduler MUST update its duplicate server status structure to indicate that the duplicate server has been updated. If not, the node scheduler MUST discard the PPDUP_CONFIG_ACK command message.

3.1.5.23 Receiving a PPDUP_ADD_OK Message

After receiving a PPDUP_ADD_OK message, a node scheduler MUST store the specified URI in the checksum database of the node scheduler.

The following figure shows the sequence of messages that includes the PPDUP_ADD_OK message.

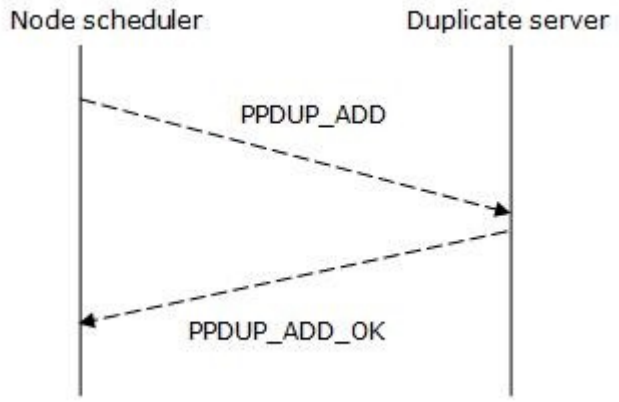


Figure 6: Receiving a PPDUP_ADD_OK message

3.1.5.24 Receiving a PPDUP_REMOVE_OK Message

After receiving a PPDUP_REMOVE_OK message, a node scheduler MUST remove the specified URI from the checksum database of the node scheduler.

3.1.5.25 Receiving a PPDUP_PROMOTE_REQ Message

After receiving a PPDUP_PROMOTE_REQ message, a node scheduler MUST recrawl all the URIs that have the same checksum as the value associated with the PKT_CSUM key in the message. For more details, see section 2.2.5.8.

3.1.5.26 Receiving a PPDUP_ERROR Message

After receiving a PPDUP_ERROR message, a node scheduler MUST log the specified error message. For more details, see section 2.2.5.2.

3.1.5.27 Receiving a PPDUP_KEEPALIVE Message

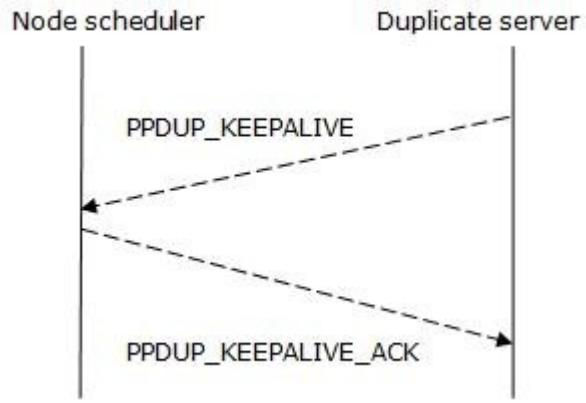


Figure 7: Receiving a PPDUP_KEEPALIVE message

After receiving a PPDUP_KEEPALIVE message, a node scheduler MUST respond with a PPDUP_KEEPALIVE_ACK message as specified in section 2.2.5.7.

3.1.6 Timer Events

The **Multinode Scheduler Keep-Alive Timeout** event logs if the multinode scheduler has not returned a CMD_UM_KEEPALIVE_ACK message since the previous **Multinode Scheduler Keep-Alive Timeout** event, and sends a CMD_UM_KEEPALIVE message to the multinode scheduler.

The **Duplicate Server Keep-Alive Timeout** event logs duplicate servers that have not returned a PPDUP_KEEPALIVE_ACK message since the previous **Duplicate Server Keep-Alive Timeout** event, sends a PPDUP_KEEPALIVE message to the currently connected duplicate servers, and adds an entry for each of those messages to the pending keep-alive list.

The **Quarantine Timeout** traverses the **Quarantine list** to check if the quarantine of any of the entries is expired, and removes the entries that are.

The **Statistics Timeout** issues CMD_UM_STAT messages to the multinode scheduler with the delta of statistics since the previous **Statistics Timeout** event.

3.1.7 Other Local Events

None.

3.2 Multinode Scheduler Details

A multinode scheduler communicates with the individual node schedulers. It manages the crawl of the configured crawl collections and distributes the work to the node schedulers.

A multinode scheduler receives new crawl collection configurations as specified in [MS-FSCCFG], from the administration API as specified in [MS-FSCADM], and relays this information to all node schedulers.

Specified start URIs for each configured crawl collection are queued to the crawl queue. The multinode scheduler then extracts entries from the crawl queue and selects a node scheduler to handle the specific crawl site. This routing decision is sent to every node scheduler, and the crawl queue entries are then sent to the selected node scheduler.

The multinode scheduler receives crawl queue entries for new crawl sites that are not yet routed from the node schedulers. The crawl queue entries are queued to the crawl queue before being extracted and, if already routed, forwarded to the responsible node scheduler. Otherwise, a routing decision is made first.

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.

Crawl configuration: A tracking of all the active crawl collections and their respective crawl configurations, as specified in section 2.2.2.3.

Crawl site IP addresses: A list of all the IP addresses that are actively being crawled, and for each IP address, which node scheduler is crawling it.

Crawl statistics: The crawl statistics, as specified in [MS-FSCADM] section 2.2.5, for each active crawl collection.

Crawl queue: A queue of URIs to be crawled for each crawl collection.

Node scheduler list: A list of all the registered node schedulers.

Routing database: A database of all the routing decisions that were made.

3.2.2 Timers

The Node Scheduler Keep-Alive timer issues CMD_UM_KEEPALIVE messages to the connected node schedulers to keep the connections alive and detect connection failures. The default value is 120 seconds.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

3.2.5.1 Receiving a CMD_UM_INIT Message

The first message that the multinode scheduler receives over a connection (2) MUST be a CMD_UM_INIT message to register a node scheduler with the multinode scheduler. If the multinode scheduler receives any other message before a CMD_UM_INIT message, the multinode scheduler MUST ignore that message and close the connection (2).

After receiving a CMD_UM_INIT message, the multinode scheduler MUST process the message as follows:

1. If the specified node identifier already is already registered with the multinode scheduler, the multinode scheduler MUST ignore the message and close the connection (2). Otherwise, the multinode scheduler MUST register the connecting node scheduler in the node scheduler list.

2. The multinode scheduler MUST send a CMD_ADM_CONF_ADD message for each crawl collection that is registered to the connecting node scheduler.

3.2.5.2 Receiving a CMD_UM_KEEPALIVE Message

After receiving a CMD_UM_KEEPALIVE message, the multinode scheduler MUST respond with a CMD_UM_KEEPALIVE_ACK message as specified in section 2.2.3.2.

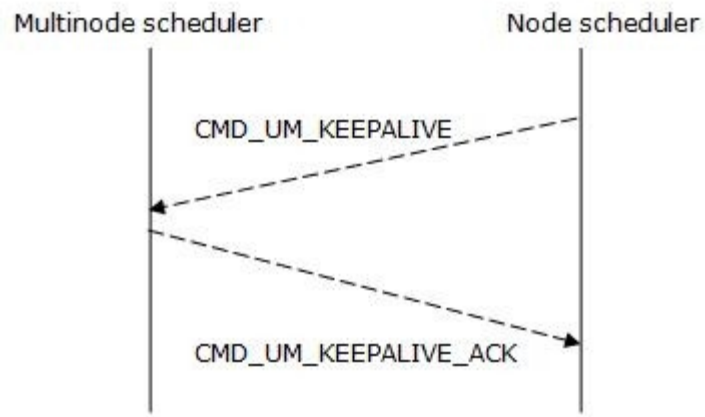


Figure 8: Receiving a CMD_UM_KEEPALIVE request from a node scheduler

3.2.5.3 Receiving a CMD_UM_KEEPALIVE_ACK Message

After receiving a CMD_UM_KEEPALIVE_ACK message, the multinode scheduler MUST remove the keep-alive time entry that it recorded for this connection (2).

3.2.5.4 Receiving a CMD_UM_DNS_REQUEST Message

After receiving a CMD_UM_DNS_REQUEST message, the multinode scheduler MUST process the message as follows:

1. The multinode scheduler MUST check its DNS database for the existence of the DNS lookup result and verify that the result is not expired.

2. If the host name does not exist or is expired, the multinode scheduler MUST attempt to resolve the host name.

3. The multinode scheduler MUST send a CMD_UM_DNS_REPLY reply message back to the node scheduler and fill in the **addrinfo** structure as follows:

4. If the multinode scheduler successfully resolves the host name, it MUST fill in the **addrinfo** structure.

5. If the multinode scheduler cannot resolve the host name, it MUST set the **addrinfo** structure to **None**.

The message exchange is illustrated in the following figure.

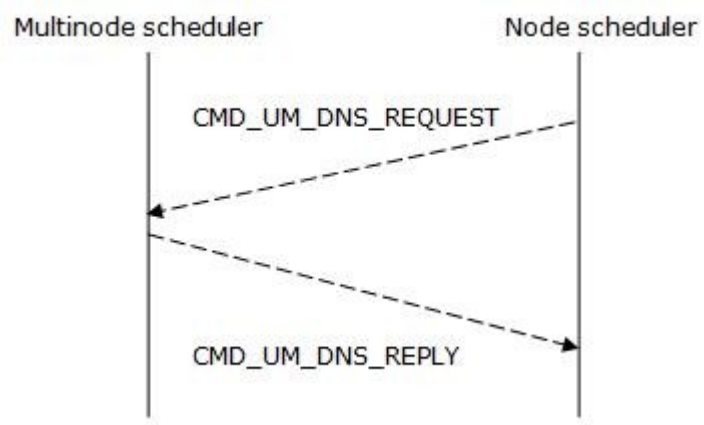


Figure 9: Receiving a CMD_UM_DNS_REQUEST message from a node scheduler

3.2.5.5 Receiving a CMD_UM_URI Message

After receiving a CMD_UM_URI message, the multinode scheduler MUST process the message as follows:

1. The multinode scheduler MUST enqueue the specified crawl queue entry to the back of the local crawl queue.

2. When the multinode scheduler later extracts the crawl queue entry from the crawl queue, it MUST process that entry as follows:

    1. If the crawl site was not already routed, the multinode scheduler MUST specify the node scheduler to handle that site. This crawl routing decision MUST persist in the crawl routing database.

    2. The multinode scheduler MUST send the crawl routing decision to all the registered node schedulers by means of a CMD_UM_ADD_ROUTE message. For more details, see section 2.2.3.8.

    3. The multinode scheduler MUST forward the crawl queue entry to the assigned node scheduler by means of a CMD_UM_URI message. For more details, see section 2.2.3.3.

3.2.5.6 Receiving a CMD_UM_STAT Message

After receiving a CMD_UM_STAT message, the multinode scheduler MUST update the crawl statistics database with the statistics that are specified in the message.

3.2.5.7 Receiving a CMD_UM_URI_URG Message

After receiving a CMD_UM_URI_URG message, the multinode scheduler MUST handle the message in the same way that it handles a CMD_UM_URI message as specified in section 3.2.5.5, except that it MUST enqueue the URI to the front of the local crawl queue rather than the back.

3.2.5.8 Receiving a CMD_UM_LOG Message

After receiving a CMD_UM_LOG message, the multinode scheduler MUST append the specified log message to the specified log file.

3.2.5.9 Receiving a CMD_UM_ADD_ROUTE Message

After receiving a CMD_UM_ADD_ROUTE message, the multinode scheduler MUST update the routing database with the specified routing information.

3.2.5.10 Receiving a CMD_UM_LAST_ROUTE Message

After receiving a CMD_UM_LAST_ROUTE message, the multinode scheduler MUST resume the normal crawl routing mode for the specified crawl collection.

3.2.5.11 Receiving a CMD_UM_START_CRAWL_IP Message

After receiving a CMD_UM_START_CRAWL_IP message, the multinode scheduler MUST process the message as follows:

- If the multinode scheduler assigns the IP address to a node scheduler other than the one that sends the message (according to the list of crawl site IP addresses), the multinode scheduler MUST respond with a CMD_UM_START_CRAWL_IP message in which the value associated with the PKT_DATA key specifies not allowed. For more details, see section 2.2.3.22.
- If the multinode scheduler did not yet assign the IP address or if it assigned the IP address to the requesting node scheduler, the multinode scheduler MUST respond with a CMD_UM_START_CRAWL_IP message in which the value associated with the PKT_DATA key specifies allowed.

The message exchange is illustrated in the following figure.

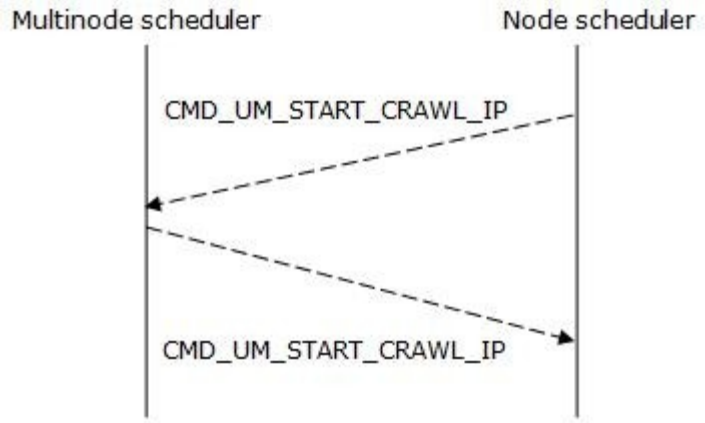


Figure 10: Receiving a CMD_UM_START_CRAWL_IP message from a node scheduler

3.2.5.12 Receiving a CMD_UM_STOP_CRAWL_IP Message

After receiving a CMD_UM_STOP_CRAWL_IP message, the multinode scheduler MUST remove the corresponding record from its list of crawl site IP addresses.

3.2.5.13 Receiving a CMD_UM_CONF_STATE Message

After receiving a CMD_UM_CONF_STATE message, the multinode scheduler MUST respond with a CMD_UM_CONF_STATE message that specifies the current state of the crawl collection.

The message exchange is illustrated in the following figure.

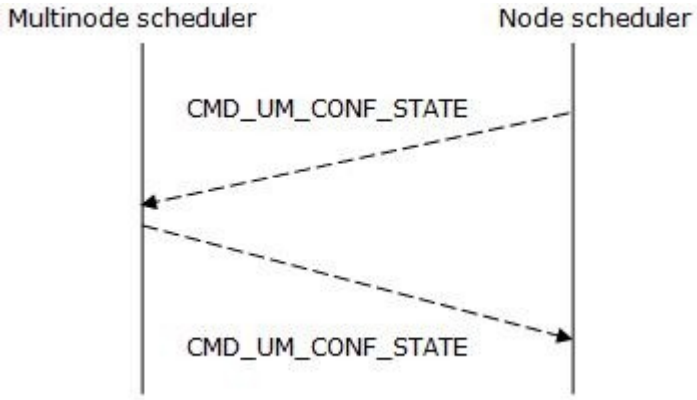


Figure 11: Receiving a CMD_UM_CONF_STATE message from a node scheduler

3.2.5.14 Receiving a CMD_UM_DELETE_OK Message

After receiving a CMD_UM_DELETE_OK message, the multinode scheduler MUST remove any remaining state information that is associated with the specified crawl collection.

3.2.6 Timer Events

The **Node Scheduler Keep-Alive Timeout** event logs the node scheduler that has not returned a CMD_UM_KEEPALIVE_ACK message since the previous **Node Scheduler Keep-Alive Timeout** event, and sends a CMD_UM_KEEPALIVE message to all the currently connected node schedulers, and adds an entry for each of those messages to the pending keep-alive list.

3.2.7 Other Local Events

None.

3.3 Duplicate Server Details

A duplicate server communicates with the node schedulers, and any connected duplicate server replica. It maintains a mapping of all document checksums it registers, and keeps this mapping in sync with the adding and removing of document checksums from the node schedulers. A node scheduler is only allowed a successful add of a document checksum if it was not added before, or the node scheduler already owns it.

If a duplicate server is configured with a duplicate server replica, the duplicate server replicates all document checksums to the duplicate server replica by forwarding all messages it receives from the node scheduler, before responding to the node scheduler.

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.

Checksum store: A store for each crawl collection that contains a mapping from each checksum to the URI and node identifier that own the checksum.

Connection status structure: A structure that maintains the status of the node schedulers and any duplicate server replica that the duplicate server communicates with.

Crawl configurations: A tracking of all the active crawl collections and their respective crawl configurations, as specified in section 2.2.2.3.

Pending checksum operations: A tracking of all the pending checksum operations that have not been committed to the checksum store on both the duplicate server and any duplicate server replica.

Pending keep-alive list: A list of all the outstanding PPDUP_KEEP_ALIVE request messages that have not received a corresponding PPDUP_KEEPALIVE_ACK message.

Vector clocks: A vector clock for the duplicate server replica, if one exists for the duplicate server.

3.3.2 Timers

The Keep-Alive timer issues a PPDUP_KEEPALIVE message to the node scheduler and any connected duplicate server replica, to keep the connections alive and detect connection failures. The default value is 120 seconds.

3.3.3 Initialization

None.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

3.3.5.1 Receiving a PPDUP_SET_CONFIG Message

After receiving a PPDUP_SET_CONFIG message, a duplicate server MUST process the message as follows:

1. The duplicate server MUST add the specified crawl configuration to its crawl configuration state.
2. The duplicate server MUST immediately send a PPDUP_CONFIG_ACK message to the node scheduler from which it originally received the message.
3. If the duplicate server has a replica associated with it, the duplicate server MUST forward the message to the duplicate server replica as follows:
    1. The duplicate server MUST change the state of the duplicate server replica to a transition state.
    2. When the duplicate server receives the first checksum message from a node scheduler after the duplicate server has applied the transition state, the duplicate server MUST forward the PPDUP_SET_CONFIG message to the duplicate server replica before forwarding the checksum message. The duplicate server MUST forward subsequent checksum messages normally.
    3. After the duplicate server receives a PPDUP_CONFIG_ACK message from the replica, the duplicate server MUST change the state of the duplicate server replica back to normal, as specified in section 3.3.5.8.

The following figure illustrates the message sequence.

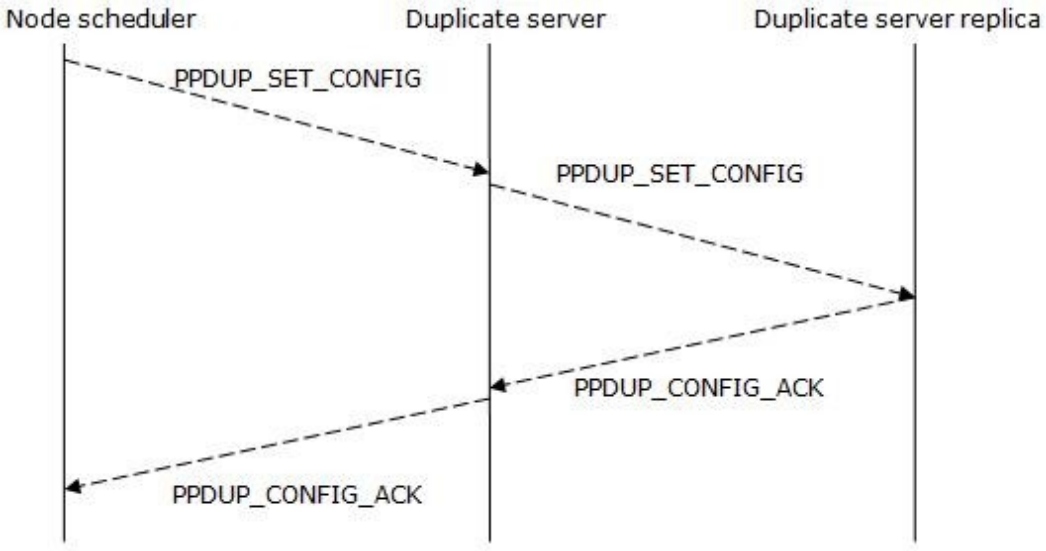


Figure 12: Receiving a PPDUP_SET_CONFIG message from a node scheduler

3.3.5.2 Receiving a PPDUP_CONF_REMOVE Message

After receiving a PPDUP_CONF_REMOVE message, a duplicate server MUST process the message as follows:

1. The duplicate server MUST delete the checksum store of the crawl collection and change the crawl configuration state to indicate that removal is in progress.

2. While the crawl configuration is in this state, the duplicate server MUST discard any subsequent messages that it receives for this crawl collection. (Note that the duplicate server MUST remove any remaining state information for this crawl collection when it receives the final PPDUP_CONF_REMOVE2 message as specified in section 3.3.5.3.)

3. If the duplicate server has a replica associated with it, the duplicate server MUST forward the message to the duplicate server replica. The forwarded message MUST include the PKT_PPDUP_ID key and its associated value, which specifies the node identifier of the duplicate server.

3.3.5.3 Receiving a PPDUP_CONF_REMOVE2 Message

After receiving a PPDUP_CONF_REMOVE2 message, a duplicate server MUST remove the specified crawl configuration from its crawl configuration state.

The following figure illustrates the message sequence for the PPDUP_CONF_REMOVE and PPDUP_CONF_REMOVE2 messages.

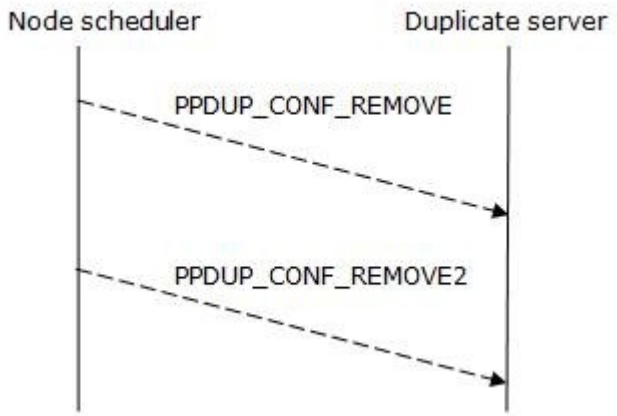


Figure 13: Receiving a PPDUP_CONF_REMOVE2 message

3.3.5.4 Receiving a PPDUP_ADD Message

After receiving a PPDUP_ADD message, a duplicate server MUST process the message as specified in this section.

If the duplicate server has no mapping for the checksum specified in the message but does have an associated replica, the duplicate server MUST perform the following steps:

1. If the duplicate server has recently forwarded a PPDUP_SET_CONFIG message to the duplicate server replica but has not yet received a PPDUP_CONFIG_ACK reply message, the duplicate

server MUST send a PPDUP_SET_CONFIG message to the replica and create a new checksum mapping.

2. The duplicate server MUST forward the PPDUP_ADD message to the duplicate server replica. The forwarded message MUST include:

3. The node identifier of the duplicate server as the value associated with the PKT_PPDUP_ID key.

4. The path of the duplicate server database for the specified crawl collection as the value associated with the PKT_DBNAME key.

5. If the duplicate server receives additional PPDUP_ADD messages containing the same checksum prior to receiving a PPDUP_ADD_OK reply message from the duplicate server replica, the duplicate server MUST queue these messages for processing until it receives the PPDUP_ADD_OK message.

6. After receiving a PPDUP_ADD_OK reply message from the duplicate server replica, the duplicate server MUST send a PPDUP_ADD_OK reply message back to the node scheduler from which it originally received the message. This message MUST include the PKT_OWNURI key.

If the duplicate server has neither a mapping for the specified checksum nor an associated replica, the duplicate server MUST perform the following steps:

1. Create a new checksum mapping.

2. Send a PPDUP_ADD_OK reply message back to the node scheduler from which it originally received the message. This message MUST include the PKT_OWNURI key.

If the duplicate server has a mapping for the specified checksum, the duplicate server MUST send a PPDUP_ADD_OK reply message back to the node scheduler. This message MUST include the PKT_OWNURI key.

The sequence of messages is illustrated by the following figure.

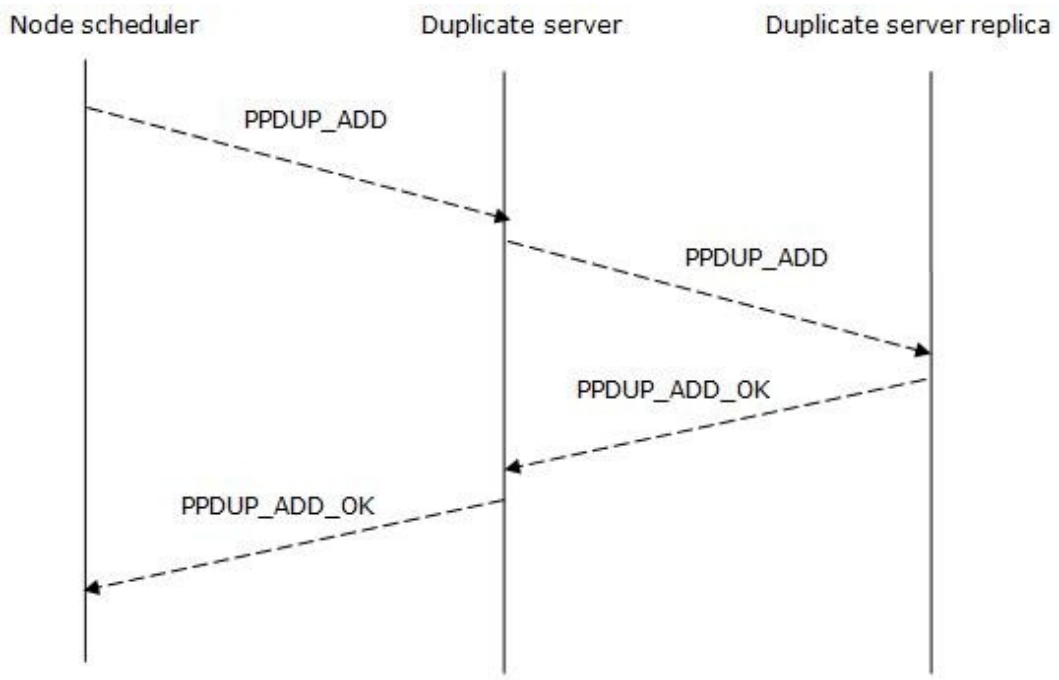


Figure 14: Receiving a PPDUP_ADD message from a node scheduler

3.3.5.5 Receiving a PPDUP_REMOVE Message

After receiving a PPDUP_REMOVE message, a duplicate server MUST process the message as follows:

1. The duplicate server MUST remove the checksum that is specified by the message from the checksum mapping.
2. If the duplicate server has an associated replica, the duplicate server MUST perform the following steps:
   - Forward the PPDUP_REMOVE message to the duplicate server replica. The forwarded message MUST include:
3. The node identifier of the duplicate server as the value associated with the PKT_PPDUP_ID key.
4. The path of the duplicate server database for the specified crawl collection as the value associated with the PKT_DBNAME key.
   - If the duplicate server receives additional PPDUP_REMOVE messages containing the same checksum prior to receiving a PPDUP_REMOVE_OK message from the replica, the duplicate server MUST queue these messages for processing until it receives a PPDUP_REMOVE_OK message.
5. The duplicate server MUST send a PPDUP_PROMOTE_REQ message to all the currently connected node schedulers that are associated with this crawl collection.
6. The duplicate server MUST send a PPDUP_REMOVE_OK reply message back to the node scheduler from which it originally received the message.

The sequence of messages is illustrated by the following figure.

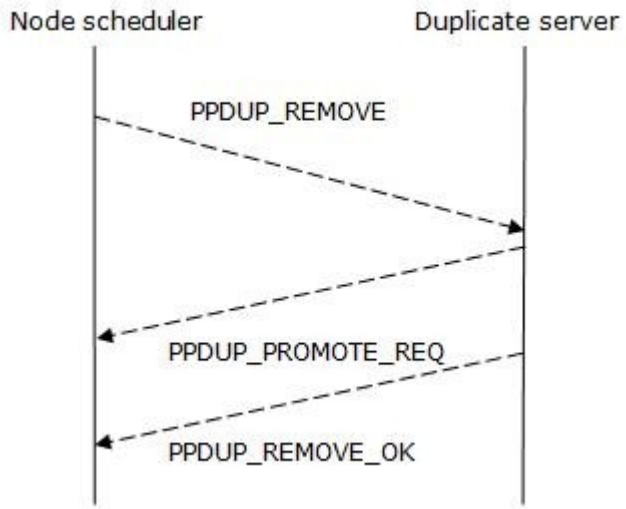


Figure 15: Sequencing of the PPDUP_REMOVE message without a duplicate server replica

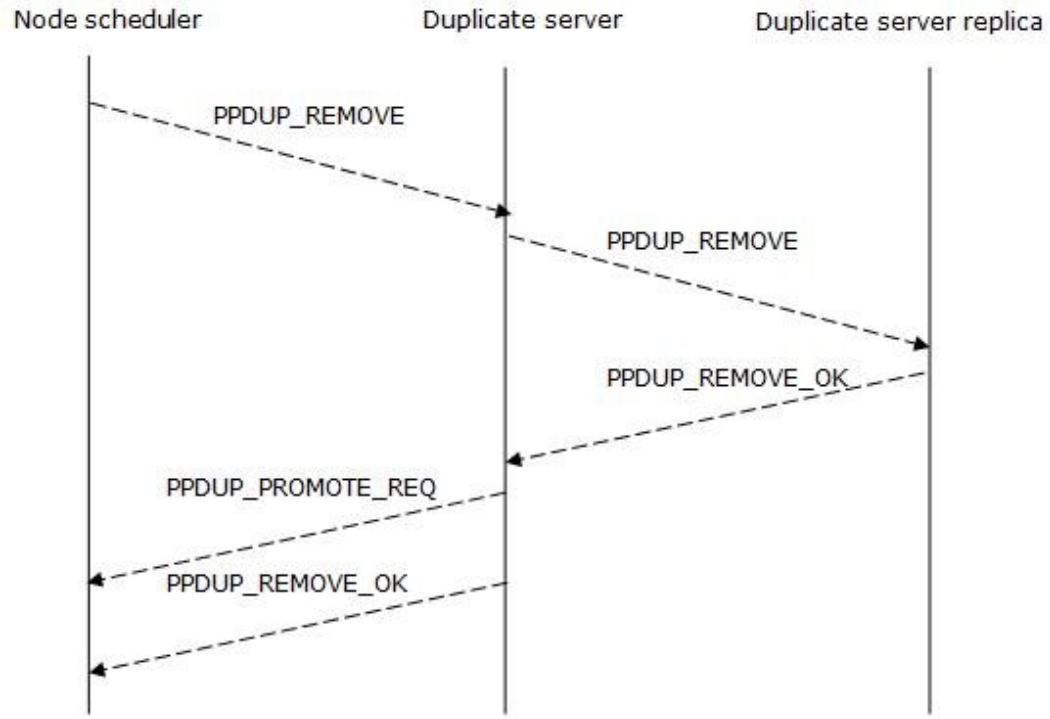


Figure 16: Receiving a PPDUP_REMOVE message from a node scheduler

3.3.5.6 Receiving a PPDUP_KEEPALIVE Message

After receiving a PPDUP_KEEPALIVE message, a duplicate server MUST send a PPDUP_KEEPALIVE_ACK reply message back to the node scheduler.

3.3.5.7 Receiving a PPDUP_ADD_OK Message

After receiving a PPDUP_ADD_OK message, a duplicate server MUST process the message as specified in section 3.3.5.4.

3.3.5.8 Receiving a PPDUP_CONFIG_ACK Message

After receiving a PPDUP_CONFIG_ACK message, a duplicate server MUST process the message as follows:

- If the value associated with the PKT_VCLOCK key in the message is valid, the duplicate server MUST update its state to record that the duplicate server replica received the most recent crawl configuration.
- Otherwise, the duplicate server MUST ignore this message.

3.3.5.9 Receiving a PPDUP_REMOVE_OK Message

After receiving a PPDUP_REMOVE_OK message, a duplicate server MUST process the message as specified in section 3.3.5.5.

3.3.5.10 Receiving a PPDUP_KEEPALIVE_ACK Message

After receiving a PPDUP_KEEPALIVE_ACK message, a duplicate server MUST remove the corresponding entry from the pending keep-alive list for the node scheduler or duplicate server replica that the message originated from.

3.3.6 Timer Events

The **Keep-Alive Timeout** event logs every node scheduler and duplicate server replica that has not returned a PPDUP_KEEPALIVE_ACK message since the previous **Keep-Alive Timeout** event, sends a PPDUP_KEEPALIVE message to all the currently connected node schedulers and duplicate server replicas, and adds an entry for each of those messages to the pending keep-alive list.

3.3.7 Other Local Events

None.

3.4 Duplicate Server Replica Details

A duplicate server replica communicates with a duplicate server. It receives the forwarded messages about new or removed checksums from the duplicate server and preserves them to the Checksum store such that it keeps an exact replica of the duplicate server Checksum store.

3.4.1 Abstract Data Model

A duplicate server replica has the same abstract data model as a duplicate server, as specified in section 3.3.1.

3.4.2 Timers

The Keep-Alive timer issues a PPDUP_KEEPALIVE message to a connected duplicate server to keep the connection alive and detect connection failures. The default value is 120 seconds.

3.4.3 Initialization

None.

3.4.4 Higher-Layer Triggered Events

None.

3.4.5 Message Processing Events and Sequencing Rules

3.4.5.1 Receiving a PPDUP_SET_CONFIG Message

After receiving a PPDUP_SET_CONFIG message, a duplicate server replica MUST process the message as follows:

1. The replica MUST add the specified crawl configuration to the crawl configuration state.

2. The replica MUST send a PPDUP_CONFIG_ACK reply message back to the duplicate server from which the message originated.

3.4.5.2 Receiving a PPDUP_CONF_REMOVE Message

After receiving a PPDUP_CONF_REMOVE message, a duplicate server replica MUST remove the specified crawl collection from all data structures, including those for the checksum store and crawl configurations.

3.4.5.3 Receiving a PPDUP_ADD Message

After receiving a PPDUP_ADD message, a duplicate server replica MUST process the message as follows:

1. The replica MUST add the specified checksum to the checksum store. If the checksum already exists in the store, the replica MUST overwrite it with the new data.

2. The replica MUST send a PPDUP_ADD_OK reply message containing the same key/value pairs as the original PPDUP_ADD message back to the duplicate server from which the message originated.

3.4.5.4 Receiving a PPDUP_REMOVE Message

After receiving a PPDUP_REMOVE message, a duplicate server replica MUST process the message as follows:

1. If the specified checksum exists in the checksum store, the replica MUST delete that checksum from the store.

2. The replica MUST send a PPDUP_REMOVE_OK reply message containing the same key/value pairs as the original PPDUP_REMOVE message back to the duplicate server from which the message originated.

3.4.5.5 Receiving a PPDUP_KEEPALIVE_ACK Message

After receiving a PPDUP_KEEPALIVE_ACK message, a duplicate server replica MUST remove the corresponding entry from the pending keep-alive list for the duplicate server from which the message originated.

3.4.6 Timer Events

The **Keep-Alive Timeout** event logs the duplicate server that has not returned a PPDUP_KEEPALIVE_ACK message since the previous **Keep-Alive Timeout** event, sends a PPDUP_KEEPALIVE message to all the currently connected duplicate server, and adds an entry for each of those messages to the pending keep-alive list.

3.4.7 Other Local Events

None.

4 Protocol Examples

4.1 Initializing a Connection

The following example data shows a CMD_UM_INIT message sent from the node scheduler to the multinode scheduler to initialize the connection (2) between them.

```
0000000: 00 00 00 55 28 02 00 00 00 7B 73 02 00 00 00 78
0000010: 70 28 02 00 00 00 73 10 00 00 00 68 6F 73 74 2E
0000020: 63 6F 6E 74 6F 73 6F 2E 63 6F 6D 69 10 27 00 00
0000030: 73 02 00 00 00 6D 69 73 0B 00 00 00 37 30 2D 38
0000040: 36 2D 31 32 2D 32 30 73 02 00 00 00 63 6D 69 18
0000050: 00 00 00 30 69 01 00 00 00
```

According to the message format for this protocol as specified in section 2.2.1, the first four bytes (`00 00 00 55`) specify the message length, which in this case corresponds to the decimal number 85.

Deserializing the message data produces a tuple of length 2 that specifies the message dictionary and priority, as specified in section 2.2.1.

The message dictionary contains the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_INIT
PKT_NODE_ID	"49-47-67-86"
PKT_XMLRPCPRT	A tuple of length 2 that contains the following entries: "host.contoso.com" 10000

4.2 Adding a New Route

The following example data contains a CMD_UM_ADD_ROUTE message sent from the multinode scheduler to a node scheduler to add a new route.

```
0000000: 00 00 00 E0 28 02 00 00 00 7B 73 02 00 00 00 64
0000010: 6E 73 07 00 00 00 65 78 61 6D 70 6C 65 73 02 00
0000020: 00 00 70 64 28 05 00 00 00 73 0B 00 00 00 37 30
0000030: 2D 38 36 2D 31 32 2D 32 30 73 0B 00 00 00 63 6F
0000040: 6E 74 6F 73 6F 2E 63 6F 6D 73 0F 00 00 00 77 77
0000050: 77 2E 63 6F 6E 74 6F 73 6F 2E 63 6F 6D 5B 02 00
0000060: 00 00 28 05 00 00 00 69 02 00 00 00 69 01 00 00
0000070: 00 69 00 00 00 00 73 00 00 00 00 28 02 00 00 00
0000080: 73 0E 00 00 00 32 30 37 2E 34 36 2E 32 33 32 2E
0000090: 31 38 32 69 00 00 00 00 28 05 00 00 00 69 02 00
00000a0: 00 00 69 01 00 00 00 69 00 00 00 00 73 00 00 00
00000b0: 00 28 02 00 00 00 73 0D 00 00 00 32 30 37 2E 34
00000c0: 36 2E 31 39 37 2E 33 32 69 00 00 00 00 69 80 51
00000d0: 01 00 73 02 00 00 00 63 6D 69 07 00 00 00 30 69
00000e0: 00 00 00 00
```

According to the message format for this protocol as specified in section 2.2.1, the first four bytes (00 00 00 E0) specify the message length, which in this case corresponds to the decimal number 224.

Deserializing the message data produces a tuple of length 2 that specifies the message dictionary and priority, as specified in section 2.2.1.

The message dictionary contains the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_ADD_ROUTE
PKT_DSPECNAME	"example"
PKT_DATA	A tuple of length 5 that contains the following entries: "49-47-67-86" "contoso.com" "www.contoso.com" An array of **addrinfo** structures for the crawl site "www.contoso.com" 86400

4.3 Looking Up a Host Name

This example shows a message command sequence, as specified in section 3.2.5.4, in which a node scheduler sends a CMD_UM_DNS_REQUEST message as specified in section 2.2.4.4 to request the lookup of a host name, and the multinode scheduler responds with the lookup result in a CMD_UM_DNS_REPLY message as specified in section 2.2.3.20.

Request

```
0000000: 00 00 00 46 28 02 00 00 00 7B 73 02 00 00 00 64
0000010: 6E 73 07 00 00 00 65 78 61 6D 70 6C 65 73 02 00
0000020: 00 00 63 6D 69 0C 00 00 00 73 02 00 00 00 73 74
0000030: 73 0F 00 00 00 77 77 77 2E 63 6F 6E 74 6F 73 6F
0000040: 2E 63 6F 6D 30 69 01 00 00 00
```

According to the message format for this protocol as specified in section 2.2.1, the first four bytes (00 00 00 46) specify the message length, which in this case corresponds to the decimal number 70.

Deserializing the message data produces a tuple of length 2 that specifies the message dictionary and priority, as specified in section 2.2.1.

The message dictionary contains the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_DNS_REQUEST
PKT_DSPECNAME	"example"
PKT_SITE	"www.contoso.com"

After the multinode scheduler has successfully looked up the "www.contoso.com" host name, it responds with the following data, which contains a CMD_UM_DNS_REPLY message.

Response

```
0000000: 00 00 00 C3 28 02 00 00 00 7B 73 02 00 00 00 64
0000010: 6E 73 00 00 00 00 73 02 00 00 00 70 64 28 02 00
0000020: 00 00 73 0F 00 00 00 77 77 77 2E 63 6F 6E 74 6F
0000030: 73 6F 2E 63 6F 6D 28 03 00 00 00 5B 02 00 00 00
0000040: 28 05 00 00 00 69 02 00 00 00 69 01 00 00 00 69
0000050: 00 00 00 00 73 00 00 00 00 28 02 00 00 00 73 0D
0000060: 00 00 00 32 30 37 2E 34 36 2E 31 39 37 2E 33 32
0000070: 69 00 00 00 00 28 05 00 00 00 69 02 00 00 00 69
0000080: 01 00 00 00 69 00 00 00 00 73 00 00 00 00 28 02
0000090: 00 00 00 73 0E 00 00 00 32 30 37 2E 34 36 2E 32
00000a0: 33 32 2E 31 38 32 69 00 00 00 00 69 90 2C 36 4A
00000b0: 69 80 51 01 00 73 02 00 00 00 63 6D 69 0D 00 00
00000c0: 00 30 69 01 00 00 00
```

According to the message format for this protocol as specified in section 2.2.1, the first four bytes (`00 00 00 C3`) specify the message length, which in this case corresponds to the decimal number 195.

Deserializing the message data produces a tuple of length 2 that specifies the message dictionary and priority, as specified in section 2.2.1.

The message dictionary contains the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_DNS_REPLY
PKT_DSPECNAME	""
PKT_DATA	A tuple of length 2 containing the host name that was resolved and its corresponding **addrinfo** structures: "www.contoso.com" A tuple of length 3 that contains **addrinfo** structures, the time stamp at which the resolution took place, and the number of seconds until the host name needs to be resolved again: An array of **addrinfo** structures for the crawl site "www.contoso.com" 1245064336 86400

4.4 Adding a New Crawl Queue Entry

The following example data contains a CMD_UM_URI message sent from the multinode scheduler to the node scheduler to add a new crawl queue entry.

```
0000000: 00 00 01 50 28 02 00 00 00 7B 73 02 00 00 00 64
0000010: 6E 73 07 00 00 00 65 78 61 6D 70 6C 65 73 02 00
0000020: 00 00 70 64 28 09 00 00 00 73 03 00 00 00 57 51
0000030: 45 28 0E 00 00 00 73 17 00 00 00 68 74 74 70 3A
0000040: 2F 2F 77 77 77 2E 63 6F 6E 74 6F 73 6F 2E 63 6F
0000050: 6D 2F 73 04 00 00 00 68 74 74 70 73 0F 00 00 00
```

```
0000060: 77 77 77 2E 63 6F 6E 74 6F 73 6F 2E 63 6F 6D 73
0000070: 0F 00 00 00 77 77 77 2E 63 6F 6E 74 6F 73 6F 2E
0000080: 63 6F 6D 69 50 00 00 00 73 01 00 00 00 2F 73 00
0000090: 00 00 00 73 00 00 00 00 73 00 00 00 00 73 00 00
00000a0: 00 00 73 00 00 00 00 69 01 00 00 00 69 01 00 00
00000b0: 00 73 0F 00 00 00 77 77 77 2E 63 6F 6E 74 6F 73
00000c0: 6F 2E 63 6F 6D 28 0E 00 00 00 73 00 00 00 00 73
00000d0: 00 00 00 00 73 00 00 00 00 73 00 00 00 00 4E 73
00000e0: 00 00 00 00 73 00 00 00 00 73 00 00 00 00 73 00
00000f0: 00 00 00 73 00 00 00 00 73 00 00 00 00 69 01 00
0000100: 00 00 69 01 00 00 00 73 00 00 00 00 69 00 20 00
0000110: 00 69 00 00 00 00 5B 00 00 00 00 69 00 00 00 00
0000120: 69 FF FF FF FF 7B 69 10 00 00 00 4E 69 65 00 00
0000130: 00 73 0B 00 00 00 37 30 2D 38 36 2D 31 32 2D 32
0000140: 30 30 73 02 00 00 00 63 6D 69 19 00 00 00 30 69
0000150: 00 00 00 00
```

According to the message format for this protocol as specified in section 2.2.1, the first four bytes (`00 00 01 50`) specify the message length, which in this case corresponds to the decimal number 336.

Deserializing the message data produces a tuple of length 2 that specifies the message dictionary and priority, as specified in section 2.2.1.

The message dictionary contains the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	CMD_UM_URI
PKT_DSPECNAME	"example"
PKT_DATA	A crawl queue entry structure for the URI http://www.contoso.com/

4.5 Adding a Crawl Collection to a Duplicate Server

This example shows a message command sequence, as specified in section 3.3.5.1, in which a node scheduler sends a PPDUP_SET_CONFIG message as specified in section 2.2.5.10 to add a crawl collection to a duplicate server, and the duplicate server responds with a PPDUP_CONFIG_ACK message as specified in section 2.2.5.11.

Request

```
0000000: 00 00 00 77 28 02 00 00 00 7B 73 02 00 00 00 64
0000010: 6E 73 07 00 00 00 65 78 61 6D 70 6C 65 73 02 00
0000020: 00 00 76 63 69 01 00 00 00 73 02 00 00 00 70 64
0000030: 7B 73 07 00 00 00 65 78 61 6D 70 6C 65 7B 73 07
0000040: 00 00 00 63 6F 6D 70 61 63 74 69 00 00 00 00 73
0000050: 06 00 00 00 66 6F 72 6D 61 74 73 08 00 00 00 67
0000060: 69 67 61 62 61 73 65 30 30 73 02 00 00 00 63 6D
0000070: 69 12 00 00 00 30 69 01 00 00 00
```

According to the message format for this protocol as specified in section 2.2.1, the first four bytes (`00 00 00 77`) specify the message length, which in this case corresponds to the decimal number 119.

Deserializing the message data produces a tuple of length 2 that specifies the message dictionary and priority, as specified in section 2.2.1.

The message dictionary contains the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_SET_CONFIG
PKT_DSPECNAME	"example"
PKT_DATA	A dictionary containing the crawl configuration that applies to the duplicate server for the crawl collection "example"
PKT_VCLOCK	1

The duplicate server responds by sending a PPDUP_CONFIG_ACK message to the node scheduler.

Response

```
0000000: 00 00 00 4C 28 02 00 00 00 7B 73 02 00 00 00 76
0000010: 63 69 01 00 00 00 73 02 00 00 00 63 6D 69 13 00
0000020: 00 00 30 69 01 00 00 00 00 00 00 24 28 02 00 00
0000030: 00 7B 73 02 00 00 00 76 63 69 02 00 00 00 73 02
0000040: 00 00 00 63 6D 69 13 00 00 00 30 69 01 00 00 00
```

According to the message format for this protocol as specified in section 2.2.1, the first four bytes (`00 00 00 4c`) specify the message length, which in this case corresponds to the decimal number 76.

Deserializing the message data produces a tuple of length 2 that specifies the message dictionary and priority, as specified in section 2.2.1.

The message dictionary contains the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_CONFIG_ACK
PKT_VCLOCK	1

4.6 Adding a New Checksum to a Duplicate Server

This example shows a message command sequence, as specified in section 3.3.5.4., in which a node scheduler sends a PPDUP_ADD message to add a new checksum to a duplicate server as specified in section 2.2.5.3, and the duplicate server responds with a PPDUP_ADD_OK message as specified in section 2.2.5.1.

Request

```
0000000: 00 00 00 81 28 02 00 00 00 7B 73 02 00 00 00 63
0000010: 73 73 10 00 00 00 A9 EB CB 8F AC BD C7 43 77 5A
0000020: 1D 3B 52 2D A8 B6 73 02 00 00 00 64 6E 73 07 00
0000030: 00 00 65 78 61 6D 70 6C 65 73 02 00 00 00 69 64
0000040: 73 0B 00 00 00 37 30 2D 38 36 2D 31 32 2D 32 30
0000050: 73 02 00 00 00 63 6D 69 04 00 00 00 73 02 00 00
```

```
0000060: 00 75 72 73 17 00 00 00 68 74 74 70 3A 2F 2F 77
0000070: 77 77 2E 63 6F 6E 74 6F 73 6F 2E 63 6F 6D 2F 30
0000080: 69 00 00 00 00
```

According to the message format for this protocol as specified in section 2.2.1, the first four bytes (`00 00 00 81`) specify the message length, which in this case corresponds to the decimal number 129.

Deserializing the message data produces a tuple of length 2 that specifies the message dictionary and priority, as specified in section 2.2.1.

The message dictionary contains the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_ADD
PKT_DSPECNAME	"example"
PKT_ID	"70-86-12-20"
PKT_URI	"http://www.contoso.com/"
PKT_CSUM	The following byte string (in hexadecimal values): `A9 EB CB 8F AC BD C7 43 77 5A 1D 3B 52 2D A8 B6`

In response to the PPDUP_ADD message , the duplicate server sends a PPDUP_ADD_OK message to the node scheduler.

Response

```
0000000: 00 00 00 65 28 02 00 00 00 7B 73 02 00 00 00 64
0000010: 6E 73 07 00 00 00 65 78 61 6D 70 6C 65 73 02 00
0000020: 00 00 69 64 73 0B 00 00 00 37 30 2D 38 36 2D 31
0000030: 32 2D 32 30 73 02 00 00 00 63 6D 69 00 00 00 00
0000040: 73 02 00 00 00 75 72 73 17 00 00 00 68 74 74 70
0000050: 3A 2F 2F 77 77 77 2E 63 6F 6E 74 6F 73 6F 2E 63
0000060: 6F 6D 2F 30 69 00 00 00 00
```

According to the message format for this protocol as specified in section 2.2.1, the first four bytes (`00 00 00 65`) specify the message length, which in this case corresponds to the decimal number 101.

Deserializing the message data produces a tuple of length 2 that specifies the message dictionary and priority, as specified in section 2.2.1.

The message dictionary contains the key/value pairs that are listed in the following table.

Key name	Value
PKT_CMD	PPDUP_ADD_OK
PKT_DSPECNAME	"example"
PKT_ID	"70-86-12-20"

Key name	Value
PKT_URI	"http://www.contoso.com/"

5 Security

5.1 Security Considerations for Implementers

None.

5.2 Index of Security Parameters

None.

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.

<1> Section 2.2.1.1: If the vector clock detects an out-of-order message of type PPDUP_CONFIG_ACK, the message is ignored.

7 Change Tracking

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

8 Index

N

O

P

R

S

T

V