[MS-FSCMW]:
Configuration Middleware 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
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 Configuration Middleware Protocol. This protocol enables a protocol client to obtain configuration information.

1.1 Glossary

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

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

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

Cheetah
Cheetah checksum
content collection
FAST Search Interface Definition Language (FSIDL)
name 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.

[ISO-8601] International Organization for Standardization, "Data Elements and Interchange Formats - Information Interchange - Representation of Dates and Times", ISO/IEC 8601:2004, December 2004, http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=40874&ICS1=1&ICS2=140&ICS3=30

Note There is a charge to download the specification.

[MS-FSCHT] Microsoft Corporation, "Cheetah Data Structure"

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

[MS-FSMW] Microsoft Corporation, "Middleware Protocol Specification"

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

1.2.2 Informative References

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

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

1.3 Overview

This protocol enables a protocol client to manage configuration data that is stored on a protocol server. A protocol client typically stores and retrieves configuration files from the protocol server. This protocol also allows a client to retrieve and update information pertaining to the installation; this information includes **content collection** configuration, component deployment, and item processing configuration.

1.4 Relationship to Other Protocols

This protocol uses the Middleware Protocol, as described in [MS-FSMW], over **HTTP** as shown in the following layering diagram:

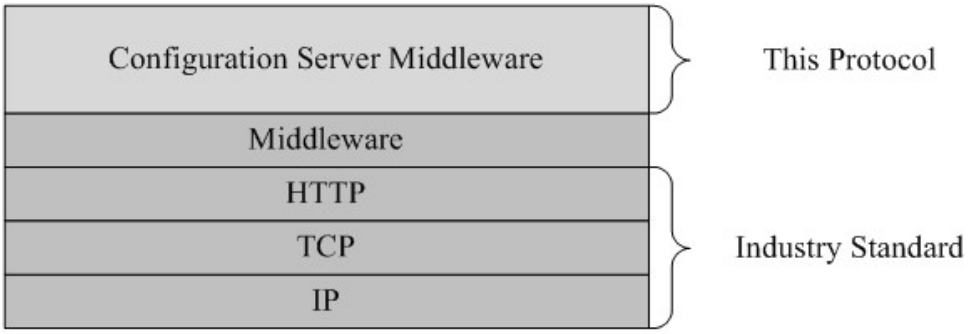

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

None.

1.6 Applicability Statement

This protocol is designed to be used by client applications that need to store configuration information on a protocol server.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

The messages supported by this protocol MUST be sent as HTTP POST messages, as specified in [MS-FSMW].

2.2 Common Data Types

FAST Search Interface Definition Language (FSIDL) data types are encoded as specified in [MS-FSMW] section 2. **Cheetah** entities are encoded as specified in [MS-FSCHT] section 2. The **Cheetah checksum** MUST be an integer with a value of -211918678. The type identifier for each Cheetah entity MUST be an integer, as shown in the following table.

Cheetah entity	Type identifier
host	0
string_sequence	1
host_sequence	2
collection_struct	4

2.2.1 cht::configservice::host

The **host** entity is a structure that the protocol server uses to identify a module. The combination of **fully qualified domain name (FQDN)** and port number MUST be unique.

```
root entity host {
  attribute string name;
  attriute int port;
};
```

name: An FQDN.

port: A port number.

2.2.2 cht::configservice::string_sequence

The **string_sequence** entity is a collection of strings.

```
root entity string_sequence {
  collection string strings;
};
```

strings: A collection of strings.

2.2.3 cht::configservice::host_sequence

The **host_sequence** entity is a collection of **host** entities.

```
root entity host_sequence {
```

```
  collection host hosts;
};
```

hosts: A collection of **host** entities.

2.2.4 cht::configservice::collection_struct

The **collection_struct** entity is a structure that contains attributes of a content collection.

```
root entity collection_struct {
  attribute string name;
  attribute string cluster;
  attribute string description;
  attribute string pipeline;
  attribute string created;
  attribute string cleared;
  attribute host_sequence datasources;
};
```

name: A string that contains the name of the content collection.

cluster: A string that MUST contain the value "webcluster".

description: A string that contains a description for the content collection.

pipeline: A string that MUST contain the value "Office14 (webcluster)".

created: A string that contains the date of creation formatted as specified in [ISO-8601].

cleared: A string that contains the date of last clearance formatted as specified in [ISO-8601].

datasources: A **host_sequence** that contains the data sources for the content collection.

2.2.5 cht::configservice::config_exception

The **config_exception** entity is a wrapper for exceptions thrown at a lower level.

```
exception config_exception {
      string message;
    };
```

message: A string that contains a description of the exception.

3 Protocol Details

This protocol specifies the **configservice::config** interface. This interface requires no additional timers or other state on the protocol client. Calls that the protocol client makes are sent directly to the protocol server, and the results that the protocol server returns are sent directly to the protocol client.

3.1 configservice::config Server Details

The **configservice::config** interface offers methods that an implementation can use to store configuration files on the protocol server. In addition, an implementation can manage the complete lifecycle of content collections and retrieve information about registered components of the installation.

3.1.1 Abstract Data Model

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

The protocol server MUST maintain the following states:

content collections: The current set of content collections in the installation.

rows and columns: An installation can be configured in a row/column configuration for performance and fault-tolerance.

modules: A set of processes that are registered with the protocol server. An installation is comprised of multiple modules. A subset of those processes is registered with the protocol server and will be monitored for responsiveness. The protocol server maintains the state of active versus non-active modules.

3.1.2 Timers

None.

3.1.3 Initialization

The protocol server MUST use the **bind** method, as specified in [MS-FSMW] section 2.2.4.4 to register a configservice::config server object in the **name server**.

The parameters for the **bind** method are encapsulated in an abstract object reference, as described in [MS-FSMW] section 2.2.3.1:

name: A string that MUST contain the value "fds/configservice".

object_id: An integer that MUST be unique for each server object.

host: A string that MUST be the FQDN of the server object on the protocol server.

port: An integer that MUST be the port number of the server object on the protocol server.

interface_type: A string that MUST contain the value "configservice::config".

interface_version: A string that MUST contain the value "5.2".

3.1.4 Message Processing Events and Sequencing Rules

This interface includes the following methods.

Method	Description
create_collection	Creates a new content collection.
get_active_module_list	Retrieves the list of active modules of a given type.
get_clusters	Retrieves all existing clusters.
get_cluster_collections	Retrieves a list of all content collections in a given cluster.
get_collection	Retrieves a content collection by name.
get_collection_cluster	Retrieves the cluster associated with a specified content collection.
get_collection_list	Retrieves a list of all existing content collections.
get_number_of_columns	Retrieves the number of columns in a specified cluster.
get_num_cluster_column_rows	Retrieves the number of rows in a specified column and cluster.
get_pipeline_names	Retrieves a list of all existing content processing pipelines.
load_config_file	Retrieves a stored configuration file.
remove_collection	Deletes a content collection.
reset_procservers	Restarts all registered document processor servers.
save_config_file	Stores a configuration file.
update_collection	Updates the properties of a content collection.

3.1.4.1 create_collection

The **create_collection** method creates a content collection with the specified name.

```
long create_collection(
  in string name,
  in string description,
  in string cluster,
  in string pipeline,
  in cht::configservice::host_sequence datasources
)
```

name: A string that contains the name of the content collection to create. The name MUST NOT be longer than 16 alphanumeric characters.

description: A string that contains a description for the content collection.

cluster: A string that MUST contain the value "webcluster".

pipeline: A string that MUST contain the value "Office14 (webcluster)".

datasources: A **host_sequence** that contains a collection of data sources for the content collection. The data sources MUST already exist.

Return values: A long that indicates whether the collection was created. A value of **1** indicates that the collection was successfully created. In addition, the protocol server MUST implement the processing of the **AddCollection** message as specified in [MS-FSCX] section 3.1.4.1 in order to have the correct alerts (for more information, see the Abstract Data Model in [MS-FSCX] section 3.2.1).

Exceptions: Raises a **SystemException**, as specified in [MS-FSMW] section 2.2.12 in the following situations:

- Existing or invalid content collection names.
- Invalid cluster names.

3.1.4.2 get_active_module_list

The **get_active_module_list** method returns a list of registered modules that are marked as "active" on the protocol server.

```
cht::configservice::host_sequence get_active_module_list(
  in string module_type_name)
```

module_type_name: Represents a string that MUST be one of the following values:

Value
ContentDistributor
DataSource
fdmworker
FilterAlerter
FilterInterface
IndexingDispatcher (**C**)
MLSFileStore
NodeControl
ProcessorServer
Search Dispatcher
Search Engine
SPRel
WaLinkStoreReceiver
WebAnalyzer

Return values: Returns a list of host-port 2-tuples.

Exceptions: Raises no exceptions beyond those that the underlying protocol raises.

3.1.4.3 get_clusters

The **get_clusters** method returns a list of existing cluster names.

```
cht::configservice::string_sequence get_clusters()
```

Return values: Returns a **string_sequence** that contains the **String** "webcluster".

Exceptions: Raises no exceptions beyond those that the underlying protocol raises.

3.1.4.4 get_cluster_collections

The **get_cluster_collections** method returns a list of all content collections in a specified cluster.

```
cht::configservice::string_sequence get_cluster_collections(in string cluster_name)
raises (configservice::config_exception)
```

cluster_name: A string that MUST contain the value "webcluster".

Return values: Returns a **string_sequence** that contains the content collection names.

Exceptions: Raises a **configservice::config_exception** for invalid cluster names.

3.1.4.5 get_collection

The **get_collection** method returns an object representing a specific content collection.

```
cht::configservice::collection_struct get_collection(in string name)
```

name: A string that contains the name of an existing content collection.

Return values: A **collection_struct** object that is initialized with the attributes of the specified content collection.

Exceptions: Returns a **SystemException**, as specified in [MS-FSMW] section 2.2.12, if the name parameter contains a nonexistent content collection name.

3.1.4.6 get_collection_cluster

The **get_collection_cluster** method returns the name of the cluster that is associated with the specified content collection.

```
string get_collection_cluster(in string collection_name)
```

collection_name: A string that contains the name of an existing content collection.

Return values: Returns a string that MUST contain the value "webcluster".

Exceptions: Raises no exceptions beyond those that the underlying protocol raises.

3.1.4.7 get_collection_list

The **get_collection_list** method returns a list of the names of all existing content collections.

```
cht::configservice::string_sequence get_collection_list()
```

Return values: A **string_sequence** of content collection names.

Exceptions: Raises no exceptions beyond those that the underlying protocol raises.

3.1.4.8 get_number_of_columns

The **get_number_of_columns** method returns the number of columns.

```
long get_number_of_columns(in string cluster_name)
```

cluster name: A string that MUST contain the value "webcluster".

Return values: A long that indicates the number of columns in the specified cluster.

Exceptions: Raises a **SystemException**, as specified in [MS-FSMW] section 2.2.12, for invalid cluster names.

3.1.4.9 get_num_cluster_column_rows

The **get_num_cluster_column_rows** method returns the number of rows in a specified column in a cluster.

```
long get_num_cluster_column_rows(
  in string cluster_name,
  in long column_num)
raises (configservice::config_exception)
```

cluster_name: A string that MUST contain the value "webcluster".

column_num: A long that contains the number of a column in the specified cluster.

Return values: Returns the number of rows in a specified column of the cluster.

Exceptions: Raises a **configservice::config_exception** if the cluster name or column number is invalid.

3.1.4.10 get_pipeline_names

The **get_pipeline_names** method returns a list of names for the existing content-processing pipelines.

```
cht::configservice::string_sequence get_pipeline_names()
```

Return values: Returns a **string_sequence** that contains the value "Office14 (webcluster)".

Exceptions: Raises no exceptions beyond those that the underlying protocol raises.

3.1.4.11 load_config_file

The **load_config_file** method returns the contents of a specified file. A protocol client typically uses this method to load the configuration file.

```
string load_config_file(
  in string modulename,
  in string filepath,
  in boolean optional,
  in boolean base64encoded
)
```

modulename: A string that contains the namespace of the configuration file. This value is typically the name of the protocol client module that requests the file.

filepath: A string that contains a relative path to the file, including the file name.

optional: A boolean that MUST be **FALSE** to force the method to fail if the requested file does not exist. The method MUST return an empty **String** if **optional** is set to **TRUE** and the requested file does not exist.

base64encoded: A boolean that indicates whether the file contents use Base 64 encoding. A value of **TRUE** indicates that the protocol client requires the file contents to use Base 64 encoding.

Return values: Returns a string that contains the contents of the specified file.

Exceptions: Raises a **SystemException**, as specified in [MS-FSMW] section 2.2.12, in the following cases:

- The **modulename** is invalid.
- The **filepath** is invalid.
- The **filepath** does not represent an existing file and **optional** is set to **FALSE**.

3.1.4.12 remove_collection

The **remove_collection** method is used to delete a content collection.

```
long remove_collection(in string collection_name)
```

collection_name: A string that contains the name of an existing content collection.

Return values: Returns a long that indicates whether a deletion is successful. A value of 1 indicates a successful deletion. In addition, the protocol server MUST implement the processing of the **RemoveCollection** message as specified in [MS-FSCX] section 3.1.4.31 in order to have the correct alerts (for more information, see the Abstract Data Model in [MS-FSCX] section 3.2.1).

Exceptions: Raises a **SystemException**, as specified in [MS-FSMW] section 2.2.12, for invalid content collection names.

3.1.4.13 reset_procservers

The **reset_procservers** method restarts all registered document processor servers.

```
long reset_procservers()
```

Return values: Returns a long that indicates success of the method call. A value of 1 indicates a successful method call.

Exceptions: Raises no exceptions beyond those that the underlying protocol raises.

3.1.4.14 save_config_file

The **save_config_file** method saves a file on the protocol server. Protocol clients use this method to update a configuration file that they will later read by using the **load_config_file** method.

```
long save_config_file(
  in string modulename,
  in string filepath,
  in string data,
  in boolean base64encoded
)
```

modulename: A string that contains the namespace of the file path. This value is typically the name of the client module that is using the file for configuration.

filepath: A string that contains a relative path to the file, including the file name.

data: A string that contains the contents of the file to be stored.

base64encoded: A boolean that indicates whether the contents of the file use Base 64 encoding. This MUST be **TRUE** if the protocol client has encoded the contents of the file using Base 64 encoding.

Return values: Returns a long that indicates whether the operation was successful. A value of **1** indicates the file was successfully saved. In addition, the protocol server MUST implement the processing of the **SaveConfigFile** message as specified in [MS-FSCX] section 3.1.4.33 in order to have the correct alerts (for more information, see the Abstract Data Model in [MS-FSCX] section 3.2.1).

Exceptions: Raises no exceptions beyond those that the underlying protocol raises.

3.1.4.15 update_collection

The **update_collection** method is used to set all the attributes of a content collection at once.

```
long update_collection(
  in string name,
  in string description,
  in string cluster,
  in string pipeline,
  in string cleared,
  in cht::configservice::host_sequence datasources
)
```

name: A string that contains the name of a content collection. It MAY be a nonexistent name, which will cause a new content collection to be created.

description: A string that contains the description of the content collection.

cluster: A string that MUST contain the value "webcluster".

pipeline: A string that MUST contain the value "Office14 (webcluster)".

cleared: A string that contains the date of last clearance formatted as specified in [ISO-8601].

datasources: A **host_sequence** that contains a collection of data sources for the content collection. The data sources MUST already exist.

Return values: A long that indicates whether the operation was successful. A value of **1** indicates the update was successful. In addition, the protocol server MUST implement the processing of the **UpdateCollection** message as specified in [MS-FSCX] section 3.1.4.38 in order to have the correct alerts (for more information, see the Abstract Data Model in [MS-FSCX] section 3.2.1).

Exceptions: Raises a **SystemException** as specified in [MS-FSMW] section 2.2.12 in the following situations:

- The specified cluster is invalid.
- The specified content processing pipeline is invalid.
- Any of the specified data sources are invalid.

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

4 Protocol Examples

4.1 Binding to a Name Server

The initialization of the protocol server consists of binding to the **fds/configservice** interface of the name server.

The protocol server sends the following request.

```
POST /nameservice::nameserver/1.0/0/bind
```

The POST data contains the following values.

```
example.com
configservice::config
5.2
fds/configservice
```

The name server sends the following response.

```
HTTP response code 200 (OK)
```

The name server responds with data containing the following **Long**.

```
0
```

4.2 Creating a Collection

A protocol client creates a content collection with the following attributes:

name = endtoendcoll

description = test collection

cluster = webcluster

pipeline = Office14 (webcluster)

datasources = None

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/create_collection
```

The POST data contains the following values.

```
entoendcoll
test collection
webcluster
Office14 (webcluster)
```

```
[an emtpy host_sequence]
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following **Long**.

```
1
```

4.3 Retrieving a List of Active Modules

A protocol client determines where all the active document-processing servers that are registered with the protocol server are running.

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/get_active_module_list
```

The POST data contains the following values.

```
ProcessorServer
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following Cheetah **host_sequence**.

```
host_sequence(hosts = [
                  host(name = 'host01.example.com',
                       port = 13395)
              ])
```

4.4 Retrieving Clusters

A protocol client determines which clusters exist in the current installation.

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/get_clusters
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following Cheetah **string_sequence**.

```
string_sequence(strings = ['webcluster'])
```

4.5 Retrieving Collections in a Cluster

A protocol client determines which content collections are associated with the cluster named "webcluster".

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/get_cluster_collections
```

The POST data contains the following value.

```
webcluster
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following Cheetah **string_sequence**.

```
string_sequence(strings = ['truncatecoll', 'sp', 'endtoendcoll'])
```

4.6 Retrieving a Collection

A protocol client retrieves the Cheetah **collection_struct** object that represents the content collection named "sp".

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/get_collection
```

The POST data contains the following value.

```
sp
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following Cheetah **collection_struct**.

```
collection_struct(name = 'sp',
                  cluster = 'webcluster',
                  description = 'Default collection for SharePoint content',
                  pipeline = 'Office14 (webcluster)',
                  datasources = host_sequence(hosts = []))
```

4.7 Retrieving the Cluster for a Collection

A protocol client determines which cluster the content collection named "sp" belongs to.

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/get_collection_cluster
```

The POST data contains the following value.

```
sp
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following **String**.

```
'webcluster'
```

4.8 Retrieving a List of Collections

A protocol client determines the names of all existing content collections.

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/get_collection_list
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following Cheetah **string_sequence**.

```
string_sequence(strings = ['truncatecoll', 'sp', 'endtoendcoll'])
```

4.9 Retrieving the Number of Columns

A protocol client determines how many columns exist in the configuration of the cluster named "webcluster".

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/get_number_of_columns
```

The POST data contains the following value.

```
webcluster
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following **Long**.

```
1
```

4.10 Retrieving the Number of Rows in a Column

A protocol client determines how many rows are configured in the first column of the cluster named "webcluster".

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/get_num_cluster_column_rows
```

The POST data contains the following values.

```
webcluster
0
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following **Long**.

```
1
```

4.11 Retrieving a List of Pipeline Names

A protocol client determines the names of all the existing content-processing pipelines.

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/get_pipeline_names
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following Cheetah **string_sequence**.

```
string _sequence(strings = ['Office14 (webcluster)'])
```

4.12 Loading a Configuration File

A protocol client retrieves the contents of a configuration file named PropertyCategories.xml at the root of the namespace DocumentProcessor. The user does not require an exception to be raised if the file is not stored on the protocol server; nor does the protocol client require the file contents to use Base 64 encoding.

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/load_config_file
```

The POST data contains the following values.

```
DocumentProcessor
PropertyCategories.xml
0
0
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following **String**.

```
'<?xml version="1.0" encoding="utf-8"?>\n\n<properties>\n  <category name="sharepoint"
indexed="yes" discover="yes">\n    <propset name="00020329-0000-0000-c000-000000000046" />\n
<propset name="00130329-0000-0130-c000-000000131346" />\n    <propset name="00140329-0000-
0140-c000-000000141446" />\n  </category>\n  <category name="mail" indexed="yes"
discover="yes">\n    <propset name="aa568eec-e0e5-11cf-8fda-00aa00a14f93" />\n  </category>\n
<category name="office" indexed="yes" discover="yes">\n    <propset name="f29f85e0-4ff9-1068-
ab91-08002b27b3d9" />\n  </category>\n  <category name="basic" indexed="yes"
discover="yes">\n<propset name="0b63e343-9ccc-11d0-bcdb-00805fccce04" />\n    <propset
name="0b63e350-9ccc-11d0-bcdb-00805fccce04" />\n    <propset name="b725f130-47ef-101a-a5f1-
02608c9eebac" />\n    <propset name="49691c90-7e17-101a-a91c-08002b2ecda9" />\n
</category>\n  <category name="web" indexed="yes" discover="yes">\n    <propset
name="70eb7a10-55d9-11cf-b75b-00aa0051fe20" />\n    <propset name="D1B5D3F0-C0B3-11CF-9A92-
00A0C908DBF1" />\n  </category>\n  <category name="outsidein" indexed="yes" discover="yes">\n
<propset name="0f451d82-37a2-4831-a5d7-e5d57c3ad793" />\n  </category>\n  <category
name="mesg linguistics" indexed="no" discover="no">\n    <propset name="48385C54-CDFC-4E84-
8117-C95B3CF8911C" />\n  </category>\n</properties>\n'
```

4.13 Deleting a Collection

A protocol client deletes the content collection named "endtoendcoll".

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/remove_collection
```

The POST data contains the following value.

```
endtoendcoll
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following **Long**.

```
1
```

4.14 Restarting Document Processor Servers

A protocol client restarts all registered document-processing servers.

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/reset_procservers
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following **Long**.

```
1
```

4.15 Saving a Configuration File

A protocol client stores a configuration file named Test.xml at the root of the DocumentProcessor namespace with the following content.

```
<?xml version="1.0" encoding="utf-8"?>
```

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/save_config_file
```

The POST data contains the following values.

```
DocumentProcessor
test.xml
<?xml version="1.0" encoding="utf-8"?>
0
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following **Long**.

```
1
```

4.16 Modifying a Collection

A protocol client changes the description of the content collection named "truncatecoll" to the value "another test collection".

The protocol client sends the following request.

```
POST /configservice::config/5.2/1243942282/update_collection
```

The POST data contains the following values.

```
truncatecoll
another test collection
webcluster
Office14 (webcluster)
[an emtpy host_sequence
```

The protocol server sends the following response.

```
HTTP response code 200 (OK)
```

The response data contains the following **Long**.

```
1
```

5 Security

5.1 Security Considerations for Implementers

Security is resolved in the underlying protocol, as described in [MS-FSMW].

5.2 Index of Security Parameters

None.

6 Appendix A: Full IDL

```
module cht {
module configservice {
typedef sequence<octet> cheetah;
typedef cheetah host;
typedef cheetah string_sequence;
typedef cheetah host_sequence;
typedef cheetah collection_struct;
};
};
module interfaces {
module configservice {
interface config;
exception config_exception {
string message;
};
interface config {
#pragma version config 5.2
cht::configservice::string_sequence get_collection_list();
cht::configservice::string_sequence get_pipeline_names();
cht::configservice::string_sequence get_clusters();
cht::configservice::string_sequence get_cluster_collections(
in string cluster_name)
raises (config_exception);
string get_collection_cluster(
in string collection_name);
long get_number_of_columns(
in string cluster_name);
cht::configservice::host_sequence get_active_module_list(
in string module_type_name);
```

```
long remove_collection(
in string collection_name);
long get_num_cluster_column_rows(
in string cluster_name,
in long column_num)
raises (config_exception);
long update_collection(
in string name,
in string description,
in string cluster,
in string pipeline,
in string cleared,
in cht::configservice::host_sequence datasources);
long create_collection(
in string name,
in string description,
in string cluster,
in string pipeline,
in cht::configservice::host_sequence datasources);
cht::configservice::collection_struct get_collection(in string name);
string load_config_file(
in string modulename,
in string filepath,
in boolean optional,
in boolean base64encoded);
long save_config_file(
in string modulename,
in string filepath,
in string data,
in boolean base64encoded);
```

```
long reset_procservers();
};
};
#ifdef _TAO_IDL_
typeprefix ::interfaces::configservice "";
#endif
};
#endif // dsmodel_configservice_idl
```

7 Appendix B: Product Behavior

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

- Microsoft® FAST™ Search Server 2010

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

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

8 Change Tracking

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

9 Index

V