Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft’s delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. 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

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision History</th>
<th>Revision Class</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>02/19/2010</td>
<td>1.0</td>
<td>Major</td>
<td>Initial Availability</td>
</tr>
<tr>
<td>03/31/2010</td>
<td>1.01</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>04/30/2010</td>
<td>1.02</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>06/07/2010</td>
<td>1.03</td>
<td>Editorial</td>
<td>Revised and edited the technical content</td>
</tr>
<tr>
<td>06/29/2010</td>
<td>1.04</td>
<td>Editorial</td>
<td>Changed language and formatting in the technical content.</td>
</tr>
<tr>
<td>07/23/2010</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>09/27/2010</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>11/15/2010</td>
<td>1.04</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>12/17/2010</td>
<td>1.05</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>03/18/2011</td>
<td>1.05</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>06/10/2011</td>
<td>1.05</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>01/20/2012</td>
<td>1.6</td>
<td>Minor</td>
<td>Clarified the meaning of the technical content.</td>
</tr>
<tr>
<td>04/11/2012</td>
<td>1.6</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>07/16/2012</td>
<td>1.6</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
</tbody>
</table>
Table of Contents

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

2 Messages .............................................................................................................................. 9
   2.1 Transport ....................................................................................................................... 9
   2.2 Common Data Types .................................................................................................... 9
       2.2.1 cht::rtsmessages::exclusionlist_entry ................................................................ 9
       2.2.2 cht::rtsmessages::exclusionlist_entries ......................................................... 9
       2.2.3 rtsearch::indexes_not_ready ............................................................................. 10

3 Protocol Details ................................................................................................................ 11
   3.1 rtsearch::search_controller Server Details ............................................................. 11
       3.1.1 Abstract Data Model ......................................................................................... 11
       3.1.2 Timers ................................................................................................................. 12
       3.1.3 Initialization ....................................................................................................... 12
       3.1.4 Message Processing Events and Sequencing Rules ........................................ 12
           3.1.4.1 notify_new_indexes .................................................................................... 12
           3.1.4.2 add_to_pending_exclusionlist ................................................................. 13
           3.1.4.3 activate_new_indexes ............................................................................... 13
           3.1.4.4 get_fdispatch_ptrport .............................................................................. 14
           3.1.4.5 get_status ................................................................................................. 14
           3.1.4.6 exclusionlist_documents .......................................................................... 15
           3.1.4.7 set_dictionary ........................................................................................... 15
           3.1.4.8 create_generation .................................................................................... 15
           3.1.4.9 commit_exclusionlist ................................................................................ 16
           3.1.4.10 abort_pending_generation ..................................................................... 16
       3.1.5 Timer Events ......................................................................................................... 16
       3.1.6 Other Local Events .............................................................................................. 16
   3.2 rtsearch::search_controller Client Details ............................................................... 16
       3.2.1 Abstract Data Model ......................................................................................... 16
       3.2.2 Timers ................................................................................................................. 17
       3.2.3 Initialization ....................................................................................................... 17
       3.2.4 Message Processing Events and Sequencing Rules ........................................ 17
       3.2.5 Timer Events ....................................................................................................... 17
       3.2.6 Other Local Events ............................................................................................ 17
   3.3 rtsearch::search_master Server Details ................................................................. 17
       3.3.1 Abstract Data Model ......................................................................................... 17
       3.3.2 Timers ................................................................................................................. 18
       3.3.3 Initialization ....................................................................................................... 18
       3.3.4 Message Processing Events and Sequencing Rules ........................................ 18
<table>
<thead>
<tr>
<th></th>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.4.1</td>
<td>connect_controller</td>
<td>19</td>
</tr>
<tr>
<td>3.3.4.2</td>
<td>connect_receiver</td>
<td>19</td>
</tr>
<tr>
<td>3.3.4.3</td>
<td>disconnect</td>
<td>20</td>
</tr>
<tr>
<td>3.3.4.4</td>
<td>get_hostname</td>
<td>20</td>
</tr>
<tr>
<td>3.3.4.5</td>
<td>get_fsearch_cache</td>
<td>20</td>
</tr>
<tr>
<td>3.3.4.6</td>
<td>get_search_overlap</td>
<td>21</td>
</tr>
<tr>
<td>3.3.4.7</td>
<td>get_index_id_set</td>
<td>21</td>
</tr>
<tr>
<td>3.3.4.8</td>
<td>request_exclusionlist_update</td>
<td>21</td>
</tr>
<tr>
<td>3.3.4.9</td>
<td>get_num_exclusionlisted</td>
<td>22</td>
</tr>
<tr>
<td>3.3.5</td>
<td>Timer Events</td>
<td>22</td>
</tr>
<tr>
<td>3.3.6</td>
<td>Other Local Events</td>
<td>22</td>
</tr>
<tr>
<td>3.4</td>
<td>rtsearch::search_master Client Details</td>
<td>22</td>
</tr>
<tr>
<td>3.4.1</td>
<td>Abstract Data Model</td>
<td>22</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Timers</td>
<td>22</td>
</tr>
<tr>
<td>3.4.3</td>
<td>Initialization</td>
<td>22</td>
</tr>
<tr>
<td>3.4.4</td>
<td>Message Processing Events and Sequencing Rules</td>
<td>23</td>
</tr>
<tr>
<td>3.4.5</td>
<td>Timer Events</td>
<td>23</td>
</tr>
<tr>
<td>3.4.6</td>
<td>Other Local Events</td>
<td>23</td>
</tr>
<tr>
<td>4</td>
<td>Protocol Examples</td>
<td>24</td>
</tr>
<tr>
<td>4.1</td>
<td>Enable a Search Controller Node to Retrieve the Fully Qualified Domain Name</td>
<td>24</td>
</tr>
<tr>
<td>4.1.1</td>
<td>get_hostname Transaction Code</td>
<td>24</td>
</tr>
<tr>
<td>4.2</td>
<td>Index Activation</td>
<td>25</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Index Activation Transaction Code</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>Security</td>
<td>31</td>
</tr>
<tr>
<td>5.1</td>
<td>Security Considerations for Implementers</td>
<td>31</td>
</tr>
<tr>
<td>5.2</td>
<td>Index of Security Parameters</td>
<td>31</td>
</tr>
<tr>
<td>6</td>
<td>Appendix A: Full FSIDL</td>
<td>32</td>
</tr>
<tr>
<td>6.1</td>
<td>Full FSIDL</td>
<td>32</td>
</tr>
<tr>
<td>6.2</td>
<td>Cheetah Entities</td>
<td>33</td>
</tr>
<tr>
<td>7</td>
<td>Appendix B: Product Behavior</td>
<td>34</td>
</tr>
<tr>
<td>8</td>
<td>Change Tracking</td>
<td>35</td>
</tr>
<tr>
<td>9</td>
<td>Index</td>
<td>36</td>
</tr>
</tbody>
</table>
1 Introduction

This document specifies the Index Publication and Activation Protocol, which is used to prepare and activate a new index so it becomes searchable by the search application.

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

1.1 Glossary

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

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

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

- abstract object reference (AOR)
- base port
- Cheetah
- Cheetah checksum
- client proxy
- document identifier
- exclusion list
- FAST Search Interface Definition Language (FSIDL)
- host name
- index column
- index generation identifier
- index partition
- indexing node
- item
- name server
- query matching node
- query processing
- search application
- search service application

The following terms are specific to this document:

**search controller node**: A search component that is hosted on a server computer or virtual server and is configured to enable or disable inclusion of one or more search indexes.

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

1.2 References

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

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

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


1.2.2 Informative References

[MS-FSO] Microsoft Corporation, "FAST Search System Overview".


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

1.3 Protocol Overview

This protocol enables a new index to become searchable by the search application. To enable the search application to serve searches while a new index is built, each collection of index data and corresponding dictionaries and exclusion lists is associated with a generation identifier. The generation currently being used by the search application to provide search is named the active generation, while the generation ready to be used for search is named the pending generation. The search application replaces data in the active generation with data from the pending generation in an atomic operation. This is called to activate the pending generation. The following steps and figure describe these operations.

1. Create the new index on the indexing node.

2. Copy the new index, dictionaries, and exclusion lists to the search controller node.

3. Notify the search controller node about the new index.

4. The search controller node creates and prepares the pending generation, collecting and verifying data from the index, dictionaries, and exclusion lists.

5. The search controller starts up a new instance of the search service application using the pending generation not yet activated for search.

6. Activate the pending generation on the search controller node using an atomic operation. The pending generation becomes the active generation.

7. Stop the previous active generation.
Figure 1: Two-phase notify and activate process

The protocols used to perform the actual copying of the new index are described in [MS-FSRFCO] and [MS-FSRFC].

In addition to the methods required to perform the earlier mentioned activation steps, this protocol also enables performing various related administrative tasks, such as retrieving fully qualified domain name (FQDN) and status.

This protocol consists of two different interfaces; the search controller interface and the search master interface. The search controller interface enables an indexing node to talk to a search controller node. The search master interface enables a search controller node to talk to an indexing node. Both these interfaces are specified in this document.

For an overview of the system that this protocol is a part of, see [MS-FSO].

1.4 Relationship to Other Protocols

This protocol uses Middleware, a Hypertext Transfer Protocol (HTTP) based protocol, as described in [MS-FSMW]. Custom data types are encoded over the wire using Cheetah, as described in [MS-FSCHT]. The following diagram shows the relationship of this protocol to other protocols:
1.5 Prerequisites/Preconditions

None.

1.6 Applicability Statement

This protocol is used between a single indexing node and a single search controller node to control and activate new index generations so they become searchable by the search application. This protocol does not atomically switch index generation across multiple indexer or search nodes.

1.7 Versioning and Capability Negotiation

None.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.
2 Messages

2.1 Transport

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

2.2 Common Data Types

FAST Search Interface Definition Language (FSIDL) data types are encoded as defined in [MS-FSMW] section 2. Cheetah entities are encoded as defined in [MS-FSCHT] section 2. The Cheetah checksum MUST be an integer with a value of 135802250. The type identifier for the Cheetah entities MUST be integers as specified in the following table.

<table>
<thead>
<tr>
<th>Cheetah entity</th>
<th>Type identifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>exclusionlist_entry</td>
<td>0</td>
</tr>
<tr>
<td>exclusionlist_entries</td>
<td>1</td>
</tr>
</tbody>
</table>

The full FSIDL for this protocol is provided in section 6.1. The complete listing of Cheetah entities used for this protocol is provided in section 6.2.

2.2.1 cht::rtsmessages::exclusionlist_entry

The exclusionlist_entry Cheetah entity contains information about a specific item to exclude from the index by adding it to an exclusion list, as follows.

```plaintext
entity exclusionlist_entry {
    attribute string document_id;
    attribute int partition;
    attribute longint index_id;
};
```

document_id: A string holding the document identifier (3) of the item.

partition: An integer equal to or greater than zero holding the index partition (2) of the item. An index generation identifier has the format "i_p", where i is the index partition (2).

index_id: An integer equal to or greater than zero holding the last unique part "p" of the index generation identifier that has the format "i_p".

2.2.2 cht::rtsmessages::exclusionlist_entries

The exclusionlist_entries Cheetah entity is a collection of exclusionlist_entry Cheetah entities, containing items to exclude from the index, as follows.

```plaintext
root entity exclusionlist_entries {
    collection exclusionlist_entry entries;
};
```

entries: A collection of exclusionlist_entry Cheetah entities, as specified in section 2.2.1.
2.2.3 rtsearch::indexes_not_ready

The **indexes_not_ready** exception states that the indexing service is not ready, as follows.

```cpp
exception indexes_not_ready {
    string what;
};
```

**what:** A string holding verbose information about the reason for the exception.
3 Protocol Details

This document defines a protocol used between an indexing node and a search controller node, to make a new index searchable by the search application.

This protocol consists of two interfaces, the search controller interface for communication between an indexing node and a search controller node, and the search master interface for communication between a search controller node and an indexing node.

When using the search controller interface, the indexing node is the protocol client and the search controller node is the protocol server. When using the search master interface, the roles are reversed, and the search controller node is the protocol client and the indexing node is the protocol server. This is illustrated in the following figure.

![Figure 3: Interfaces between indexing node and search controller node](image)

3.1 `rtsearch::search_controller` Server Details

A search controller node serving the `search_controller` interface receives messages from an indexing node, and manages the task of making an index searchable by the search service application.

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 search controller interface protocol server MUST maintain the following states:

- **index activation identifier**: A state identifying an ongoing index activation process.
- **active generation**: A state containing information about all data and files related to the current active index generation, including the index, dictionary, and exclusion list. It is used by the search application to serve searches. It MUST be possible to activate the pending generation state by replacing the active generation state with the pending generation state using an atomic operation, making the data from the pending generation state searchable by the search application.

- **pending generation**: A state containing information about all data and files related to a new pending index generation, including the index, dictionary, and exclusion list, based on the newest index, dictionary, and exclusion list found on a search controller node. It MUST be possible to activate the pending generation state by replacing the active generation state with the pending generation state using an atomic operation, making the data from the pending generation state searchable by the search application.
3.1.2 Timers
None.

3.1.3 Initialization
The search controller interface protocol server MUST send a client proxy of itself to the search controller interface protocol client, using a call to the search_master::connect_controller method, as specified in section 3.3.4.1.

3.1.4 Message Processing Events and Sequencing Rules
This interface includes the methods described in the following table.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>notify_new_indexes</td>
<td>Prepares a new index on a search controller node.</td>
</tr>
<tr>
<td>add_to_pending_exclusionlist</td>
<td>Adds items to the exclusion list in the pending generation state.</td>
</tr>
<tr>
<td>activate_new_indexes</td>
<td>Activates the pending generation state.</td>
</tr>
<tr>
<td>get_fdispatch_ptport</td>
<td>Returns the port number used for communication within the search service application.</td>
</tr>
<tr>
<td>get_status</td>
<td>Returns the value of a state holding the status of the search controller node.</td>
</tr>
<tr>
<td>exclusionlist_documents</td>
<td>Adds items to the exclusion list in the pending generation state and activates that state.</td>
</tr>
<tr>
<td>set_dictionary</td>
<td>Sets the dictionary directory in the pending generation state and activates that state.</td>
</tr>
<tr>
<td>create_generation</td>
<td>Creates and initializes a pending generation state.</td>
</tr>
<tr>
<td>commit_exclusionlist</td>
<td>Activates the pending generation state.</td>
</tr>
<tr>
<td>abort_pending_generation</td>
<td>Aborts a pending generation, resetting the pending generation state.</td>
</tr>
</tbody>
</table>

3.1.4.1 notify_new_indexes
The notify_new_indexes method prepares a new index on a search controller node, initializing the pending generation state, as follows.

```c
void notify_new_indexes(in long job_number,
                        in index_id_set_t index_ids);
```

**job_number**: An integer holding a unique number identifying the index activation task used to activate a new index generation. The value of this parameter MUST be a number equal to or greater than zero.

**index_ids**: A sequence of strings, as specified in section 6.1, holding the index generation identifiers of the new index.

**Return values**: None.
Exceptions: No exceptions are raised beyond those raised by the underlying Middleware protocol as specified in [MS-FSMW].

This method MUST initialize the pending generation state, checking that all related data is available, and make the search application ready for activating the pending generation state.

This method MUST store job_number in the index activation identifier state.

3.1.4.2 add_to_pending_exclusionlist

The add_to_pending_exclusionlist method adds items to the exclusion list in the pending generation state if the index activation identifier state is equal to job_number, as follows.

```cpp
void add_to_pending_exclusionlist(in long job_number,
                                 in cht::rtsmessages::exclusionlist_entries docs);
```

job_number: An integer holding a unique number identifying the index activation task used to activate a new index generation. The value of this parameter MUST be a number equal to or greater than zero.

docs: An exclusionlist_entries Cheetah message, as specified in section 2.2.2, holding the items to add to the exclusion list.

Return values: None.

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

This method MUST add the items in docs to the exclusion list in the pending generation state.

This method MUST return without performing any actions specified in this section if job_number is not identical to the index activation identifier state.

3.1.4.3 activate_new_indexes

The activate_new_indexes method activates the pending generation state, if the index activation identifier state is equal to the job_number, as follows.

```cpp
void activate_new_indexes(in long job_number,
                          in index_id_set_t index_ids,
                          in long overlap_time,
                          in long total_num_exclusionlisted);
```

job_number: An integer holding a unique number identifying the index activation task used to activate a new index generation. The value of this parameter MUST be a number equal to or greater than zero.

index_ids: A sequence of strings, as specified in section 6.1, holding the index generation identifiers of the new index.

overlap_time: A search initialized by the search application using data from the active generation state requires this data to be available until the search result has been returned to the search client and the search is finished. The search application hence MUST keep data from the active generation state available for a period of time after the activation of the pending
**generation** state. The number of seconds to keep this overlap is specified by **overlap_time** that MUST be an integer equal to or greater than zero.

**total_num_exclusionlisted**: An integer holding the number of items in the exclusion list, added by previous calls to **add_to_pending_exclusionlist**, as specified in section 3.1.4.2. It MUST be an integer equal to or greater than zero.

**Return values**: None.

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

This method MUST verify that the **total_num_exclusionlisted** is identical to the number of items in the exclusion list on the indexing node. This triggers a call to **search_master::get_num_exclusionlisted** method, as specified in section 3.3.4.9. If there is a mismatch, it MUST request a new exclusion list from the indexing node that triggers a call to the **search_master::request_exclusionlist_update** method, as specified in section 3.3.4.8.

This method MUST return without performing any actions specified in this section if the **job_number** is not identical to the **index activation identifier** state.

### 3.1.4.4 get_fdispatch_ptport

The **get_fdispatch_ptport** method returns the port number used for communication within the search service application, as follows, as follows.

```c++
long get_fdispatch_ptport();
```

**Return values**: An integer that MUST be within the system-specific legal port range.

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

### 3.1.4.5 get_status

The **get_status** method returns the value of a state holding the status of the search controller node, as follows.

```c++
string get_status();
```

**Return values**: A string that MUST be as listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Down</td>
<td>The node is down.</td>
</tr>
<tr>
<td>needs_exclusionlist</td>
<td>The node requires an updated version of the exclusion list.</td>
</tr>
<tr>
<td>Initializing</td>
<td>The node has just started up and is initializing.</td>
</tr>
<tr>
<td>Ok</td>
<td>The node is running normal</td>
</tr>
</tbody>
</table>

**Exceptions**: No exceptions are raised beyond those raised by the underlying Middleware protocol as specified in [MS-FSMW].
This method MUST return the status field in the search controller state.

### 3.1.4.6 exclusionlist_documents

The exclusionlist_documents method adds items to the exclusion list in the pending generation state and activates that state, as follows.

```cpp
void exclusionlist_documents(in cht::rtsmessages::exclusionlist_entries docs);
```

**docs**: An exclusionlist_entries Cheetah entity, as specified in section 2.2.2, holding the items to add to the exclusion list.

**Return values**: None.

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

### 3.1.4.7 set_dictionary

The set_dictionary method sets the dictionary directory in the pending generation state and then activates that state. See section 1.3 for more information about activating the pending generation state, as follows.

```cpp
void set_dictionary(in string directory);
```

**directory**: A string holding the dictionary directory. The value of this parameter MUST be a valid directory path.

**Return values**: None.

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

### 3.1.4.8 create_generation

The create_generation method creates and initializes a pending generation state, as follows.

```cpp
void create_generation(in long job_number, in boolean clear_exclusionlist);
```

**job_number**: An integer holding a unique number identifying the index activation task used to activate a new index generation. The value of this parameter MUST be a number equal to or greater than zero.

**clear_exclusionlist**: A Boolean value. If true, the exclusion list of the pending generation state MUST be reset: If false, no action is required in this regard.

**Return values**: None.

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

The commit_exclusionlist method activates the pending generation state if job_number is equal to the index activation identifier state, as follows.

    void commit_exclusionlist(in long job_number);

job_number: An integer holding a unique number identifying the index activation task used to activate a new index generation. The value of this parameter MUST be a number equal to or greater than zero.

Return values: None.

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

If the job_number is not identical to the index activation identifier state, the method MUST return without performing any actions specified in this section.

3.1.4.10  abort_pending_generation

The abort_pending_generation method aborts a pending generation, resetting the pending generation state, as follows.

    void abort_pending_generation();

Return values: None.

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

This method MUST free up and reset resources allocated in the pending generation state, making the state ready to be used for a new pending generation.

3.1.5  Timer Events

None.

3.1.6  Other Local Events

None.

3.2  rtsearch::search_controller Client Details

An indexing node sends messages on the search_controller interface to a search controller node to manage the task of making an index searchable by the search application.

3.2.1  Abstract Data Model

This section describes a conceptual model of possible data organization that an implementation maintains to participate in this protocol. The described organization is provided to facilitate the explanation of how the protocol behaves. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.
The search controller interface protocol client, the indexing node, MUST maintain the following states:

**search controller**: A state containing information about the search controller nodes that are registered with the indexing node using the `search_master::connect_controller` method, as specified in section 3.3.4.1. It MUST at least contain client proxy references to the `search_controller` interface on the search controller node, and a status field as specified in section 3.1.4.5. This state makes it possible for the indexing node to communicate with individual search controller nodes.

**file receiver**: A state containing client proxy references to the `file_receiver` interface on search controller nodes and indexing nodes that have registered with the indexing node using the `search_master::connect_receiver` method, as specified in section 3.3.4.2. This state makes it possible for the indexing node to communicate with individual nodes that request to receive copies of the index related files. The `file_receiver` interface is specified in [MS-FSRFCO].

### 3.2.2 Timers

None.

### 3.2.3 Initialization

The search controller interface protocol client MUST use the search controller client proxy stored as part of the `search_controller` state to gain access to the search controller interface protocol server.

### 3.2.4 Message Processing Events and Sequencing Rules

None.

### 3.2.5 Timer Events

None.

### 3.2.6 Other Local Events

None.

### 3.3 `rtsearch::search_master` Server Details

An indexing node serving the `search_master` interface receives messages from a search controller node, to enable connectivity between the indexing node and the search controller node on the `search_controller` interface.

#### 3.3.1 Abstract Data Model

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

The search master interface protocol server MUST maintain the `search controller` state and `file receiver` state, as specified in section 3.2.1.
3.3.2 Timers

None.

3.3.3 Initialization

The `search_master` interface protocol server MUST use the Middleware `bind` method to register a `search_master` server object in the `name server`, as specified in [MS-FSMW] section 3.4.4.2.

The parameters for the `bind` method are encapsulated in an abstract object reference (AOR), as specified in [MS-FSMW] section 2.2.18.

**name**: This MUST be a string holding the value "esp/clusters/webcluster/indexing/indexer-C/searchmaster", where C is the index column number.

**object_id**: This MUST be an integer that is unique for each server object.

**host**: A string specifying the host name of the server hosting the server object.

**port**: This MUST be an integer that contains the port number of the server object on the protocol server. The value is base port plus 390.

**interface_type**: This MUST be a string holding the value "rtsearch::search_master".

**interface_version**: This MUST be a string holding the value "5.13".

3.3.4 Message Processing Events and Sequencing Rules

This interface includes the methods described in the following table.

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connect_controller</td>
<td>Registers a client proxy for the <code>search_controller</code> interface in the <code>search controller</code> state.</td>
</tr>
<tr>
<td>connect_receiver</td>
<td>Registers a client proxy for the <code>file_receiver</code> interface in the file receiver state.</td>
</tr>
<tr>
<td>disconnect</td>
<td>Removes the client proxies for the <code>search_controller</code> and file_receiver interfaces from the <code>search controller</code> state and the file receiver state.</td>
</tr>
<tr>
<td>get_hostname</td>
<td>Returns a string holding the FQDN of the indexing node.</td>
</tr>
<tr>
<td>get_fsearch_cache</td>
<td>Returns verbose information about the cache used by the search application for a specific index partition (2).</td>
</tr>
<tr>
<td>get_search_overlap</td>
<td>Returns the number of seconds’ data from the previous active generation state is kept searchable by the search application after activation of the pending generation state.</td>
</tr>
<tr>
<td>get_index_id_set</td>
<td>Returns a comma separated string holding the index generation identifiers of the index.</td>
</tr>
<tr>
<td>request_exclusionlist_update</td>
<td>Requests an update of the exclusion list.</td>
</tr>
<tr>
<td>get_num_exclusionlisted</td>
<td>Returns the total number of items on the exclusion lists across all index partitions (2)</td>
</tr>
</tbody>
</table>
3.3.4.1 connect_controller

The `connect_controller` method registers a client proxy for the `search_controller` interface in the `search controller` state, so the indexing node can communicate with the search controller node using `controller`, as follows.

```java
boolean connect_controller(in search_controller controller, 
                          in string hostname, 
                          in long port, in string role )
```

**controller:** A `search_controller` client proxy, as specified in section 3.1.

**hostname:** A string holding the FQDN of the search controller node.

**port:** An integer holding the port number used for communication within the search service application. This MUST be an integer within the system-specific legal port range.

**role:** A string that MUST be listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>search</td>
<td>The search controller node is a query matching node.</td>
</tr>
<tr>
<td>dispatch</td>
<td>The search controller node is a query processing node, dispatching queries to other query matching nodes.</td>
</tr>
</tbody>
</table>

**Return values:** A Boolean that MUST be `true` if the protocol server was able to register `controller` and no exceptions were caught. Otherwise, it MUST be `false`.

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

This method MUST add `controller` to the `search controller` state, making it possible to identify by `hostname` and `port` as key.

If updated dictionaries exist on the indexing node, a call to the `search_controller::set_dictionary` method, as specified in section 3.1.4.7, on the `controller` client proxy, with the directory holding the updated dictionaries as parameter, MUST be performed.

3.3.4.2 connect_receiver

The `connect_receiver` method registers a client proxy for the `file_receiver` interface in the `file receiver` state, so the file receiver node committing the call can subscribe to copies of index files, as follows.

```java
boolean connect_receiver(in file_receiver receiver, 
                         in string hostname, 
                         in long port);
```

**receiver:** A `file_receiver` client proxy, as specified in [MS-FSRFCO].

**hostname:** A string holding the FQDN of the file receiver node.

**port:** An integer holding the port number used for communication within the search service application. This MUST be an integer within the system-specific legal port range.
Return values: A Boolean that MUST be true if the protocol server was able to register receiver and no exceptions were caught; otherwise, it MUST be false.

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

This method MUST add receiver to the file receiver state, making it possible to identify by hostname and port as key.

3.3.4.3 disconnect

The disconnect method removes the protocol client proxies for the search_controller and file_receiver interfaces from the search_controller state and the file_receiver state, as follows.

    boolean disconnect(string hostname, long port);

hostname: A string holding the FQDN of the node to remove from the search_controller state and file_receiver state.

port: An integer holding the port number used for communication within the search service application. This MUST be an integer within the system-specific legal port range, and MUST be equal to the port number of the node to remove from the search_controller state and file_receiver state.

Return values: A Boolean that MUST always be true.

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

This method MUST remove the entry identified by hostname and port from the search_controller state and file_receiver state.

3.3.4.4 get_hostname

The get_hostname method returns a string holding the FQDN of the indexing node, as follows.

    string get_hostname();

Return values: A string that MUST hold the FQDN of the indexing node.

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

3.3.4.5 get_fsearch_cache

The get_fsearch_cache method returns verbose information about the cache used by the search application for a specific index partition (2), as follows.

    string get_fsearch_cache(long partition_id);

partition_id: An integer identifying an index partition (2). This MUST be an integer equal to or greater than zero.
Return values: A string holding verbose information. The content of the string is implementation-specific of the higher level application.

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

3.3.4.6 get_search_overlap

The get_search_overlap method returns the number of seconds data from the previous active generation state is kept searchable by the search application after activation of the pending generation state, as follows.

```java
long get_search_overlap();
```

Return values: An integer equal to or greater than zero.

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

A search initialized by the search application using data from the active generation state requires this data to be available until the search result has been returned to the search client and the search is finished. The search application therefore MUST keep data from the active generation state available for a period of time after the activation of the pending generation state. The number of seconds to keep this overlap MUST be returned by the get_search_overlap method.

3.3.4.7 get_index_id_set

The get_index_id_set method returns a comma separated string holding the index generation identifiers of the index, as follows.

```java
index_id_set_t get_index_id_set()
  raises (indexes_not_ready);
```

Return values: A comma separated string holding the index generation identifiers of the index.

Exceptions: The exception indexes_not_ready MUST be raised if the index is not ready.

3.3.4.8 request_exclusionlist_update

The request_exclusionlist_update method requests an update of the exclusion list, as follows.

```java
void request_exclusionlist_update(in string controller_host,
   in long controller_port);
```

controller_host: A string holding the FQDN of the search controller node requesting the update.

ccontroller_port: An integer holding the port number of the search controller node requesting the update.

Return values: None

Exceptions: No exceptions are raised beyond those raised by the underlying Middleware protocol as specified in [MS-FSMW].
This method MUST update the **status** field in the **search controller** state to **needs_exclusionlist** to indicate that the search controller node identified by **controller_host** and **controller_port** requires a new copy of the exclusion list from the indexing node. See section **3.1.4.5** for more information about the **status** field.

### 3.3.4.9 get_num_exclusionlisted

The **get_num_exclusionlisted** method returns the total number of items on the exclusion lists across all index partitions (2), as follows.

```cpp
long get_num_exclusionlisted();
```

**Return values:** An integer that MUST be equal to or greater than zero.

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

### 3.3.5 Timer Events

None.

### 3.3.6 Other Local Events

None.

### 3.4 rtsearch::search_master Client Details

A search controller node sends messages on the **search_master** interface to an indexing node to enable the connectivity between the indexing node and the search controller node using the **search_controller** interface.

#### 3.4.1 Abstract Data Model

None.

#### 3.4.2 Timers

None.

#### 3.4.3 Initialization

The search master interface protocol client MUST use the Middleware **resolve** method to find the client proxy to the **search_master** server object bound in the name server, as specified in [MS-FSMW]. The parameters for the **resolve** method are:

**name:** This MUST be a string holding the value "esp/clusters/webcluster=indexing/indexer-C/searchmaster", where C is the index column number.

**interface_type:** This MUST be a string holding the value "rtsearch::search_master".

**version:** This MUST be a string holding the value "5.13".
3.4.4 Message Processing Events and Sequencing Rules
None.

3.4.5 Timer Events
None.

3.4.6 Other Local Events
None.
4 Protocol Examples

4.1 Enable a Search Controller Node to Retrieve the Fully Qualified Domain Name

This example shows how to use the get_hostname method of the search_master interface, as described in section 3.3.4.4, so a search controller node can retrieve the FQDN from an indexing node.

First the protocol server creates a server object implementing the search_master interface, and registers it in the name server. The protocol client then acquires a client proxy to this search_master interface by resolving the server object in the name server. This is possible because both the protocol client and the protocol server have agreed a priori on both the location of the shared name server, and the symbolic name of the server object.

The protocol client is now ready to call the get_hostname method on the search_master client proxy.

4.1.1 get_hostname Transaction Code

Protocol server initialization

```plaintext
SET server_object_instance TO INSTANCE OF search_master SERVER OBJECT
SET server_object_host TO "myserver.mydomain.com"
SET server_object_port TO "1234"
SET server_object_interface_type TO "rtsearch::search_master"
SET server_object_interface_version TO "5.13"
SET server_object_name TO "esp/clusters/webcluster/indexing/indexer-0/searchmaster"
SET server_object_aor TO server_object_host, server_object_port, server_object_interface_type, server_object_interface_version AND server_object_name
CALL nameserver.bind WITH server_object_name AND server_object_aor
```

Protocol client initialization

```plaintext
SET server_object_name TO "esp/clusters/webcluster/indexing/indexer-0/searchmaster"
SET server_object_type TO "rtsearch::search_master"
SET server_object_version TO "5.13"
CALL nameserver.resolve WITH server_object_name, server_object_type AND server_object_version RETURNING search_master_client_proxy
```

Protocol client message

```plaintext
CALL search_master_client_proxy.get_hostname RETURNING hostname
```

Server response
4.2 Index Activation

A search controller node registers as a query matching node with the master indexing node. It also registers as a file receiver node, subscribing to "index" and "dictionary" updates. After a new pending index is created on the master indexing node, it is copied to the search controller node. A list of duplicate items, in this example just one item in partition "1_3", is added to the exclusion list on the newly copied pending index on the search controller node. The search controller node is then told by the master indexing node to activate the new pending index, making it the active index. The new index is now searchable.

Before activating the new index, the search controller node compares the number of items in the received exclusion list to the number on the master indexing node. If they differ, the search controller node sets its status to "needs exclusion list". The master indexing node checks this status, and sends the updated exclusion list to the search controller node upon request. In this example, an item in partition "2_2" is sent.

This example includes methods described in other documents. The file_receiver::data_needed method is described in [MS-FSRFCO] section 3.1.4.2. The methods for copying folders and files are described in [MS-FSRFCO] and [MS-FSIPC].

4.2.1 Index Activation Transaction Code

Indexing node

Register the search_master interface in the name server.

SET bind_server_object_instance TO INSTANCE OF search_master SERVER_OBJECT

SET bind_server_object_host TO "indexer.mydomain.com"
SET bind_server_object_port TO "1001"
SET bind_server_object_interface_type TO "rtsearch::search_master"
SET bind_server_object_interface_version TO "5.13"
SET bind_server_object_name TO "esp/clusters/webcluster/indexing/indexer-0/searchmaster"
SET bind_server_object_aor TO bind_server_object_host, bind_server_object_port, bind_server_object_interface_type, bind_server_object_interface_version AND bind_server_object_name
CALL nameserver.bind WITH bind_server_object_name AND bind_server_object_aor

Search controller node

Resolve the search_master interface from the name server, retrieving the client proxy, enabling the search controller node to communicate with the indexing node.

SET resolve_server_object_name TO "esp/clusters/webcluster/indexing/indexer-0/searchmaster"
SET resolve_server_object_type TO "rtsearch::search_master"
SET resolve_server_object_version TO "5.13"
CALL nameserver.resolve WITH resolve_server_object_name, resolve_server_object_type AND resolve_server_object_version RETURNING search_master_client_proxy
Search controller node

Register as search controller node with the indexing node, enabling the indexing node to communicate with the search controller node.

```
SET search_controller_role TO "search"
SET search_controller_server_object_instance TO INSTANCE OF search_controller SERVER OBJECT
SET search_controller_server_object_host TO "searchcontroller.mydomain.com"
SET search_controller_server_object_port TO "1002"
SET search_controller_server_object_type TO search_controller_role
CALL search_master_client_proxy.connect_controller WITH
    search_controller_server_object_instance,
    search_controller_server_object_host,
    search_controller_server_object_port AND
    search_controller_server_object_type
```

Indexing node

Register the search controller node, identified by the FQDN and port number.

```
SET search_controller_server_object_instance TO
    connect_controller.search_controller_server_object_instance
SET search_controller_host TO
    connect_controller.search_controller_server_object_host
SET search_controller_port TO
    connect_controller.search_controller_server_object_port
SET search_controller_type TO
    search_controller.search_controller_server_object_type
SET search_controller_id TO search_controller_host + ":" + search_controller_port
SET search_controller_state[search_controller_id].instance TO
    search_controller_server_object_instance
SET search_controller_state[search_controller_id].role TO search_controller_type
```

Search controller node

Register as file receiver node with the indexing node, enabling the search controller node to receive files from the indexing node. In this example, the file receiver node subscribes to the "index" and "dictionary" related files and directories.

```
SET file_receiver_subscription TO "subscribe::index" + "subscribe::dictionary"
SET file_receiver_server_object_instance TO INSTANCE OF file_receiver SERVER OBJECT
SET file_receiver_server_object_host TO "searchcontroller.mydomain.com"
SET file_receiver_server_object_port TO "1003"
SET file_receiver_server_object_interface_type TO "rtsearch::file_receiver"
SET file_receiver_server_object_interface_version TO "5.13"
SET file_receiver_server_object_instance.aor TO
    file_receiver_server_object_host,
    file_receiver_server_object_port,
    file_receiver_server_object_interface_type AND
    file_receiver_server_object_interface_version
CALL search_master_client_proxy.connect_receiver WITH
    file_receiver_server_object_instance,
    file_receiver_server_object_host,
```

[MS-FSIPA] — v20120630
Index Publication and Activation Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
file_receiver_server_object_port

**Indexing node**

Register the file receiver node, identified by the FQDN and port number.

```
SET file_receiver_server_object_instance TO connect_receiver.file_receiver_server_object_instance
SET file_receiver_host TO connect_receiver.file_receiver_server_object_host
SET file_receiver_port TO connect_receiver.file_receiver_server_object_port
SET file_receiver_id TO file_receiver_host + "::" + file_receiver_port
SET file_receiver_state[file_receiver_id] TO file_receiver_server_object_instance
```

**Indexing node**

Create a new index.

```
CREATE index
SET index_job TO "1"
SET index_ids TO "0_4 1_3 2_2"
```

**Indexing node**

Copy the new index to registered file receiver nodes that subscribe to the specific subscription type.

```
SET subscription_types TO "subscribe::index",
"subscribe::generation",
"subscribe::counter",
"subscribe::state" AND
"subscribe::dictionary"
FOREACH subscription_type IN subscription_types DO
    FOREACH file_receiver_id IN file_receiver_state DO
        SET datatype TO subscription_type
        CALL file_receiver_state[file_receiver_id].data_needed WITH datatype RETURNING data_needed
        IF data_needed IS TRUE DO
            COPY FILES ASSOCIATED WITH datatype TO file_receiver_state[file_receiver_id]
        END IF
    DONE
DONE
```

**Indexing node**

Notify search controller nodes about the new index.

```
FOREACH search_controller_id IN search_controller_state DO
    CALL search_controller_state[search_controller_id].notify_new_indexes WITH index_job AND index_ids
DONE
```

---

[MS-FSIPA] — v20120630
Index Publication and Activation Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
Search controller node

Initialize the new pending index.

INITIALIZE pending_generation_state INSTANCE
SET index_activation_identifier TO notify_new_indexes.index_job

Indexing node

Add a duplicate item to the pending exclusion lists on search controller nodes for partition "1_3".

SET one_excluded_document TO INSTANCE OF exclusionlist_entry ENTITY
SET one_excluded_document.document_id TO "1234"
SET one_excluded_document.partition TO "1"
SET one_excluded_document.index_id TO "3"
SET excluded_documents TO INSTANCE OF exclusionlist_entries ENTITY
ADD one_excluded_document TO excluded_documents
FOREACH search_controller_id IN search_controller_state DO
  IF search_controller_state[search_controller_id].role IS "search" DO
    SET search_controller TO
    search_controller_state[search_controller_id].instance
    CALL search_controller.add_to_pending_exclusionlist WITH
    index_job AND excluded_documents
  DONE
DONE

Search controller node

Add the item to the exclusion list of the pending index.

SET excluded_documents TO add_to_pending_exclusionlist.excluded_documents
SET index_job TO add_to_pending_exclusionlist.index_job
IF index_job EQUAL index_activation_identifier DO
  ADD excluded_documents TO pending_generation_state
DONE

Indexing node

Activate the pending index on the search controller nodes.

SET overlap_time TO "60"
SET number_of_documents_exclusionlisted TO 1
FOREACH search_controller_id IN search_controller_state DO
  SET search_controller TO
  search_controller_state[search_controller_id].instance
  CALL search_controller.activate_new_indexes WITH
  index_job, index_ids, overlap_time AND
  number_of_documents_exclusionlisted
DONE

Search controller node

Activate the pending index, making it the active index from which new search queries are serviced.
The current active index is kept alive for a specified number of seconds to service search queries.
currently being processed. If the number of items in the exclusion list on the indexing node differs from the received exclusion list, the status is set to request a new version of the exclusion list.

```plaintext
SET index_job TO activate_new_indexes.index_job
SET overlap_time TO activate_new_indexes.overlap_time
SET received_number_of_documents_exclusionlisted TO activate_new_indexes.number_of_documents_exclusionlisted
IF index_job EQUAL index_activation_identifier DO
  CALL search_master_client_proxy.get_num_exclusionlisted RETURNING
    original_number_of_documents_exclusionlisted
  IF original_number_of_documents_exclusionlisted NOT EQUAL
    received_number_of_documents_exclusionlisted DO
    CALL search_master_client_proxy.request_exclusionlist_update WITH
      search_controller_server_object_host AND
      search_controller_server_object.port
    SET search_controller.status TO "needs_exclusionlist"
  DONE
  DEACTIVATE active_generation_state AND STOP AFTER overlap_time SECONDS
  SET active_generation_state TO pending_generation_state
  DONE

Indexing node

Send updated exclusion list for document in partition "2_2" to search controller nodes that request new copies.

SET one_excluded_document TO INSTANCE OF exclusionlist_entry ENTITY
SET one_excluded_document.document_id TO "5678"
SET one_excluded_document.partition TO "2"
SET one_excluded_document.index_id TO "2"
SET excluded_documents TO INSTANCE OF exclusionlist_entries ENTITY
ADD one_excluded_document TO excluded_documents
FOREACH search_controller_id IN search_controller_state DO
  SET search_controller TO
    search_controller_state[search_controller_id].instance
  IF search_controller.get_status IS "needs_exclusionlist" DO
    IF search_controller_state[search_controller_id].role IS "search" DO
      SET index_job TO "0"
      SET clear_exclusionlist TO "true"
      CALL search_controller.create_generation WITH
        index_job AND clear_exclusionlist
      CALL search_controller.add_to_pending_exclusionlist WITH
        index_job AND excluded_documents
      CALL search_controller.commit_exclusionlist WITH index_job
      SET search_controller_state[search_controller_id].status TO "OK"
    DONE
  DONE
DONE
DONE

Search controller node

Update active index with updated exclusion list.

IF search_controller_role IS "search" DO
  INITIALIZE pending_generation_state INSTANCE
  SET excluded_documents TO add_to_pending_exclusionlist.excluded_documents
```

29 / 37

[MS-FSIPA] — v20120630
Index Publication and Activation Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
SET index_job TO add_to_pending_exclusionlist.index_job
IF index_job EQUAL index_activation_identifier DO
   ADD excluded_documents TO pending_generation_state
DONE
SET active_generation_state TO pending_generation_state
DONE
5 Security

5.1 Security Considerations for Implementers

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

5.2 Index of Security Parameters

None.
6 Appendix A: Full FSIDL

For ease of implementation, the full FSIDL and complete listing of Cheetah entities used in this protocol are provided in the following sections.

6.1 Full FSIDL

```cpp
module cht {
    module rtsmessages {
        typedef sequence<octet> cheetah;
        typedef cheetah exclusionlist_entries;
    };
};

module interfaces {
    module rtsearch {
        typedef sequence<string> index_id_set_t;
        exception indexes_not_ready {
            string what;
        }
        interface search_controller {
            #pragma version search_controller 5.20
            void notify_new_indexes(in long job_number,
                                    in index_id_set_t index_ids);
            void add_to_pending_exclusionlist(in long job_number,
                                               in cht::rtsmessages::exclusionlist_entries docs);
            void activate_new_indexes(in long job_number,
                                       in index_id_set_t index_ids,
                                       in long overlap_time,
                                       in long total_num_exclusionlisted);
            long get_fdispatch_ptport();
            string get_status();
            void exclusionlist_documents(in cht::rtsmessages::exclusionlist_entries docs);
            void set_dictionary(in string directory);
            void create_generation(in long job_number, in boolean clear_exclusionlist);
            void commit_exclusionlist(in long job_number);
            void abort_pending_generation();
        };
        interface search_master {
            #pragma version search_master 5.13
            boolean connect_controller(in search_controller controller,
                                        in string hostname,
                                        in string h...
```
in long port,
in string role);

boolean connect_receiver(in file_receiver receiver,
in string hostname,
in long port);

boolean disconnect(in string hostname, in long port);

string get_hostname();

long get_data_transfer_port();

string get_fsearch_cache(in long partition_id);

long get_search_overlap();

index_id_set_t get_index_id_set()
  raises (indexes_not_ready);

void request_exclusionlist_update(in string controller_host,
in long controller_port);

long get_num_exclusionlisted();

};

};

};

6.2 Cheetah Entities

entity exclusionlist_entry {
  attribute string document_id;
  attribute int partition;
  attribute longint index_id;
};

root entity exclusionlist_entries {
  collection exclusionlist_entry entries;
};
7 Appendix B: Product Behavior

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

- Microsoft® FAST™ Search Server 2010

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

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

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

A

abort_pending_generation method 16
Abstract data model
  client (section 3.2.1 16, section 3.4.1 22)
  server (section 3.1.1 11, section 3.3.1 17)
activate_new_indexes method 13
add_to_pending_exclusionlist method 13
Applicability 8

C

Capability negotiation 8
Change tracking 35
tcht::rtsmessages::exclusionlist_entries data type 9
tcht::rtsmessages::exclusionlist_entry data type 9
Client
  abstract data model (section 3.2.1 16, section 3.4.1 22)
  initialization (section 3.2.3 17, section 3.4.3 22)
  local events (section 3.2.6 17, section 3.4.6 23)
  message processing (section 3.2.4 17, section 3.4.4 23)
  overview (section 3 11, section 3.2 16, section 3.4 22)
  rtsearch::search_controller interface 16
  rtsearch::search_master interface 22
  sequencing rules (section 3.2.4 17, section 3.4.4 23)
  timer events (section 3.2.5 17, section 3.4.5 23)
  timers (section 3.2.2 17, section 3.4.2 22)
commit_exclusionlist_method 16
Common
  overview 11
  Common data types 9
  connect_controller method 19
  connect_receiver method 19
  create_generation method 15

D

Data model - abstract
  client (section 3.2.1 16, section 3.4.1 22)
  server (section 3.1.1 11, section 3.3.1 17)
Data types
tcht::rtsmessages::exclusionlist_entries 9
tcht::rtsmessages::exclusionlist_entry 9
  common - overview 9
  rtsearch::indexes_not_ready 10
disconnect method 20

E

Enable a search controller node to retrieve the fully qualified domain name example 24
get_hostname transaction code 24
Events
  local - client (section 3.2.6 17, section 3.4.6 23)
  local - server (section 3.1.6 16, section 3.3.6 22)
timer - client (section 3.2.5 17, section 3.4.5 23)
timer - server (section 3.1.5 16, section 3.3.5 22)
Examples
  enable a search controller node to retrieve the fully qualified domain name 24
  index activation 25
  exclusionlist_documents_method 15

F

Fields - vendor-extensible 8
FSIDL 32
Full FSIDL 32

G

gle dispatch_ptport method 14
gfsearch_cache method 20
gf_hostname_method 20
gf_hostname_transaction_code example 24
gf_index_id_set_method 21
gf_num_exclusionlisted_method 22
gf_search_overlap_method 21
gf_status_method 14
Glossary 5

I

Implementer - security considerations 31
Index activation example 25
Index of security parameters 31
Informative references 6
Initialization
  client (section 3.2.3 17, section 3.4.3 22)
  server (section 3.1.3 12, section 3.3.3 18)
Interfaces - client
  rtsearch::search_controller 16
  rtsearch::search_master 22
Interfaces - server
  rtsearch::search_controller 11
  rtsearch::search_master 17
Introduction 5

L

Local events
  client (section 3.2.6 17, section 3.4.6 23)
  server (section 3.1.6 16, section 3.3.6 22)

M

Message processing
  client (section 3.2.4 17, section 3.4.4 23)
  server (section 3.1.4 12, section 3.3.4 18)
Messages
tcht::rtsmessages::exclusionlist_entries data type 9
tcht::rtsmessages::exclusionlist_entry data type 9
common data types 9
rtsearch::indexes_not_ready data type 10
transport 9

Methods
abort_pending_generation 16
activate_new_indexes 13
add_to_pending_exclusionlist 13
commit_exclusionlist 16
connect_controller 19
connect_receiver 19
create_generation 15
disconnect 20
exclusionlist_documents 15
get_fdispatch_ptpport 14
get_fsearch_cache 20
get_hostname 20
get_index_id_set 21
get_num_exclusionlisted 22
get_status 14
notify_new_indexes 12
request_exclusionlist_update 21
set_dictionary 15

N
Normative references 6
notify_new_indexes method 12

O
Overview (synopsis) 6

P
Parameters - security index 31
Preconditions 8
Prerequisites 8
Product behavior 34

R
References 5
informative 6
normative 6
Relationship to other protocols 7
request_exclusionlist_update method 21
rtsearch::indexes_not_ready data type 10
rtsearch::search_controller interface (section 3.1
11, section 3.2 16)
rtsearch::search_master interface (section 3.3 17, section 3.4 22)

S
Security
implementer considerations 31
parameter index 31
Sequencing rules
client (section 3.2.4 17, section 3.4.4 23)
server (section 3.1.4 12, section 3.3.4 18)
Server

T
Timer events
client (section 3.2.5 17, section 3.4.5 23)
server (section 3.1.5 16, section 3.3.5 22)
Timers
client (section 3.2.2 17, section 3.4.2 22)
server (section 3.1.2 12, section 3.3.2 18)
Tracking changes 35
Transport 9

V
Vendor-extensible fields 8
Versioning 8