[MS-FSCADM]:
Crawler Administration and Status Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
11/06/2009	0.1	Major	Initial Availability
02/19/2010	1.0	Editorial	Revised and edited the technical content
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.
03/18/2011	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	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 Administration and Status Protocol. This protocol operates between a protocol client and a **Web crawler** server or server farm. The protocol enables the protocol client to make requests of the Web crawler, as well as query the Web crawler for status information. This protocol is a pure client/server protocol, where the administration process is the protocol client and the Web crawler process is the protocol server.

1.1 Glossary

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

attribute
checksum
Coordinated Universal Time (UTC)
Domain Name System (DNS)
file system
fully qualified domain name (FQDN)
Hypertext Transfer Protocol (HTTP)
Hypertext Transfer Protocol over Secure Sockets Layer (HTTPS)
Internet Protocol version 6 (IPv6)
IPv4 address in string format
IPv6 address in string format
path
Transmission Control Protocol (TCP)
UTF-8
XML

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

absolute URI
base port
connection
crawl collection
crawl limit
crawl queue
crawl refetch
crawl refresh cycle
crawl site
crawl statistics cycle
crawl subcollection
document
file
File Transfer Protocol (FTP)
hyperlink
MIME type
multinode scheduler
node
node scheduler
RSS channel
start URI
Uniform Resource Identifier (URI)
Web crawler
Web server

The following terms are specific to this document:

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

1.2 References

1.2.1 Normative References

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

[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-FSCCFG] Microsoft Corporation, "Crawler Configuration File Format Specification"

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

[MS-FSCX] Microsoft Corporation, "Configuration (XML-RPC) Protocol Specification"

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

[MS-FSXTAPI] Microsoft Corporation, "XML-RPC Translatable API Structure Specification"

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", STD 13, RFC 1035, November 1987, http://www.ietf.org/rfc/rfc1035.txt

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

[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

[RFC2696] Weider, C., Herron, A., Anantha, A., and Howes, T., "LDAP Control Extension for Simple Paged Results Manipulation", RFC 2696, September 1999, http://www.ietf.org/rfc/rfc2696.txt

[ROBOTSTXT] Koster, M., "A Method for Web Robots Control", November 1996, http://www.robotstxt.org/norobots-rfc.txt

[SITEMAPS] Sitemaps Org, "Sitemaps XML format", http://sitemaps.org/protocol.php

[XML-RPC] Winer, D., "XML-RPC Specification", June 1999, http://www.xmlrpc.com/spec

1.2.2 Informative References

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

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

1.3 Protocol Overview (Synopsis)

The Crawler Administration and Status protocol uses **XML** to encode methods and responses and uses **HTTP** as a transport mechanism. It provides a means for a protocol client to issue a set of administrative and status calls to a Web crawler application.

Communication occurs when the protocol client sends an XML-encapsulated request to the protocol server. The protocol server sends a similarly encoded XML response to the protocol client. The protocol server does not initiate communication with the protocol client. The protocol client is aware of the hostname and port of the protocol server for Web crawler requests. The crawler uses the configuration that is specified in the Crawler Configuration File Format Specification [MS-FSCCFG].

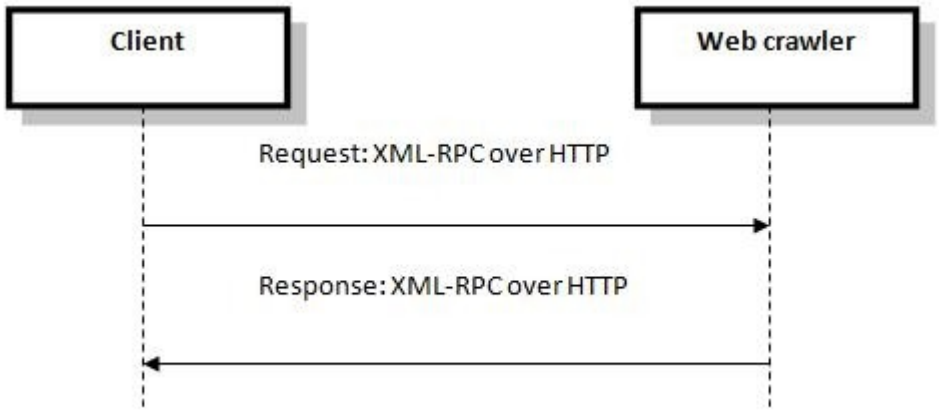

Figure 1: Communication overview for this protocol

The Web crawler supports two distinct protocol server roles. The first role is the **node scheduler** role, where the Web crawler manages the current **node** only. The second role is the **multinode scheduler** role, where the Web crawler manages multiple Web crawler nodes. The supported protocol methods and behaviors are dependent on which one of the two roles is active.

1.4 Relationship to Other Protocols

This protocol uses the [XML-RPC] protocol to format requests and responses. It transmits these messages using the HTTP protocol as specified in [RFC2616].

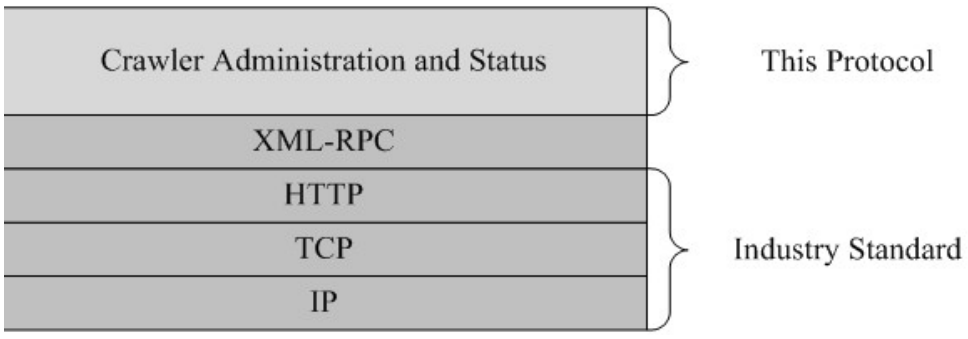

Figure 2: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

This protocol assumes that the protocol client has obtained the hostname and port of the protocol server prior to issuing a request.

1.6 Applicability Statement

None.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

This protocol uses the transport protocol specified in [XML-RPC].

2.2 Message Syntax

The format of the XML body requests and responses is specified in [MS-FSXTAPI]. The HTTP POST **path**, as specified in [RFC2616], MUST be "/RPC2".

2.2.1 Basic Types

The following sections specify the basic types used within this protocol.

2.2.1.1 int

An **int** is a 32 bit signed integer.

2.2.1.2 long

A **long** is a 64-bit signed integer.

2.2.1.3 double

A **double** is a double-point number that MUST adhere to the standard specified in [IEEE754]. Only signed values within a double-precision 64-bit range are supported.

2.2.1.4 string

A **string** is a **UTF-8** encoded string literal.

2.2.1.5 timestamp

A **timestamp** is an **int** or **double** and MUST specify time in seconds since 1970-01-01 00:00:00 **UTC**.

2.2.2 Complex Types

The following sections specify the complex types used by this protocol.

2.2.2.1 cresult

A **cresult** is an array that contains exactly two elements. It specifies a return value of success or failure for methods that require it.

When a method finishes successfully, it sets the first element to an integer value of 1. If a method fails, it sets the first element to less than 1. The second element is a **string** that contains a textual description of the result of the method.

See section 4.2 for an example of how a **cresult** array is encoded in an [XML-RPC] response.

2.2.2.2 hostport

A **hostport** encapsulates a hostname and port number into a **string**.

The format of the **string** MUST be <hostname>:<port>. The hostname MUST be a hostname, a **fully qualified domain name (FQDN)**, an **IPv4 address in string format**, or an **IPv6 address in string format**. If the hostname is an **Internet Protocol version 6 (IPv6)** address, the hostname MUST be contained in brackets, for example, "[::1]". The specified hostname MUST resolve successfully on the local node.

2.2.2.3 contentdest

A **contentdest** represents the destination for a content submission. If a **contentdest** is an empty **string**, the protocol server uses the default configuration. Otherwise, a **contentdest** is either the name of a content submission destination as specified in [MS-FSCCFG] section 2.2.4.9 or it is the **string** "default:<content collection>" where <content collection> specifies a valid content collection name.

2.2.2.4 dictionary

A **dictionary** structure is composed of key-value pairs, where each key maps to exactly one value. The keys MUST be a **string** data type, and the values MUST be one of the following data types: **int**, **long**, **double**, **string** or **dictionary**.

In an [XML-RPC] request or response, a **dictionary** MUST be encoded as a structure that contains members. The **dictionary** keys and values are specified by the name and value XML tags that are specified in the associated data type.

See section 4.3, for an example of how a nested **dictionary** is encoded as a structure.

2.2.3 Simple Types

2.2.3.1 URI Skip Codes

URI Skip Codes specify the reason that the protocol server discarded a **URI** immediately after extracting it from a web document, without adding it to the **crawl queue**. The **URI skip code** is a two letter **string** that is specified in the following table.

Value	Meaning
nf	No follow: The URI was tagged as "NoFollow" by an HTML META tag as specified by [HTML], or was rejected because of the **RSS channel** crawl configuration settings, as specified in [MS-FSCCFG] section 2.2.4.30.
ch	Scheme: The URI scheme of the URI was not included in the schemes configured by the crawl configuration, as specified in [MS-FSCCFG].
ur	URI exclusion: The URI is not a URI to be crawled, as specified within the URI crawl rules in the crawl configuration. For more details, see [MS-FSCCFG].
ro	Robots disallow: The URI was excluded by a Disallow robots.txt directive, as specified in [ROBOTSTXT].
do	Hostname exclusion: The URI is not within the list of hostnames to crawl, as specified in the hostname crawl rules in the crawl configuration. For more details, see [MS-FSCCFG].

Value	Meaning
ic	Cache exclusion: The URI was present in an internal cache; therefore it has already been added on the crawl queue.
fo	Out of focus: The crawl configuration specified focused crawling and the URI was outside the focus. For more details, see [MS-FSCCFG] section 2.2.4.19.
de	Depth exceeded: The crawl configuration specifies a limited depth crawl and the URI exceeded this depth. For more details, see [MS-FSCCFG] section 2.2.4.28.

2.2.3.2 Web Document Skip Codes

These specify the reason a web document is discarded after downloading it from the remote **Web server**. The skip code values are specified in the following table.

Value	Meaning
he	Header error: The web document contains an HTTP header that is not valid.
mi	**MIME type**: The web document MIME type does not match any of the MIME types specified in the crawl configuration. For more details, see [MS-FSCCFG].
ti	Timeout: Download of the web document exceeded the timeout as specified in the crawl configuration. For more details, see [MS-FSCCFG].
tl	Too large: The web document exceeded the maximum size as specified in the crawl configuration. For more details, see [MS-FSCCFG].
hx	Header excluded: The Web server sent an HTTP header that contains a header line that is excluded by the crawl configuration. For more details, see [MS-FSCCFG].
en	Web document encoding: The encoding header specified in the HTTP header was not valid or was unsupported by the Web crawler. For more details, see [MS-FSCCFG].
cu	Chunk decode error: The web document encoding was chunked, as specified in [RFC2616], but the Web crawler could not be decoded it.
nr	No redirection target: The HTTP header contains an HTTP redirect, as specified in [RFC2616], but no redirect URI could be found or correctly parsed.
ni	No Index: The web document contained a "NoIndex" HTML META tag as specified by [HTML] and MUST NOT be indexed.
cs	Duplicate: The web document is identical to a previously crawled web document with a different URI.
fs	Out of focus: The crawl configuration specifies focused crawling and the web document is outside the focus. For more details on focused crawling, see [MS-FSCCFG] section 2.2.4.19.
in	Incomplete: The web document was only partially downloaded.
co	Connect failure: A socket **connection** to the Web server was not established.
ct	Connect timeout: A socket connection timed out.
ne	Network error: A network error occurred.
em	Empty web document: The Web server sent an empty web document.

Value	Meaning
eh	Empty HTTP header: The Web server sent an empty HTTP header.
rs	RSS No Index: The web document is an RSS channel feed and the crawl configuration specifies that RSS channels are not indexed separately. For more details, see [MS-FSCCFG] section 2.2.4.30.
ot	Other error: Error not specified by one of the previous categories.

2.2.3.3 Crawl Collection Integer Status Codes

These specify the state of **crawl collections**. The status code MUST be of type **int** and specified in the following table.

Value	Meaning
1	The crawl collection is either crawling or idle.
2	Reserved, MUST NOT be used.
3	The Web crawler is deleting the crawl collection.
4	Crawl collection is suspended.
5	Crawl collection is suspended because of the lack of free disk space on the local **file system**.
6	Crawl collection executing regular cleanup routine.
7	Reserved, MUST NOT be used.
8	Crawl collection is suspended because of a variable delay configuration. For details, see [MS-FSCCFG] section 2.2.4.23.

2.2.3.4 Crawl Collection String Status Codes

These specify the state of crawl collections. The status code MUST be of type **string** and specified in the following table.

Value	Meaning
crawling	The crawl collection is crawling.
suspended	The crawl collection is suspended. Crawling MUST NOT occur.
compacting	The Web crawler is performing disk storage cleanup. Crawling MUST NOT occur.
zombie	The crawl collection is being deleted. Crawling MUST NOT occur while the deletion is processing. A new crawl collection with the same name cannot be created while the delete process is executing.

2.2.3.5 Log Levels

Log level codes MUST be specified as a bitmask from the following table.

Name	Value
Info	0x00000004
Debug	0x00000008
Warning	0x00000010
Error	0x00000020
Critical	0x00000040
Verbose	0x00000080

2.2.3.6 Crawl Mode String Status Codes

These specify the crawl mode of the Web crawler when a crawl limit has been reached. The limits are determined by the crawl configuration, as specified in [MS-FSCCFG] section 2.2.4.18.

The status code MUST be of type **string** and specified in the following table. If no crawl limit has been reached this value MUST be the empty string.

Value	Meaning
Refreshing(limit=max_doc)	The maximum web document count has been reached.
Refreshing(limit=disk_free)	The amount of disk free space is less than the minimum.
Refreshing(limit=user)	The refreshing mode has been manually engaged, for example by the CollectionEnableRefreshOnly method in section 2.2.10.2.

2.2.4 Statistics Structure

The following figure specifies the data structure that the protocol server sends by using the **CollectionGetStatistics** method that is specified in section 2.2.10.5. Both the "dspec" and "global" keys MUST be present.

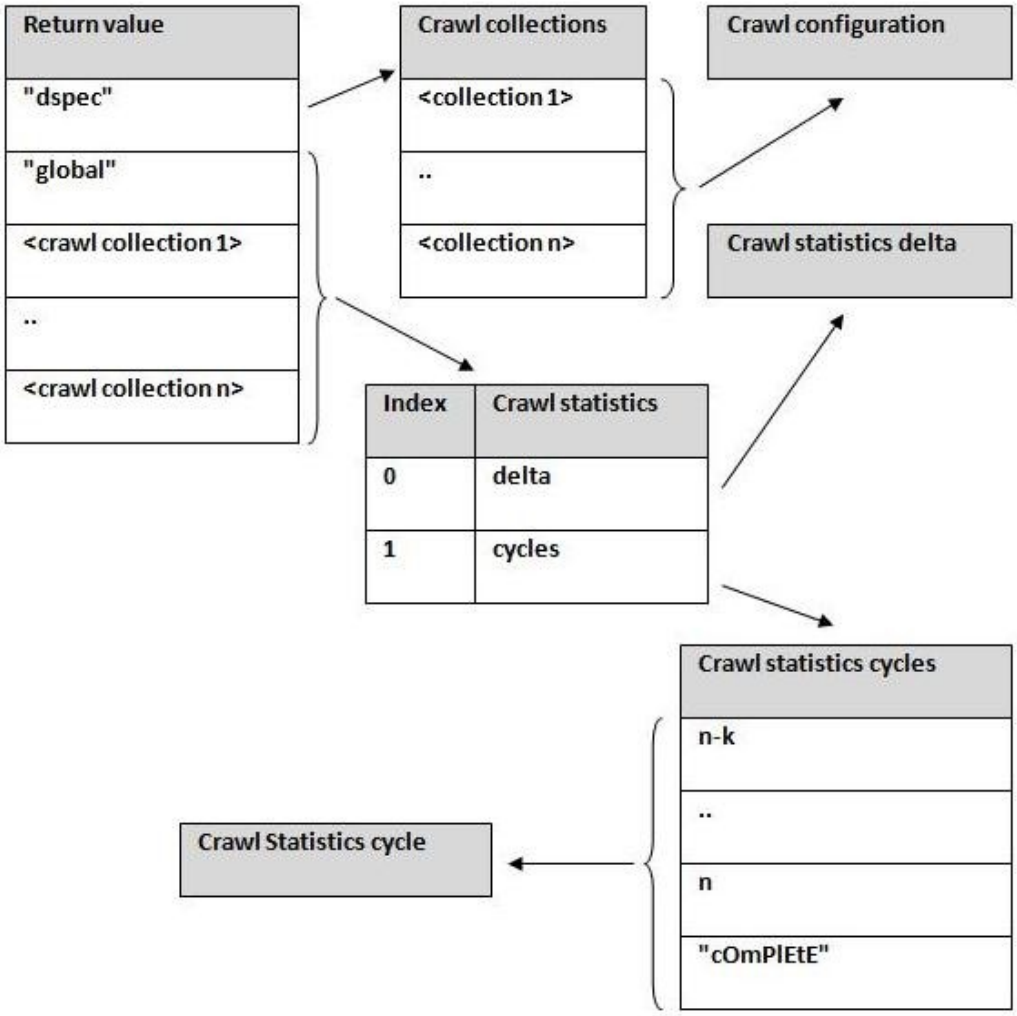

Figure 3: Crawl Statistics Structures

The **CollectionGetSiteStatistics** method specified in section 2.2.10.4 sends a similar structure where crawl collection MUST be substituted with the **crawl site** and the "global" key MUST NOT be present.

2.2.4.1 Statistics dictionary

The **Statistics dictionary** is a **dictionary** with keys of data type **string**, whose contents are specified in the following table.

Name	Value	Meaning
dspec	**dictionary**	A **dictionary** of crawl configuration options, see [MS-FSCCFG], that the protocol client uses to calculate progress information. See section 2.2.2.4.
global	**tuple**	Global statistics.

Name	Value	Meaning
<crawl collection name>	Tuple	Crawl collection statistics.

2.2.4.2 Crawl collections dictionary

The **Crawl collections dictionary** MUST contain one or more **string** keys that are equivalent to the names of crawl collections that are associated with the Web crawler. Each key maps to a **dictionary** that specifies configuration options for that crawl collection.

This **dictionary** MUST contain keys of data type **int** that are converted to data type **string**, as specified in the following table.

Name	Meaning
"refresh"	Specifies the **crawl refresh cycle** length, in minutes, for the crawl collection.
"refresh_mode"	Specifies the refresh mode. MUST be a value from the following table. For more details about the refresh modes, see [MS-FSCCFG].

The table below specifies the valid values for the "refresh_mode" key in the above **dictionary**.

Value	Meaning
1	Append refresh mode.
2	Prepend Refresh mode.
3	Reserved.
4	Scratch refresh mode.
5-7	Reserved.
8	Soft refresh mode.
9-31	Reserved.
32	Adaptive refresh mode.

2.2.4.3 Statistics tuple

The **statistics tuple** consists of the **crawl statistics delta** and the **crawl statistics cycles**. The **crawl statistics delta** specifies any statistics updates that are not yet merged into the appropriate **statistics cycle** structures. The crawl statistics cycles is an array that contains statistics for the last k crawl refresh cycles, in addition to the cycle indicated by the key value of "cOmPlEtE", which tracks the entire crawl. The value of k MUST be an **int** value that is greater than 0. The cycle specified by the "cOmPlEtE" key MUST contain statistics aggregated for all cycles.

2.2.4.4 Statistics delta

The **crawl statistics delta** MUST be implemented as one of three schemas. It is a global schema, a crawl collection schema, or a crawl site schema. It MUST contain only statistics that are aggregated over the last 1-2 minutes and have not yet been merged into a crawl statistics cycle. See section

2.2.4.5, for more details. When a delta is merged into a cycle, all its values MUST be re-initialized to 0. The __startts__ and __stopts__ **attributes** are used to track the beginning and ending times of the statistics delta. See section 2.2.4.7 for more details.

2.2.4.5 Statistics cycles

The crawl statistics cycles are in a **dictionary** that contains the most recent statistics. A crawl statistics cycle corresponds to a crawl refresh cycle.

The keys in the **dictionary** are associated with the number of the crawl refresh cycle, which MUST be of type **int** and MUST begin at 0 and increase by 1 for each cycle. This means that the highest number will be the most recent crawl statistics cycle.

The key "cOmPlEtE" MUST be used to specify the crawl statistics cycle that covers all cycles. The complete crawl statistics cycle contains the sum of all aggregated statistics cycles, including the current cycle, even if the cycles are no longer present in the statistics **dictionary**.

The values in the **dictionary** MUST be of type global schema, crawl collection schema or crawl site schema.

2.2.4.6 Statistics data types

The statistics data types are a set of composite data types that encapsulate four basic data types **int**, **long**, **double** and **string**. Numeric values are dynamically created as data type **int** or **long** depending on the size of the value. The **double** data type is used to store large numeric values because it is natively supported as specified in [XML-RPC]. The protocol client MUST support all four data types. The actual data type that the protocol MUST use to transport these structures is based on the data marshal format that is specified in [MS-FSWCU].

2.2.4.6.1 Continuous

The **continuous** data type is the data type for a continuous variable, suitable for graphing. It MUST be encoded as an array of length four, as specified in the following table. Each element within the table is of data type **int**, **long**, or **double**.

Array Index	Meaning
0	The current value of the variable.
1	The minimum value of the variable.
2	The maximum value of the variable.
3	The number of times the variable has been updated.

2.2.4.6.2 Discrete / Multi-discrete

The **discrete** data type encapsulates values that are represented independently of earlier samples. The data type MUST be encoded as an array of length 4, as specified in section 2.2.4.6.1.

The **multidiscrete** data type represents a value that MUST be a sum of individual **discrete** values. These values are not contained in the data type. For this protocol, the **multidiscrete** data type MUST be identical to the **discrete** data type.

2.2.4.6.3 Histogram

The **histogram** data type represents a set of values for a frequency distribution bar graph. The protocol MUST associate each numeric value with its number of occurrences. This data type MUST be encoded as an array of length 4, as specified by the following table.

Array Index	Data Type	Meaning
0	**dictionary**	**Dictionary** representing the histogram. In the key-value pairs, the keys specify the data values, and the values specify the occurrences of each data value. **Dictionary** keys are of type **string**. **Dictionary** values are of type **int**, **long** or **double**.
1	**string**	MUST contain the value "None".
2	**string**	MUST contain the value "None".
3	**int**, **long** or **double**	The number of times the variable has been updated.

2.2.4.6.4 Multi-dimension

The **multidimension** data type specifies a two-dimensional **array** that contains attributes that are associated with a set of values. This data type MUST be encoded as an **array** of length four, as specified by the following table.

Value	Data Type	Meaning
0	**dictionary**	**Dictionary** that represents the two dimensional **array**. Keys MUST be of type **string**, and specify the first dimension. Values MUST be of type **array**, and specify the second dimension. Each **array** MUST contain a sequence of 0 or more values of data type **int**, **long** or **double**.
1	**string**	Reserved, MUST contain the value "None".
2	**string**	Reserved, MUST contain the value "None".
3	**int**, **long**, or **double**	The number of times the variable has been updated.

2.2.4.7 Schema

This specifies the base class schema for the statistics, as specified in the following table. It consists of a **dictionary** whose keys are specified as lower case **strings**, and whose values specify the schema type, metadata, and contents. The mandatory keys MUST be included in the **dictionary**.

Name	Mandatory	Data Type	Meaning
__schemaname__	Yes	**string**	Specifies the schema type, which MUST be one of "globalschema", "siteschema", or "collectionschema".
__startts__	No	**timestamp**	Specifies when statistics aggregation was started.
__stopts__	No	**timestamp**	Specifies when statistics aggregation was stopped.

Name	Mandatory	Data Type	Meaning
name	Yes	**string**	See the following sections.
discrete	No	**array**	Specifies an array that contains discrete statistics attributes.
continuous	No	**array**	Specifies an array that contains continuous values.
histogram	No	**array**	Specifies an array that contains histogram values.
multidimension	No	**array**	Specifies an array that contains **multidimension** values.
multidiscrete	No	**array**	See discrete.

The contents of the **discrete**, **continuous**, **histogram**, **multidimension**, and **multidiscrete** keys MUST be an array of two elements as specified in the following table.

Value	Type	Meaning
0	string	The name of the statistics attribute. This value MUST be a valid name based on the current schema and attribute type.
1	array	The statistics value, which MUST be specified according to the definition of the attribute type. The attribute type MUST be one of **discrete**, **continuous**, **histogram**, **multidimension** or **multidiscrete**.

2.2.4.8 Global Statistics Schema

The global statistics schema contains statistics that is not specific to any specific crawl collection, and MUST be formatted as specified in section 2.2.4.7. The __schemaname__ MUST contain the value "globalschema". The attributes and their contents are specified in the following table.

Attribute	Aggregation Type	Data Type	Meaning
Uptime	Discrete	**timestamp**	The time that the protocol server began recording statistics which SHOULD be the time when the Web crawler began running.
DNSWrite	Continuous	**int**, **long** or **double**	Specifies, in bytes, the amount of **Domain Name System (DNS)** traffic that the protocol server has sent to a DNS server.
DNSRead	Continuous	**int**, **long** or **double**	Specifies, in bytes, the amount of DNS traffic from a DNS server.
DNSRequests	Continuous	**int**, **long** or **double**	Specifies the number of DNS requests that the protocol server has sent to a DNS server.
DNSResponses	Continuous	**int**, **long** or **double**	Specifies the number of DNS responses that the protocol server has received from a DNS server.
DNSRetries	Continuous	**int**, **long** or **double**	Specifies the number of DNS requests that the protocol server has sent to a DNS server as retries.
DNSTimeout	Continuous	**int**, **long** or	Specifies the number of DNS requests that resulted

Attribute	Aggregation Type	Data Type	Meaning
		double	in a timeout.
DNSResponse	Histogram	**string**	A **histogram** of DNS responses from a DNS server. These are specified as a number expressed as a **string**, and MUST be one of the DNS RCODE values that are specified in [RFC1035].
DNSRateLimit	Discrete	**int**, **long** or **double**	Specifies the maximum number of DNS requests that the protocol server has sent, in requests per second.

2.2.4.9 The Crawl Site Statistics Schema

The crawl site statistics are differentiated at a per-crawl-site level, and they MUST be formatted as specified in section 2.2.4.7. The __schemaname__ variable MUST contain the value "siteschema". The attributes and their contents are of data type **int**, **long**, or **double**, and MUST correspond to the values in the following table.

Attribute	Aggregation Type	Meaning
Processed	Continuous	Number of URIs extracted from the crawl queue and requested from the Web server during the crawl process.
Downloaded	Continuous	Number of HTTP responses that the Web server sent during crawling.
Stored	Continuous	Number of web documents, both modified and new, that were written to the web document storage during crawling.
Modified	Continuous	Number of modified web documents that were encountered during crawling. The Web crawler MUST track each URI that was downloaded and determine whether subsequent web documents from the same URI are the same identical web document or a modified version.
Deleted	Continuous	Number of web documents that were deleted from the Web crawler web document storage during crawling.
Unchanged	Continuous	Number of unchanged web documents that were encountered during crawling. A web document is unchanged if its computed **checksum** is the same as the value computed when the same URI was crawled in a previous crawl refresh cycle.
DBUpdates	Continuous	Number of URI metadata write processes.
ReadNet	Continuous	Number of bytes received on HTTP, **HTTPS** and **File Transfer Protocol (FTP)** network connections. This value MUST include information only from network connections that request URIs during crawling. Crawler inter process communication MUST NOT be included.
WriteNet	Continuous	Number of bytes sent on HTTP, HTTPS and FTP network connections. Crawler interprocess communication MUST NOT be included in this number.
HasLastMod	Continuous	Number of web documents that were downloaded during crawling that contain a "Last-Modified" HTTP header, as specified in

Attribute	Aggregation Type	Meaning
		[RFC2616].
HasETag	Continuous	Number of web documents that were downloaded during crawling that contain an "ETag" HTTP header, as specified in [RFC2616].
DLTime	Continuous	Total accumulated download time, in seconds, for all web documents that were crawled.
DocSize	Continuous	Total size, in bytes, of all web documents that were crawled.
Retries	Continuous	Number of URIs that failed crawling and resulted or will result in a retry to download the same URI.
HasSitemap	Continuous	Number of robots.txt **files** that were downloaded and that contain one or more "Sitemap:" directives that represent the URI of sitemaps, as specified in [SITEMAPS].
NodeSchedulerFwd	Continuous	Number of URIs that were forwarded between crawler nodes in a multinode installation.
LocalFwd	Continuous	Number of URIs that are **hyperlinks** to other crawl sites and were not forwarded to a different crawler node.
Local	Continuous	Number of URIs that are local hyperlinks within a crawl site.
LastUri	Discrete	The most recent URI crawled.
QueueLength	Discrete	The number of URIs on the Web crawler's URI queue.
CrawlStarted	Discrete	A **timestamp** that specifies the time when the crawl site began crawling. This value MUST be updated whenever the Web crawler begins a new crawl refresh cycle on the crawl site. The value MUST be updated whenever the Web crawler begins crawling the crawl site after a Web crawler restart.
CrawlEnded	Discrete	Specifies the time when the crawl site stopped crawling.
Epoch	Discrete	The crawl refresh cycle number. This value MUST be increased by 1 whenever the Web crawler begins crawling a crawl site.
LastRefresh	Discrete	A **timestamp** that specifies the time when the crawl site most recently began crawling.
HTTPResponse	Histogram	Specifies a **histogram** of HTTP response codes that were sent to the Web crawler, as specified by [RFC2616]. HTTP response codes MUST be specified as a number expressed as a string.
URISkip	Histogram	Specifies a histogram of URI skip codes. A skipped URI is one that was extracted from a web document, but not added to the crawl queue or crawled for some specific reason. **URI** skip codes MUST be one of the values specified in section 2.2.3.1.
DocSkip	Histogram	Specifies a **histogram** of web document skip codes. A skipped web document was downloaded from the remote Web server, but was not stored or sent to the indexing engine. Web document skip codes are specified in section 2.2.3.2.
MimeType	Histogram	Specifies a **histogram** of web document content types that were encountered in HTTP Content-Type headers during crawling. MUST

Attribute	Aggregation Type	Meaning
		be as specified in [RFC2616].

2.2.4.10 Crawl Collection Statistics Schema

The crawl collection statistics schema MUST specify crawl collection specific statistics level and MUST use the definition specified in section 2.2.4.7. The __schemaname__ MUST be specified as the **string** "collectionschema".

The attributes and their contents are specified in section 2.2.4.9 and in the following table. Section 2.2.4.9 applies to the entire crawl collection, rather than any specific crawl site.

Attribute	Aggregation Type	Data Type	Meaning
Uptime	Discrete	**timestamp**	Specifies the **timestamp** when the crawler began recording statistics for this crawl collection. This MUST correspond to the time when the crawl collection was added to the Web crawler.
ActiveSites	Discrete	**int**, **long** or **double**	Specifies the number of crawl sites currently being crawled. This MUST be 0 if the Web crawler is idle.
Feeding	Discrete	**int**	Specifies whether the Web crawler is currently submitting Web **documents** to the indexing engine. This MUST be 0 if content submission is paused, 1 otherwise.
Status	Discrete	**int**, **long** or **double**	Specifies the crawl collection status of the Web crawler. The value MUST be as specified in section 2.2.3.3.
CrawlMode	Discrete	**string**	If a **crawl limit** has been reached, the value of this string MUST be specified according to section 2.2.3.6.
StartUriStatus	Multidimension	**string**	Specifies the status of URIs being added on the crawl queue from **start URI** files, as specified by [MS-FSCCFG]. If start URIs are not currently being added to the crawl queue, then the structure MUST be empty. Otherwise, the structure MUST associate the start URI files to the status **strings**. The status **strings** are of the format "<filesize> <fileposition>", where <filesize> MUST specify the total file size in bytes and <fileposition> MUST specify the current file pointer position in bytes.
FeedingDestinations	Multidimension	**int**, **long** or **double**	Specifies the content collections that are associated with this crawl collection, and also specifies whether the process that submits site content to the indexing engine is currently active or paused. Each **dictionary** key MUST be associated with a

Attribute	Aggregation Type	Data Type	Meaning
			content destination in the crawl configuration as specified in [MS-FSCCFG] section 2.2.4.9. If content submission is paused, the associated **dictionary** value MUST have a **string** value of "0"; otherwise, the **string** MUST have a value of "1". The structure MUST be empty if there are no content destinations configured for the crawl collection that are not the default destination.
SumActiveSites	Multidiscrete	**int**, **long** or **double**	Multinode scheduler behavior: The attribute MUST specify the sum of the ActiveSites attribute across all node schedulers. Single node scheduler behavior: The attribute MUST be 0.
PPFailed	Continuous	**int**, **long** or **double**	Specifies the number of URIs that failed submission to the indexing system.
PPSucceeded	Continuous	**int**, **long** or **double**	The number of URIs that were successfully indexed by the indexing system.
PPFed	Continuous	**int**, **long** or **double**	The number of URIs that were successfully submitted to the indexing system.
PPDeleted	Continuous	**int**, **long** or **double**	The number of URIs that were successfully deleted from the indexing system
PPAdded	Continuous	**int**, **long** or **double**	The number of URIs that were processed by the Web crawler and that were previously unseen and have unique web document checksums.
PPModified	Continuous	**int**, **long** or **double**	The number of URIs that were processed by the Web crawler and that were previously seen and have unique web document checksums.
PPURLsChange	Continuous	**int**, **long** or **double**	The number of **urlschange_operations**, as specified in [MS-FSCF] section 2.2.42, that were sent to the indexing engine.
PPPartialUpdate	Continuous	**int**, **long** or **double**	Reserved, MUST be set to 0 or 0.0.
PPChecksums	Discrete	**int**, **long** or **double**	The number of unique web document checksums that exist in the Web crawler store.
SumPPChecksums	Multidiscrete	**int**, **long** or **double**	Multinode scheduler behavior: The attribute MUST specify the sum of the PPChecksums attribute across all node schedulers. Node scheduler behavior: The attribute MUST be 0.

2.2.5 Flattened Crawl collection Statistics Structure

This structure is used to encapsulate crawl collection statistics in a dictionary, which MUST correspond to the specification in the following table.

Attribute	Data Type	Meaning
ActiveSites	**int**	See section 2.2.4.10.
Processed	**double**	See section 2.2.4.9.
Downloaded	**double**	See section 2.2.4.9.
Modified	**double**	See section 2.2.4.9.
Stored	**double**	See section 2.2.4.9.
Deleted	**double**	See section 2.2.4.9.
ReadNet	**double**	See section 2.2.4.9.
WriteNet	**double**	See section 2.2.4.9.
DocSize	**double**	Total aggregated web document size.
DocSizeAvg	**double**	Average web document size in bytes.
DocSizeMax	**double**	Maximum web document size in bytes.
Retries	**double**	See section 2.2.4.9.
NodeSchedulerFwd	**double**	See section 2.2.4.9.
LocalFwd	**double**	See section 2.2.4.9.
Local	**double**	See section 2.2.4.9.
Epoch	**int**	See section 2.2.4.9.
LastRefresh	**int**	See section 2.2.4.9.
Feeding	**int**	See section 2.2.4.10.
CrawlMode	**string**	See section 2.2.4.10.
MimeType	**dictionary**	A **histogram** of web document content types that were encountered during crawling, as specified in [RFC2616]. Dictionary keys MUST be the MIME types, and their corresponding values MUST be the frequency.
HTTPResponse	**dictionary**	Specifies a **histogram** of HTTP response codes that were sent to the Web crawler, as specified by [RFC2696]. HTTP response codes MUST be a **string** literal that contains the HTTP response code number. Dictionary keys MUST be the HTTP response codes, and their corresponding values MUST be the frequency.
Uptime	**double**	See section 2.2.4.10.
DLTime	**double**	See section 2.2.4.9.
DLTimeAvg	**double**	Average time, in seconds, that is spent downloading a web document.
DLTimeMax	**double**	Maximum time, in seconds, that is spent downloading a web document.
Status	**string**	Crawling status. MUST be one of the following strings:

Attribute	Data Type	Meaning
		"Crawling", "Zombie", "Suspended", "Disk full", "Compacting", and "Variable delay suspended".
URISkip	**dictionary**	See section 2.2.4.9.
DocSkip	**dictionary**	See section 2.2.4.9.
DocumentStore	**int**	Estimated number of web documents with unique URIs that reside in the Web crawler store. MUST be calculated using the following formula: DocumentStore = Stored - Modified - Deleted
Progress	**array**	crawl refresh cycle progress. MUST be an array of two elements. The first element specifies the completion percentage of the current crawl refresh cycle. The second element is a human readable **string** that contains the amount of time until the next refresh occurs.
FirstUpdate	**double**	A **timestamp** that specifies when the crawl collection started crawling. This value MUST be updated whenever the Web crawler begins crawling the crawl site. The value MUST be updated when the Web crawler begins a new crawl after a crawler restart.
StatUpdate	**double**	Specifies the time when the crawl collection ended the crawl.
DocRate	**double**	Average web document download rate specified as web documents per second.
DataRateIn	**double**	Average incoming data rate from protocol headers and web document data for the sites that are being crawled. MUST be specified as bytes per second.
DataRateOut	**double**	Average outgoing data rate for information such as HTTP request messages. MUST be specified as bytes per second.
PPFailed	**double**	See section 2.2.4.10.
PPSucceeded	**double**	See section 2.2.4.10.
PPFed	**double**	See section 2.2.4.10.
PPDeleted	**double**	See section 2.2.4.10.
PPAdded	**double**	See section 2.2.4.10.
PPModified	**double**	See section 2.2.4.10.
PPURLsChange	**double**	See section 2.2.4.10.
PPChecksums	**double**	See section 2.2.4.10.

2.2.6 Flattened Global Statistics Structure

This structure is used to encapsulate global crawl statistics in a **dictionary**, which MUST correspond to the specification in the following table.

Attribute	Data Type	Meaning
DNSWrite	**double**	See section 2.2.4.8.
DNSRead	**double**	See section 2.2.4.8.
DNSRequest	**double**	See section 2.2.4.8.
DNSResponse	**double**	See section 2.2.4.8.
DNSResponses	**dictionary**	See section 2.2.4.8.
DNSRetries	**double**	See section 2.2.4.8.
DNSTimeout	**double**	See section 2.2.4.8.
DNSRateLimit	**double**	See section 2.2.4.8.
DNSRate	**double**	Number of sent DNS requests per second.
DNSDataRateIn	**double**	Number of DNS traffic bytes received per second.
DNSDataRateOut	**double**	Number of DNS traffic bytes sent per second.

2.2.7 Error Handling

The protocol supports a special fault response message for reporting errors to the protocol client, as specified in [XML-RPC]. The message consists of a fault code whose value MUST be 1 and a fault string that specifies the error that occurred.

Most errors that occur in the methods of this specification generate faults when an error occurs. When a fault is generated, it replaces the return value of the method.

2.2.8 Web crawler Management

All methods specified in this section MUST have all arguments specified.

2.2.8.1 GetLogLevel

The **GetLogLevel** method queries the protocol server for its current log verbosity level.

```
int GetLogLevel()
```

Return value: MUST return the log level, which MUST be a mask of one or more of the log levels specified in section 2.2.3.5.

2.2.8.2 GetNodeSchedulerXmlRpcPorts

The **GetNodeSchedulerXmlRpcPort** method queries the protocol server for a list of known node schedulers.

```
array GetNodeSchedulerXmlRpcPorts()
```

Return value: If the protocol server is a multinode scheduler, then the return value MUST specify an array with at least 1 element. The array elements MUST be arrays that are two elements long,

where the first element specifies a **hostname** as a **string** and the second specifies a **port** as an **int**. The combination of a **hostname** and **port** MUST uniquely identify a node scheduler that is registered with the multinode scheduler.

If the protocol server is not a multinode scheduler, the protocol server MUST send a fault as specified in [XML-RPC].

2.2.8.3 GetSiteManagerNum

The **GetSiteManagerNum** method queries the protocol server for the number of site managers that the crawler node controls. A site manager is a service that runs on the crawler node and manages crawling for a set of crawl sites.

```
int GetSiteManagerNum()
```

Return value: MUST return the number of site managers that the crawler node controls. If the protocol server is a crawler multinode scheduler, the number of site managers MUST be 0.

2.2.8.4 IsDistributed

The **IsDistributed** method queries the protocol server for the multinode Web crawler setup state.

```
int IsDistributed()
```

Return value: MUST return a value of 1 if the protocol server is part of a multinode Web crawler setup, regardless of the role of the protocol server. Otherwise, it MUST return a value of 0.

2.2.8.5 IsMultiNodeScheduler

The **IsMultiNodeScheduler** method queries the protocol server whether it is a multinode scheduler or a node scheduler.

```
int IsMultiNodeScheduler()
```

Return value: MUST return 1 if the protocol server represents a multinode scheduler in a multinode Web crawler setup. If the protocol server is a node scheduler the value returned MUST be 0.

2.2.8.6 SetLogLevel

The **SetLogLevel** method sets the log verbosity level for the protocol server. The protocol server MUST set its log verbosity to the level specified as a result of this method. If the protocol server has the role of a multinode scheduler, this method MUST also be forwarded to each individual node scheduler before the method returns the response.

```
int SetLogLevel(int Level)
```

Level: The log level. See section 2.2.3.5 for more details about log levels.

Return value: MUST return 1 if the method succeeds, 0 otherwise.

2.2.9 Crawl Collection Management

All of the methods that are specified in this section MUST specify all arguments.

2.2.9.1 AddURIs

The **AddURIs** method adds one or more URIs to the crawl queue.

```
cresult AddURIs(string Collection, int Urgent, array URIs)
```

Collection: The name of crawl collection.

Urgent: MUST be either 0 or 1. When set to 1 the URIs MUST be processed as soon as possible by the Web crawler.

URIs: An array of **strings** specifying URIs that are scheduled for crawling. The URIs MUST be **absolute URI**s.

Return value: The result of the method.

2.2.9.2 AddURIFile

The **AddURIFile** method adds one or more URIs contained in a text file to the crawl queue.

```
cresult AddURIFile(string Collection, int Urgent, string URIFile)
```

Collection: The name of crawl collection. If the crawl collection does not exist then a fault MUST be sent, as specified in [XML-RPC].

Urgent: MUST be either 0 or 1. When set to 1 the URIs MUST be processed as soon as possible by the Web crawler.

URIFile: The absolute path of a file on the local file system on the protocol server. The file MUST be in the UTF-8 encoding. Each individual URI MUST be specified on a single line. The URIs MUST be absolute.

Return value: The result of the method.

2.2.9.3 CollectionAdd

The **CollectionAdd** method adds a new crawl collection to the Web crawler. If the crawl collection already exists the existing configuration MUST be updated instead. The crawler MUST use the configuration specified in the Crawler Configuration File Format Specification [MS-FSCCFG].

A partial configuration can be specified. If a partial configuration is submitted and the crawl collection already exists, this method merges the partial configuration with the existing configuration, a process where all settings that are specified in both configurations MUST be overridden by the new configuration. If the crawl collection does not already exist it MUST be merged with the default configuration values as specified in [MS-FSCCFG].

```
cresult CollectionAdd(string ConfigData, int Force)
```

ConfigData: A crawl configuration in XML format.

force: MUST be either 0 or 1. If the value is 1 then the configuration MUST be accepted even if the protocol server is acting as a node scheduler in a multinode setup, otherwise a fault MUST be returned. A multinode scheduler or node scheduler that performs as a single node MUST ignore this option and always add the crawl collection.

Return value: The result of the method.

2.2.9.4 CollectionDelete

The **CollectionDelete** method deletes an existing crawl collection from the Web crawler.

```
cresult CollectionDelete(string Collection, int Force)
```

Collection: The name of the crawl collection to be deleted.

Force: MUST be either 0 or 1. If the value is 1 then the configuration MUST be deleted even if the protocol server is acting as a node scheduler in a multinode setup, otherwise a fault MUST be returned. A multinode scheduler or node scheduler that performs as a single node MUST ignore this option and always delete the crawl collection.

Return value: The result of the method.

2.2.9.5 CollectionDeleteSite

The **CollectionDeleteSite** method deletes a crawl site from the specified crawl collection in the Web crawler store and from the searchable index.

```
cresult CollectionDeleteSite(string Collection, hostport Site)
```

Collection: The name of the crawl collection.

Site: The crawl site to be deleted.

Return value: The result of the method.

2.2.9.6 CollectionDeleteURIs

The **CollectionDeleteURIs** method deletes one or more URIs from the specified crawl collection in the Web crawler store and from the searchable index.

```
cresult CollectionDeleteURIs(string Collection, array URIs)
```

Collection: The name of the crawl collection.

URIs: An array of **strings** that specifies one or more URIs to be deleted. The URIs specified MUST be absolute.

Return value: The result of the method.

2.2.9.7 CollectionDeleteURIFile

The **CollectionDeleteURIFile** method deletes one or more URIs specified in a file from the specified crawl collection in the Web crawler store and from the searchable index.

```
cresult CollectionDeleteURIFile(string Collection, string URIFile)
```

Collection: The name of the crawl collection.

URIFile: The absolute path of a file on the local file system on the protocol server. The file MUST be in the UTF-8 encoding. Each individual URI to be deleted MUST be specified on a single line. The URIs MUST be absolute.

Return value: The result of the method.

2.2.9.8 CollectionGetConfigurationXML

The **CollectionGetConfigurationXML** method returns the specified crawl configuration as XML.

```
string CollectionGetConfigurationXML(string Collection)
```

Collection: The name of the crawl collection.

Return value: The crawl configuration in XML, as specified in [MS-FSCCFG].

2.2.9.9 CollectionGetList

The **CollectionGetList** method returns the list of crawl collections currently configured in the Web crawler.

```
array CollectionGetList()
```

Return value: An array of **strings**, each of which specifies the name of a crawl collection.

2.2.9.10 CollectionGetSiteRoutingAddress

The **CollectionGetSiteRoutingAddress** method returns information about which node scheduler is responsible for the specified crawl site in a multinode Web crawler setup.

```
cresult CollectionGetSiteRoutingAddress(string Collection, hostport Site)
```

Collection: The name of the crawl collection.

Site: The crawl site for which to retrieve routing information.

Return value: On success the second element in the **cresult** is an array of two elements that specify the node scheduler to which to route the crawl site. The first element is the hostname and the second element is the port of the node scheduler on that node.

2.2.9.11 CollectionPreemptSite

The **CollectionPreemptSite** method stops the crawling that is being performed on the specified crawl site. The protocol server SHOULD stop crawling immediately, but MUST resume the crawling at a later time.

```
cresult CollectionPreemptSite(string Collection, hostport Site)
```

Collection: The name of the crawl collection.

Site: The crawl site to be preempted.

Return value: The result of the method.

2.2.9.12 CollectionQuarantineSite

The **CollectionQuarantineSite** method suspends crawling of the specified crawl site for the specified time interval. Any URIs that belong to the quarantined site found during quarantine MUST be added on a separate queue for the duration of the quarantine.

```
cresult CollectionQuarantineSite(string Collection, hostport Site, int Time)
```

Collection: The name of the crawl collection.

Site: The crawl site to be preempted.

Time: The duration of the quarantine in seconds.

Return value: The result of the method.

2.2.9.13 CollectionRefetch

The **CollectionRefetch** method schedules a **crawl refetch** of the specified crawl collection.

```
cresult CollectionRefetch(string Collection, string SubCollection)
```

Collection: The name of the crawl collection.

SubCollection: The name of a **crawl subcollection**, as specified in [MS-FSCCFG]. This option MUST be an empty **string**, in which case the entire crawl collection MUST be re-crawled, or the refetch process MUST only apply to the specified crawl subcollection.

Return value: The result of the method.

2.2.9.14 CollectionRefetchURI

The **CollectionRefetchURI** method schedules a crawl refetch process for the specified URI or crawl site.

```
cresult CollectionRefetchURI(string Collection, string URI, int Refeed, int Site, int
Immediately)
```

Collection: The name of the crawl collection.

URI: The URI to retrieved. MUST specify an absolute URI.

Refeed: MUST be either 0 or 1. When the value is 1, the URI MUST also be sent to the processing pipeline, regardless of whether it was modified after the last time it was retrieved.

Site: MUST be either 0 or 1. When the value is 1 the method MUST re-crawl the entire crawl site associated with the URI, not just the URI itself.

Immediately: MUST be either 0 or 1. When the value is 1 the URI MUST be treated as an urgent URI, meaning the protocol server SHOULD crawl the URI as soon as possible.

Return value: The result of the method.

2.2.9.15 CollectionReprocessSite

The **CollectionReprocessSite** method sends all known web documents from the specified crawl site to the indexing engine for indexing.

```
cresult CollectionReprocessSite(string Collection, contentdest Destination, hostport Site)
```

Collection: The name of the crawl collection.

Destination: The content submission destination.

Site: The crawl site to be processed.

Return value: The result of the method.

2.2.9.16 CollectionReprocessSitePrefix

The **CollectionReprocessSitePrefix** method sends all known web documents from the specified crawl site matching the URI prefix to the indexing engine for indexing.

```
cresult CollectionReprocessSitePrefix(string Collection, hostport Site, string Prefix,
contentdest Destination)
```

Collection: The name of the crawl collection.

Site: The crawl site to be processed.

Prefix: A URI prefix which MUST be used to perform a prefix match on each URI considered a candidate for processing.

Destination: The content submission destination.

Return value: The result of the method.

2.2.9.17 CollectionReprocessURI

The **CollectionReprocessURI** method sends all known web documents from the specified crawl site to the indexing engine for indexing.

```
cresult CollectionReprocessURI(string Collection, contentdest Destination, array URIs)
```

Collection: The name of the crawl collection.

Destination: The content submission destination.

URIs: An array of strings that specifies one or more URIs to be processed. The URIs specified MUST be absolute URIs. The protocol server SHOULD continue processing URIs even if non-valid URIs are encountered.

Return value: The result of the method.

2.2.9.18 CollectionResume

The **CollectionResume** method resumes crawling of a crawl collection that was previously suspended, for example by using a call to the **CollectionSuspend** method as specified in section 2.2.9.20. If the collection is already suspended, the protocol server MUST send a fault as specified in [XML-RPC].

```
cresult CollectionResume(string Collection)
```

collection: The name of the crawl collection.

Return value: The result of the method.

2.2.9.19 CollectionResumeFeeding

The **CollectionResumeFeeding** method resumes submission of crawled content to the indexing engine for a crawl collection that was previously suspended, for example by using a call to the **CollectionSuspendFeeding** method as specified in section 2.2.9.21.

```
cresult CollectionResumeFeeding(string Collection, contentdest Destination)
```

Collection: The name of the crawl collection.

Destination: The content submission destination.

Return value: The result of the method.

2.2.9.20 CollectionSuspend

The **CollectionSuspend** method suspends crawling of a crawl collection. Crawling of the crawl collection MUST NOT already be suspended.

```
cresult CollectionSuspend(string Collection)
```

Collection: The name of the crawl collection.

Return value: The result of the method.

2.2.9.21 CollectionSuspendFeeding

The **CollectionSuspendFeeding** method suspends content submissions of crawled content for indexing. The crawl collection MUST NOT already have web document submission set to suspended.

```
cresult CollectionSuspendFeeding(string Collection, contentdest Destination)
```

Collection: The name of the crawl collection.

Destination: The content submission destination.

Return value: The result of the method.

2.2.10 Crawl Collection Status

All methods specified in this section MUST have all arguments specified.

2.2.10.1 CollectionDisableRefreshOnly

The **CollectionDisableRefreshOnly** method changes from crawling only previously visited content to also crawling new content. The protocol server MUST now download new content as well as revisiting old content.

```
cresult CollectionDisableRefreshOnly(string Collection)
```

Collection: The name of the crawl collection.

Return value: The result of the method.

2.2.10.2 CollectionEnableRefreshOnly

The **CollectionEnableRefreshOnly** method only revisits previously crawled content only. The Web crawler MUST only revisit previously crawled content, and MUST NOT download any URIs not already present in the Web crawler store.

```
cresult CollectionEnableRefreshOnly(string Collection)
```

Collection: The name of the crawl collection.

Return value: The result of the method.

2.2.10.3 CollectionGetStatus

The **CollectionGetStatus** method returns the status of the Web crawler.

```
string CollectionGetStatus(string Collection)
```

Collection: The name of the crawl collection.

Return value: The current state of the crawl collection, formatted as specified in section 2.2.3.3.

2.2.10.4 CollectionGetSiteStatistics

The **CollectionGetSiteStatistics** method MUST return the aggregated statistics for the specified crawl site.

```
struct CollectionGetSiteStatistics(string Collection, string Site)
```

Collection: MUST specify a valid crawl collection name.

Site: MUST specify a valid crawl site crawled by the crawl collection.

Return value: The method MUST return a valid statistics structure as specified in section 2.2.3.5. The statistics structure MUST contain statistics for the specified crawl site and the structure MUST be serialized using the pyfastmarshal format, as specified in [MS-FSWCU].

2.2.10.5 CollectionGetStatistics

The **CollectionGetStatistics** method MUST return the aggregated statistics for the specified crawl collection. The method MUST also return global statistics. If no crawl collections are specified the method MUST return statistics for all configured crawl collections.

```
struct CollectionGetStatistics(string Collection)
```

Collection: MUST specify a valid crawl collection name or an empty **string**.

Return value: The method MUST return a valid statistics structure as specified in section 2.2.4. The statistics structure MUST contain statistics for the specified crawl collection as well as global statistics. The structure MUST be marshalled using the pyfastmarshal format, as specified in [MS-FSWCU].

2.2.10.6 CollectionGetStatistics2

The **CollectionGetStatistics2** method MUST return the aggregated statistics for the specified crawl collection.

```
array CollectionGetStatistics2(string Collection)
```

Collection: MUST specify a valid crawl collection name.

Return value: The method MUST return an array of two elements. The array MUST be as specified in the following table.

Value	Meaning
0	Specifies The result of the method. Its value MUST be 1 or less than 1. If the **int** has a value of 1, the method was successful and the other element of the array contains a dictionary type. Otherwise, the method was unsuccessful and the other element of the array contains a **string** that specifies the error text.
1	Contains a **string** or a **dictionary** data type. If this method returns a **dictionary** data type, it MUST contain one, two, or three keys of type **string**. The valid values are "cur", "prev" and "complete", each of which refers to a **dictionary** as specified in section 2.2.4. The contents of the dictionaries MUST be one of the following: ▪ The "cur" **dictionary** data type refers to a **dictionary** that MUST contain statistics from the current crawl refresh cycle. ▪ The "prev" **dictionary** data type refers to a **dictionary** that MUST contain statistics from the previous crawl refresh cycle. If there is no previous cycle available this key MUST NOT be returned. ▪ The "complete" **dictionary** data type refers to a **dictionary** that MUST contain statistics for the entire lifetime of the crawl collection. The **dictionary** referred to by the key MUST be empty when no statistics are available.

2.2.10.7 GetGlobalStatistics

The **GetGlobalStatistics** method MUST return the aggregated global statistics for the Web crawler.

```
array GetGlobalStatistics()
```

Return value: The method MUST return an array of two elements. The array MUST be as specified in the following table.

Value	Meaning
0	Specifies The result of the method. Its value MUST be 1 or less than 1. If the **int** has a value of 1, the method was successful and the other element of the array contains a **dictionary** data type. Otherwise, the method was unsuccessful and the other element of the array contains a **string** that specifies the error text.
1	Contains a **string** or a **dictionary** data type. If this value contains a **dictionary** data type, it MUST contain one or two keys of type **string**. The valid values are "delta" and "complete", each of which refers to a **dictionary** as specified in section 2.2.4.8. The contents of the dictionaries MUST be one of the following: ▪ The "delta" **dictionary** type refers to a **dictionary** that MUST contain recently aggregated statistics over a short time window. ▪ The "complete" **dictionary** data type refers to a **dictionary** that MUST contain statistics for the entire lifetime of the Web crawler. The **dictionary** to which the key refers MUST be empty when no statistics are available.

3 Protocol Details

3.1 Common Details

None.

3.2 Client Details

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.

3.2.2 Timers

None.

3.2.3 Initialization

The protocol requires setup of a **Transmission Control Protocol (TCP)** connection between protocol client and protocol server. The port numbers used for the connection is **base port**. The protocol client MUST initiate the connection.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

None.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Server Details

In its initial state, the Web crawler MUST have no configured crawl collections. Methods specified in section 2 are used to alter the state of the Web crawler, by adding or removing crawl collections. The protocol server MUST persist these altered states.

3.3.1 Abstract Data Model

The protocol server MUST listen for incoming connections, process incoming XML-RPC requests and respond to them in a timely manner. If an unexpected error occurs during processing the protocol server MUST send a fault as specified in [XML-RPC].

The protocol server MUST persist the following information:

Crawl collection configurations: A data structure that contains information associated with crawl collections that are currently known to the Web crawler, in addition to their respective configuration settings. This data MUST be persisted to disk so the Web crawler can restart and resume crawling.

Statistics: Statistics MUST be updated during run time and MUST also be persisted to disk between restarts. The current and global crawl statistics cycles MUST always be retained. Statistics for recent crawl refresh cycles MUST also be retained, but the number of cycles to retain is implementation dependent.

Crawl queues: Individual crawl queues for each crawl collection MUST be persisted to disk. Quarantined URIs MUST be persisted to different queues, one or more per crawl collection.

Current activity: The protocol server MUST keep track of which crawl sites are being crawled at any time, as well as the state of crawls being performed on each crawl collection. The state of the crawl collection MUST also be persisted to disk between restarts.

Crawl store: The protocol server MUST maintain a persistent Web crawler store on disk. The minimum information that MUST be stored in the Web crawler store is the fields that specify which URIs have been crawled, the checksum of each URI, and when the URI was last crawled.

3.3.2 Timers

None.

3.3.3 Initialization

The protocol server MUST start its XML-RPC server implementation as soon as it can process incoming requests. The protocol server MUST listen on the base port. See section 3.3.3 for more details about protocol initialization.

The protocol server MUST register with the Configuration Service as specified in [MS-FSCX], and implement the following methods that are required by that protocol: **ConfigurationChanged**, **ReRegister** and **ping**.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

Specific processing rules apply to the following methods:

CollectionAdd: This method MUST update the crawl configuration state when the new crawl collection is added. An empty statistics structure MUST be associated with the crawl collection. The crawl collection state MUST be set to a value of "running".

CollectionDelete: This method MUST initiate deletion of all crawl collection specific data structures, including crawl configuration, statistics, crawl queues, Web crawler store and other internal structures. While the delete process is in progress, the status code of the crawl collection MUST be set to a value of "zombie" until the collection has been completely removed. The Web crawler MUST remove any remaining references associated with the crawl collection from its structures so the crawl collection can be added as a new crawl collection.

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

4 Protocol Examples

These examples only contain the XML body for each XML-RPC message. See [XML-RPC] for examples of HTTP headers.

4.1 CollectionGetList

This example shows a **CollectionGetList** method, which returns the crawl collections known to the Web crawler as an array of **strings**. In this example, the crawl collections are "contoso" and "web".

Request:

```
<?xml version='1.0'?>
<methodCall>
  <methodName>CollectionGetList</methodName>
  <params>
  </params>
</methodCall>
```

Response:

```
<?xml version='1.0'?>
<methodResponse>
  <params>
    <param>
      <value>
        <array>
          <data>
            <value><string>contoso</string></value>
            <value><string>web</string></value>
          </data>
        </array>
      </value>
    </param>
  </params>
</methodResponse>
```

4.2 AddURIs

In this example, the **AddURIs** method adds two URIs to the crawl queue for the crawl collection "contoso". The result value is in the **cresult** format, and consists of an array with two elements; those two elements are a return code and a description of the results.

Request:

```
<?xml version='1.0'?>
<methodCall>
  <methodName>AddURIs</methodName>
  <params>
    <param>
      <value><string>contoso</string></value>
    </param>
    <param>
      <value><int>0</int></value>
    </param>
```

```
    <param>
      <value>
        <array>
          <data>
            <value><string>http://www.contoso.com/</string></value>
            <value><string>http://www.contoso.com/about/</string></value>
          </data>
        </array>
      </value>
    </param>
  </params>
</methodCall>
```

Response:

```
<?xml version='1.0'?>
<methodResponse>
  <params>
    <param>
      <value>
        <array>
          <data>
            <value><int>1</int></value>
            <value><string>Queued collection contoso with 2 URIs</string></value>
          </data>
        </array>
      </value>
    </param>
  </params>
</methodResponse>
```

4.3 CollectionGetStatistics2

In this example, a result is returned from the **CollectionGetStatistics2** method. It demonstrates nested structures, and a variety of data types.

Request:

```
<?xml version='1.0'?>
<methodCall>
  <methodName>CollectionGetStatistics2</methodName>
  <params>
    <param>
      <value><string>contoso</string></value>
    </param>
  </params>
</methodCall>
```

Response:

```
<?xml version='1.0'?>
<methodResponse>
  <params>
    <param>
      <value><array><data>
```

```
<value><int>1</int></value>
<value><struct>
<member>
  <name>cur</name>
  <value><struct>
  <member>
    <name>PPAdded</name>
    <value><double>40.0</double></value>
  </member>
  <member>
    <name>ReadNet</name>
    <value><double>174367.0</double></value>
  </member>
  <member>
    <name>Feeding</name>
    <value><int>1</int></value>
  </member>
  <member>
    <name>DLTimeMax</name>
    <value><double>0.20227789878845215</double></value>
  </member>
  <member>
    <name>Retries</name>
    <value><double>0.0</double></value>
  </member>
  <member>
    <name>DocSizeAvg</name>
    <value><double>4071.0</double></value>
  </member>
  <member>
    <name>PPURLsChange</name>
    <value><double>0.0</double></value>
  </member>
  <member>
    <name>DocumentStore</name>
    <value><int>40</int></value>
  </member>
  <member>
    <name>URISkip</name>
    <value><struct>
    </struct></value>
  </member>
  <member>
  <name>Uptime</name>
    <value><double>1234197854.7414169</double></value>
  </member>
  <member>
    <name>Epoch</name>
    <value><int>0</int></value>
  </member>
  <member>
    <name>FirstUpdate</name>
    <value><double>1234195119.4230039</double></value>
  </member>
  <member>
    <name>Progress</name>
    <value><array><data>
    <value><int>2</int></value>
    <value><string>1 days 8h 34m 23s until scratch refresh</string></value>
```

```
  </data></array></value>
</member>
<member>
  <name>Local</name>
  <value><double>320.0</double></value>
</member>
<member>
  <name>Status</name>
  <value><string>Crawling</string></value>
</member>
<member>
  <name>PPSucceeded</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>PPDeleted</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>Deleted</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>LocalFwd</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>DataRateIn</name>
  <value><double>63.722122866030894</double></value>
</member>
<member>
  <name>PPModified</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>Downloaded</name>
  <value><double>40.0</double></value>
</member>
<member>
  <name>DLTimeAvg</name>
  <value><double>0.007553648948669433 4</double></value>
</member>
<member>
  <name>WriteNet</name>
  <value><double>12790.0</double></value>
</member>
<member>
  <name>HTTPResponse</name>
  <value><struct>
    <member>
      <name>200</name>
      <value><double>41.0</double></value>
    </member>
    <member>
      <name>404</name>
      <value><double>1.0</double></value>
    </member>
  </struct></value>
</member>
```

```
<member>
  <name>DLTime</name>
  <value><double>0.30214595794677734</double></value>
</member>
<member>
  <name>MimeType</name>
  <value><struct>
    <member>
      <name>text/html</name>
      <value><double>40.0</double></value>
    </member>
  </struct></value>
</member>
<member>
  <name>Modified</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>DocSkip</name>
  <value><struct>
  </struct></value>
</member>
<member>
  <name>ActiveSites</name>
  <value><int>0</int></value>
</member>
<member>
  <name>DocRate</name>
  <value><double>0.014617931802699111</double></value>
</member>
<member>
  <name>Processed</name>
  <value><double>40.0</double></value>
</member>
<member>
  <name>DocSizeMax</name>
  <value><double>8460.0</double></value>
</member>
<member>
  <name>StatUpdated</name>
  <value><string>26.2s</string></value>
</member>
<member>
  <name>NodeSchedulerFwd</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>PPChecksums</name>
  <value><double>40.0</double></value>
</member>
<member>
  <name>LastRefresh</name>
  <value><int>1234195119</int></value>
</member>
<member>
  <name>LastUpdate</name>
  <value><double>1234197855.7882121</double></value>
</member>
<member>
```

```
      <name>PPFed</name>
      <value><double>0.0</double></value>
    </member>
    <member>
      <name>DataRateOut</name>
      <value><double>4.6740836939130403</double></value>
    </member>
    <member>
      <name>Unchanged</name>
      <value><double>0.0</double></value>
    </member>
    <member>
      <name>DocSize</name>
      <value><double>162862.0</double></value>
    </member>
    <member>
      <name>Stored</name>
      <value><double>40.0</double></value>
    </member>
      <member>
      <name>CrawlMode</name>
      <value><string></string></value>
    </member>
    <member>
      <name>PPPartialUpdate</name>
      <value><double>0.0</double></value>
    </member>
    <member>
      <name>PPFailed</name>
      <value><double>0.0</double></value>
    </member>
  </struct></value>
  </member>
  <member>
    <name>complete</name>
    <value><struct>
    <member>
      <name>PPAdded</name>
      <value><double>40.0</double></value>
    </member>
    <member>
      <name>ReadNet</name>
      <value><double>174367.0</double></value>
    </member>
    <member>
      <name>Feeding</name>
      <value><int>1</int></value>
    </member>
    <member>
      <name>DLTimeMax</name>
      <value><double>0.20227789878845215</double></value>
    </member>
    <member>
      <name>Retries</name>
      <value><double>0.0</double></value>
    </member>
    <member>
      <name>DocSizeAvg</name>
      <value><double>4071.0</double></value>
```

```
</member>
<member>
  <name>PPURLsChange</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>DocumentStore</name>
  <value><int>40</int></value>
</member>
<member>
  <name>URISkip</name>
  <value><struct>
  </struct></value>
</member>
<member>
  <name>Uptime</name>
  <value><double>1234197854.7414169</double></value>
</member>
<member>
  <name>DNSResponse</name>
  <value><struct>
  </struct></value>
</member>
<member>
  <name>Epoch</name>
  <value><int>0</int></value>
</member>
<member>
  <name>FirstUpdate</name>
  <value><double>1234195119.4230039</double></value>
</member>
<member>
  <name>Progress</name>
  <value><array><data>
  <value><int>2</int></value>
  <value><string>1 days 8h 34m 23s until scratch refresh</string></value>
  </data></array></value>
</member>
<member>
  <name>Local</name>
  <value><double>320.0</double></value>
</member>
<member>
  <name>DNSWrite</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>Status</name>
  <value><string>Crawling</string></value>
</member>
<member>
  <name>PPSucceeded</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>PPDeleted</name>
  <value><double>0.0</double></value>
</member>
<member>
```

```
  <name>Deleted</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>LocalFwd</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>DataRateIn</name>
  <value><double>63.722122866030894</double></value>
</member>
<member>
  <name>PPModified</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>Downloaded</name>
  <value><double>40.0</double></value>
</member>
<member>
  <name>Stored</name>
  <value><double>40.0</double></value>
</member>
<member>
  <name>DLTimeAvg</name>
  <value><double>0.007553648948669433 4</double></value>
</member>
<member>
  <name>WriteNet</name>
  <value><double>12790.0</double></value>
</member>
<member>
  <name>HTTPResponse</name>
  <value><struct>
  <member>
    <name>200</name>
    <value><double>41.0</double></value>
  </member>
  <member>
    <name>404</name>
    <value><double>1.0</double></value>
  </member>
  </struct></value>
</member>
<member>
  <name>DLTime</name>
  <value><double>0.30214595794677734</double></value>
</member>
<member>
  <name>DNSTimeout</name>
  <value><double>0.0</double></value>
</member>
  <member>
  <name>MimeType</name>
  <value><struct>
  <member>
    <name>text/html</name>
    <value><double>40.0</double></value>
  </member>
```

```
  </struct></value>
</member>
<member>
  <name>LastUpdate</name>
  <value><double>1234197855.7882121</double></value>
</member>
<member>
  <name>Modified</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>DocSkip</name>
  <value><struct>
  </struct></value>
</member>
<member>
  <name>ActiveSites</name>
  <value><int>0</int></value>
</member>
<member>
 <name>DocRate</name>
 <value><double>0.014617931802699111</double></value>
</member>
<member>
  <name>Processed</name>
  <value><double>40.0</double></value>
</member>
<member>
  <name>DocSizeMax</name>
  <value><double>8460.0</double></value>
</member>
<member>
  <name>StatUpdated</name>
  <value><string>26.2s</string></value>
</member>
<member>
  <name>NodeSchedulerFwd</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>PPChecksums</name>
  <value><double>40.0</double></value>
</member>
<member>
  <name>LastRefresh</name>
  <value><int>1234195119</int></value>
</member>
<member>
  <name>DNSRateLimit</name>
  <value><double>10.0</double></value>
</member>
<member>
  <name>PPFed</name>
  <value><double>0.0</double></value>
</member>
<member>
  <name>DataRateOut</name>
  <value><double>4.6740836939130403</double></value>
</member>
```

```
      <member>
        <name>DNSRequests</name>
        <value><double>0.0</double></value>
      </member>
      <member>
        <name>Unchanged</name>
        <value><double>0.0</double></value>
      </member>
      <member>
        <name>DNSRetries</name>
        <value><double>0.0</double></value>
      </member>
      <member>
        <name>DocSize</name>
        <value><double>162862.0</double></value>
      </member>
      <member>
        <name>DNSResponses</name>
        <value><double>0.0</double></value>
      </member>
      <member>
        <name>CrawlMode</name>
        <value><string></string></value>
      </member>
      <member>
        <name>DNSRead</name>
        <value><double>0.0</double></value>
      </member>
      <member>
        <name>PPPartialUpdate</name>
        <value><double>0.0</double></value>
      </member>
      <member>
        <name>PPFailed</name>
        <value><double>0.0</double></value>
      </member>
    </struct></value>
  </member>
  <member>
    <name>delta</name>
    <value><struct>
      <member>
       <name>DNSRateLimit</name>
        <value><double>10.0</double></value>
      </member>
      <member>
        <name>DNSDataRateOut</name>
        <value><double>0.0</double></value>
      </member>
      <member>
        <name>DNSRetries</name>
        <value><double>0.0</double></value>
      </member>
      <member>
        <name>DNSRequests</name>
        <value><double>0.0</double></value>
      </member>
      <member>
        <name>DataRateIn</name>
```

```
          <value><double>0.0</double></value>
        </member>
        <member>
          <name>DNSResponse</name>
          <value><struct>
          </struct></value>
        </member>
        <member>
          <name>DNSRate</name>
          <value><double>0.0</double></value>
        </member>
        <member>
          <name>DataRateOut</name>
          <value><double>0.0</double></value>
        </member>
        <member>
          <name>DocRate</name>
          <value><double>0.0</double></value>
        </member>
        <member>
          <name>DNSRead</name>
          <value><double>0.0</double></value>
        </member>
        <member>
          <name>DNSDataRateIn</name>
          <value><double>0.0</double></value>
        </member>
        <member>
          <name>DNSTimeout</name>
          <value><double>0.0</double></value>
        </member>
        <member>
          <name>DNSWrite</name>
          <value><double>0.0</double></value>
        </member>
        <member>
          <name>DNSResponses</name>
          <value><double>0.0</double></value>
        </member>
      </struct></value>
    </member>
    </struct></value>
    </data></array></value>
  </param>
  </params>
</methodResponse>
```

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.

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

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W