[MS-FSDP]:
Document Processing 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 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.

Copyright © 2012 Microsoft Corporation.

Revision Summary

Date	Revision History	Revision Class	Comments
11/06/2009	0.1	Major	Initial Availability
02/19/2010	1.0	Major	Updated and revised 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.
01/20/2012	1.5	Minor	Clarified the meaning of the technical content.
04/11/2012	1.5	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	1.5	No change	No changes to the meaning, language, or formatting of the technical content.

Copyright © 2012 Microsoft Corporation.

Table of Contents

Copyright © 2012 Microsoft Corporation.

Copyright © 2012 Microsoft Corporation.

Copyright © 2012 Microsoft Corporation.

1 Introduction

This document specifies the Document Processing Protocol, which is used between components in a search service application. This protocol enables certain components to submit items to another component for indexing.

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]:

attribute
UTF-8

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

abstract object reference (AOR)
base port
callback message
CDATA section
Cheetah
Cheetah checksum
Cheetah entity
client proxy
content client
content collection
content distributor
document identifier
FAST Index Markup Language (FIXML)
FAST Search Interface Definition Language (FSIDL)
host name
indexing dispatcher
indexing node
item
item processing
name server
search index
XML Path Language (XPath)

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

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.

Copyright © 2012 Microsoft Corporation.

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-FSCF] Microsoft Corporation, "Content Feeding Protocol Specification".

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

[MS-FSDPD] Microsoft Corporation, "Document Processing Distribution Protocol Specification".

[MS-FSFIXML] Microsoft Corporation, "FIXML Data Structure".

[MS-FSID] Microsoft Corporation, "Indexing Distribution 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 Protocol Overview (Synopsis)

The Document Processing Protocol enables a **content distributor** and an item processor to send **item** operations to an **indexing dispatcher** as part of a session-based item feeding chain. The process consists of three main steps: creating a new session, feeding item operations to the indexing dispatcher during the session, and retrieving status information about the items.

The components in the overall feeding chain are the **content client**, the content distributor, the item processor, the indexing dispatcher, and the **indexing nodes**. This protocol defines the communication between the content distributor and the indexing dispatcher as well as between the item processor and the indexing dispatcher, as illustrated in the following figure.

Copyright © 2012 Microsoft Corporation.

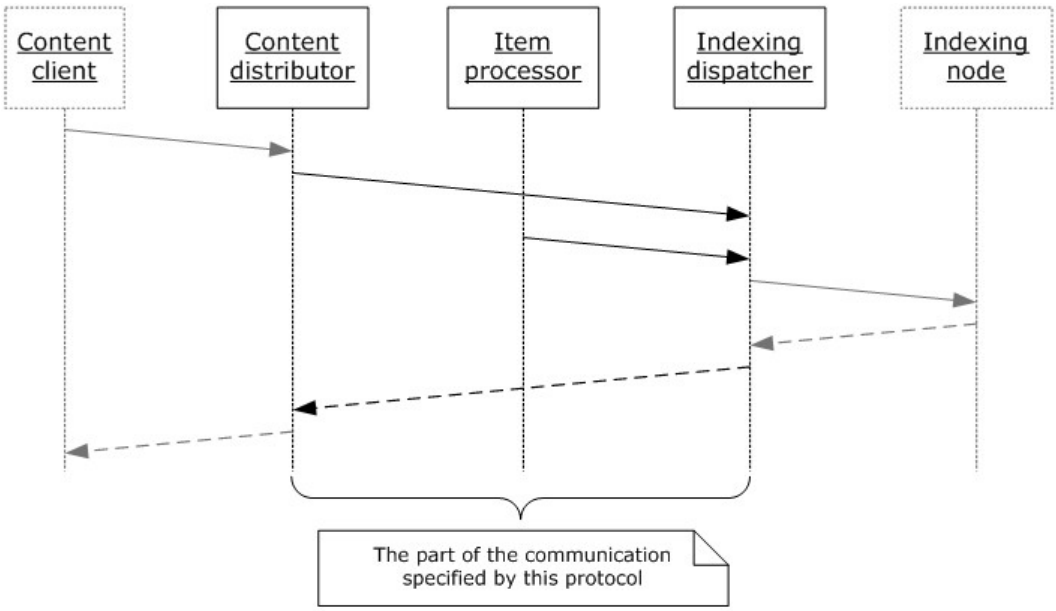

Figure 1: Components that use this protocol

A typical scenario for using this protocol involves a content client that traverses a file system. The content client submits newly recognized files from the file system to the content distributor as item-add operations. The content client submits previously updated files to the content distributor as item-update operations. For files that are no longer available, the content client submits item-remove operations.

The content distributor then submits the item operations to one or more item processors. After processing the items, each item processor submits the item operations to the indexing dispatcher. The indexing dispatcher sends **callback messages** to the content distributor after it has stored the item operations on disk and indexed the items. Finally, the content distributor sends the callback messages to the content client.

The content client uses the callback messages to log the progress in the feeding chain. The content client also notifies the application user of possible errors that occurred during the processing and indexing of item operations.

Both the content distributor and the indexing dispatcher have dual roles as protocol clients and protocol servers with regard to this protocol. For the interfaces specified in coreprocessing::session_factory Server Details (section 3.1) and coreprocessing::session Server Details (section 3.3), the content distributor acts as the protocol client, and the indexing dispatcher acts as the protocol server. For the interface specified in coreprocessing::operation_callback Server Details (section 3.4), the roles are reversed: the indexing dispatcher acts as the protocol client, and the content distributor acts as the protocol server. The item processor also acts as a protocol client for the interface specified in coreprocessing::session Server Details (section 3.3).

Copyright © 2012 Microsoft Corporation.

1.4 Relationship to Other Protocols

The Document Processing Protocol relies on the Cheetah Data Format to serialize data, as described in [MS-FSCHT], and on the Middleware Protocol to transport data, as described in [MS-FSMW]. The following diagram shows the underlying messaging and transport stack that this protocol uses:

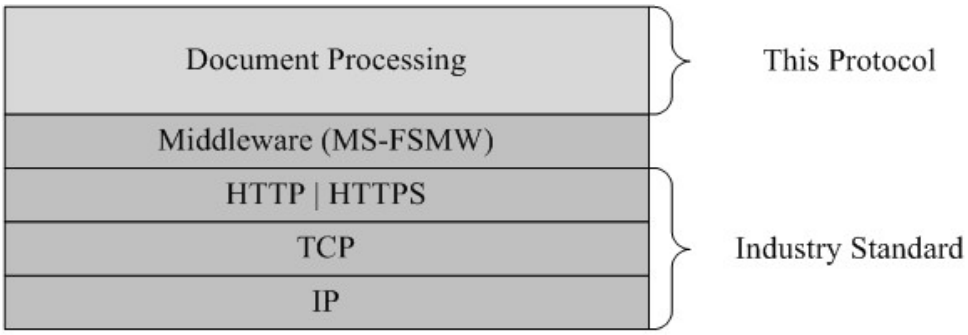

Figure 2: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

The protocol client and protocol server are expected to know the location and connection information of the shared **name server**.

1.6 Applicability Statement

The Document Processing Protocol is designed for submitting item operations from a content distributor and an item processor to an indexing dispatcher. The indexing dispatcher uses callback messages to return information to the content distributor about the indexing status of the submitted item operations.

1.7 Versioning and Capability Negotiation

Regarding capability negotiation:

- The Middleware Protocol, as described in [MS-FSMW], requires that the protocol server and the protocol client agree on the server interface version for all method invocations.
- The Cheetah Data Format, as described in [MS-FSCHT], requires that the protocol server and the protocol client agree on the Cheetah type identifiers and **Cheetah checksum** for all the **Cheetah entities** that are used by this protocol.

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

Copyright © 2012 Microsoft Corporation.

2 Messages

2.1 Transport

Messages MUST be transported via the Middleware Protocol, as specified in [MS-FSMW]. Data serialization MUST be performed by using the Cheetah Data Format, as specified in [MS-FSCHT].

2.2 Common Data Types

The messages for this protocol are specified by using **FAST Search Interface Definition Language (FSIDL)**. The allowed FSIDL data types, as specified in [MS-FSMW], are encoded as specified in [MS-FSMW] section 2. Cheetah entities are encoded as specified in [MS-FSCHT] section 2. The Cheetah checksum and Cheetah type identifier for each Cheetah entity MUST both be integers, as specified in the following table.

Cheetah entity	Cheetah type identifier	Cheetah checksum
cht::core::guarantee	3	1479218033
cht::core::feeding_priority	7	1479218033
cht::core::guarantee_set	9	1479218033
cht::documentmessages::key_value_pair	0	211918678
cht::documentmessages::key_value_collection	1	211918678
cht::documentmessages::bytearray_attribute	4	211918678
cht::documentmessages::warning	6	211918678
cht::documentmessages::operation	7	211918678
cht::documentmessages::clear_collection	9	211918678
cht::documentmessages::document_id	11	211918678
cht::documentmessages::document	12	211918678
cht::documentmessages::error	15	211918678
cht::documentmessages::failed_operation	18	211918678
cht::documentmessages::processing_error	21	211918678
cht::documentmessages::format_error	22	211918678
cht::documentmessages::indexing_error	23	211918678
cht::documentmessages::internal_partial_update_operation	25	211918678
cht::documentmessages::string_attribute	26	211918678
cht::documentmessages::insert_xml	27	211918678
cht::documentmessages::integer_attribute	28	211918678
cht::documentmessages::internal_partial_update	32	211918678

Copyright © 2012 Microsoft Corporation.

Cheetah entity	Cheetah type identifier	Cheetah checksum
cht::documentmessages::invalid_content	33	211918678
cht::documentmessages::operation_dropped	36	211918678
cht::documentmessages::operation_lost	37	211918678
cht::documentmessages::operation_set	38	211918678
cht::documentmessages::subsystem_id_set	39	211918678
cht::documentmessages::operation_status_info	40	211918678
cht::documentmessages::remove_nodes	44	211918678
cht::documentmessages::remove_operation	45	211918678
cht::documentmessages::resource_error	46	211918678
cht::documentmessages::server_unavailable	47	211918678
cht::documentmessages::string_replace	50	211918678
cht::documentmessages::unknown_document	51	211918678
cht::documentmessages::update_operation	52	211918678
cht::documentmessages::utf8_error	54	211918678
cht::documentmessages::xml_error	55	211918678

The following subsections specify the details of these Cheetah entities.

2.2.1 cht::core::guarantee_set

The **guarantee_set** Cheetah entity contains a collection of **guarantee** objects, which are specified in cht::core::guarantee (section 2.2.2). Cheetah entity specification for **guarantee_set**:

```
entity guarantee_set {
    collection guarantee guarantees;
};
```

guarantees: A collection of **guarantee** Cheetah entities.

2.2.2 cht::core::guarantee

The **guarantee** Cheetah entity is a parent class for the **feeding_priority** Cheetah entity, which is specified in cht::core::feeding_priority (section 2.2.3).Cheetah entity specification for **guarantee:**

```
entity guarantee {
};
```


Copyright © 2012 Microsoft Corporation.

2.2.3 cht::core::feeding_priority

The **feeding_priority** Cheetah entity specifies the priority for sending items to the protocol server. Cheetah entity specification for **feeding_priority**:

```
entity feeding_priority : guarantee {
    attribute int priority;
};
```

priority: An integer that MUST be 0.

2.2.4 cht::documentmessages::action

The **action** **Cheetah** enumeration specifies actions that are used in error messages. Cheetah enumeration specification for **action**:

```
enum action {
    resubmit,
    limited_resubmit,
    drop,
    terminate
};
```

resubmit: A constant specifying that the protocol client MUST resubmit the item operation.

limited_resubmit: A constant specifying that the protocol client MUST resubmit the item operation for a limited number of times.

drop: A constant specifying that the protocol client MUST NOT resubmit the item operation.

terminate: A constant that the protocol client MUST NOT use.

2.2.5 cht::documentmessages::operation_state

The **operation_state** Cheetah enumeration specifies the possible states of an item operation. Cheetah enumeration specification for **operation_state**:

```
enum operation_state {
    unknown,
    received,
    secured,
    completed,
    lost
};
```

unknown: A constant specifying that the item operation is in an unknown state.

received: A constant specifying that the protocol server has received the item operation.

secured: A constant specifying that the item operation has been saved to disk.

completed: A constant specifying that the item operation has finished running.

lost: A constant specifying that the item operation was lost during processing or indexing.

Copyright © 2012 Microsoft Corporation.

2.2.6 cht::documentmessages::error

The **error** Cheetah entity contains error information for a specific item operation. Cheetah entity specification for **error**:

```
entity error {
    attribute int error_code;
    attribute action suggested_action;
    attribute string description;
    attribute string subsystem;
    attribute int session_id;
    attribute longint operation_id;
    collection string arguments;
};
```

error_code: An integer that contains the error code.

suggested_action: An **action** Cheetah enumeration value, as specified in cht::documentmessages::action (section 2.2.4), containing the suggested action that the protocol client can perform to correct the item operation error.

description: A string that contains a description of the error.

subsystem: A string that describes where the error occurred. This string MUST have a value of either "indexing" or "processing". If the error was produced by either the content distributor or the item processor, the string value will be "processing". If the error was produced by either the indexing dispatcher or an indexing node, the string value will be "indexing".

session_id: An integer that uniquely identifies the session.

operation_id: An integer that uniquely identifies the item operation.

arguments: Unused. The value MUST be an empty Cheetah collection.

2.2.7 cht::documentmessages::processing_error

The **processing_error** Cheetah entity specifies when an error occurred during the processing of an item operation.

The **processing_error** Cheetah entity is a subclass of the **error** Cheetah entity, which is specified in cht::documentmessages::error (section 2.2.6). The **processing_error** Cheetah entity is also a common superclass for:

- The **format_error** Cheetah entity, which is specified in section cht::documentmessages::format_error (section 2.2.8).
- The **server_unavailable** Cheetah entity, which is specified in section cht::documentmessages::server_unavailable (section 2.2.11).
- The **operation_dropped** Cheetah entity, which is specified in section cht::documentmessages::operation_dropped (section 2.2.12).

Cheetah entity specification for **processing_error**:

```
entity processing_error : error {
    attribute string processor;
```


Copyright © 2012 Microsoft Corporation.

```
};
```

processor: A string that specifies the name of the item processor stage where the error occurred.

2.2.8 cht::documentmessages::format_error

The **format_error** Cheetah entity is used to indicate that an item operation has an invalid format.

The **format_error** Cheetah entity is a subclass of the **processing_error** Cheetah entity, which is specified in cht::documentmessages::processing_error (section 2.2.7). The **format_error** Cheetah entity is also a common superclass for the **xml_error** Cheetah entity, which is specified in cht::documentmessages::xml_error (section 2.2.9), and the **utf8_error** Cheetah entity, which is specified in cht::documentmessages::utf8_error (section 2.2.10). Cheetah entity specification for **format_error**:

```
entity format_error : processing_error {
};
```

2.2.9 cht::documentmessages::xml_error

The **xml_error** Cheetah entity is used to indicate that an item operation contains invalid XML that is not valid.

The **xml_error** Cheetah entity is a subclass of the **format_error** Cheetah entity, which is specified in cht::documentmessages::format_error (section 2.2.8). Cheetah entity specification for **xml_error**:

```
entity xml_error : format_error {
};
```

2.2.10 cht::documentmessages::utf8_error

The **utf8_error** Cheetah entity is used to indicate that an item operation contains invalid UTF-8 encoding.

The **utf8_error** Cheetah entity is a subclass of the **format_error** Cheetah entity, which is specified in cht::documentmessages::format_error (section 2.2.8). Cheetah entity specification for **utf8_error**.

```
entity utf8_error : format_error {
};
```

2.2.11 cht::documentmessages::server_unavailable

The **server_unavailable** Cheetah entity is used to indicate that a protocol client was unable to connect to a protocol server during the processing of an item operation.

The **server_unavailable** Cheetah entity is a subclass of the **processing_error** Cheetah entity, which is specified in cht::documentmessages::processing_error (section 2.2.7). Cheetah entity specification for **server_unavailable**:

Copyright © 2012 Microsoft Corporation.

```
entity server_unavailable : processing_error {
};
```

2.2.12 cht::documentmessages::operation_dropped

The **operation_dropped** Cheetah entity is used to indicate that the item processor has identified an item operation that MUST NOT be indexed.

The **operation_dropped** Cheetah entity is a subclass of the **processing_error** Cheetah entity, which is specified in cht::documentmessages::processing_error (section 2.2.7). Cheetah entity specification for **operation_dropped**:

```
entity operation_dropped : processing_error {
};
```

2.2.13 cht::documentmessages::operation_lost

The **operation_lost** Cheetah entity is used to indicate that an item operation has been lost during processing or indexing.

The **operation_lost** Cheetah entity is a subclass of the **error** Cheetah entity, which is specified in cht::documentmessages::error (section 2.2.6). Cheetah entity specification for **operation_lost**:

```
entity operation_lost : error {
};
```

2.2.14 cht::documentmessages::indexing_error

The **indexing_error** Cheetah entity is used to indicate that an error occurred during the indexing of an item operation.

The **indexing_error** Cheetah entity is a subclass of the **error** Cheetah entity, which is specified in cht::documentmessages::error (section 2.2.6). The **indexing_error** Cheetah entity is also a common superclass for:

- The **invalid_content** Cheetah entity, which is specified in cht::documentmessages::invalid_content (section 2.2.15).
- The **resource_error** Cheetah entity, which is specified in cht::documentmessages::error (section 2.2.6).
- The **unknown_document** Cheetah entity, which is specified in cht::documentmessages::unknown_document (section 2.2.17).

Cheetah entity specification for **indexing_error**:

```
entity indexing_error : error {
};
```


Copyright © 2012 Microsoft Corporation.

2.2.15 cht::documentmessages::invalid_content

An indexing node uses the **invalid_content** Cheetah entity to indicate that an item operation contains content that is not valid.

The **invalid_content** Cheetah entity is a subclass of the **indexing_error** Cheetah entity, which is specified in cht::documentmessages::indexing_error (section 2.2.14). Cheetah entity specification for **invalid_content**:

```
entity invalid_content : indexing_error {
};
```

2.2.16 cht::documentmessages::resource_error

An indexing node uses the **resource_error** Cheetah entity to indicate that a resource error occurred during the indexing of an item operation.

The **resource_error** Cheetah entity is a subclass of the **indexing_error** Cheetah entity, which is specified in cht::documentmessages::indexing_error (section 2.2.14). Cheetah entity specification for **resource_error**:

```
entity resource_error : indexing_error {
};
```

2.2.17 cht::documentmessages::unknown_document

An indexing node uses the **unknown_document** Cheetah entity to indicate that a **remove_operation** Cheetah entity refers to an item that does not exist in the index.

The **unknown_document** Cheetah entity is a subclass of the **indexing_error** Cheetah entity, which is specified in cht::documentmessages::indexing_error (section 2.2.14). Cheetah entity specification for **unknown_document**:

```
entity unknown_document : indexing_error {
};
```

2.2.18 cht::documentmessages::warning

The **warning** Cheetah entity contains warning information about a specific item operation.Cheetah entity specification for **warning**:

```
entity warning {
    attribute int warning_code;
    attribute string description;
    attribute string subsystem;
    attribute int session_id;
    attribute longint operation_id;
};
```

warning_code: An integer that indicates the warning code.

description: A string that contains a description of the warning.

Copyright © 2012 Microsoft Corporation.

subsystem: A string that describes where the warning occurred. This string MUST have a value of either "indexing" or "processing". If the warning was produced by either the content distributor or the item processor, the string value will be "processing". If the warning was produced by either the indexing dispatcher or an indexing node, the string value will be "indexing".

session_id: An integer that uniquely identifies the session.

operation_id: An integer that uniquely identifies the item operation.

2.2.19 cht::documentmessages::operation

The **operation** Cheetah entity is a common superclass for:

- The **update_operation** Cheetah entity, which is specified in cht::documentmessages::update_operation (section 2.2.29).
- The **internal_partial_update_operation** Cheetah entity, which is specified in cht::documentmessages::internal_partial_update_operation (section 2.2.35).
- The **remove_operation** Cheetah entity, which is specified in cht::documentmessages::remove_operation (section 2.2.31).

Cheetah entity specification for **operation**:

```
entity operation {
    attribute longint id;
    collection warning warnings;
};
```

id: A long integer that uniquely identifies the item operation. The value MUST be equal to or greater than 0.

warnings: A collection of **warning** Cheetah entities, which are specified in cht::documentmessages::warning (section 2.2.18). This collection contains all the warnings for the item operation that is identified by the **id** attribute.

2.2.20 cht::documentmessages::operation_set

The **operation_set** Cheetah entity contains a set of **operation** objects, specified in cht::documentmessages::operation (section 2.2.19). Cheetah entity specification for **operation_set**:

```
entity operation_set {
    attribute longint completed_op_id;
    collection operation operations;
};
```

completed_op_id: A long integer that contains the highest operation identifier in the sequence of operation identifiers for which the content client has received all callback messages.

operations: A collection of **operation** Cheetah entities.

Copyright © 2012 Microsoft Corporation.

2.2.21 cht::documentmessages::operation_status_info

The **operation_status_info** Cheetah entity, which contains status information about a set of operations, is used to report the status of submitted item operations to the protocol client. Cheetah entity specification for **operation_status_info**:

```
entity operation_status_info {
    attribute longint first_op_id;
    attribute longint last_op_id;
    attribute operation_state state;
    attribute string subsystem;
    collection error errors;
    collection warning warnings;
};
```

first_op_id: A long integer that contains the operation identifier of the first operation in the sequence of item operations. This value MUST be equal to or greater than 0 as well as less than or equal to the value of the **last_op_id** attribute.

last_op_id: A long integer that contains the operation identifier of the last operation in the sequence of item operations. This value MUST be equal to or greater than 0 as well as equal to or greater than the value of the **first_op_id** attribute.

state: An **operation_state** Cheetah enumeration constant, as specified in cht::documentmessages::operation_state (section 2.2.5), that represents the state of the sequence of item operations.

subsystem: A string that describes where the **operation status info** was generated. This string MUST have a value of either "indexing" or "processing". If the operation status info was produced by either the content distributor or the item processor, the string value will be "processing". If the operation status info was produced by either the indexing dispatcher or an indexing node, the string value will be "indexing".

errors: A collection of **error** Cheetah entities, which are specified in cht::documentmessages::error (section 2.2.6). This value contains the errors for the operations that are specified in the collection of item operations.

warnings: A collection of **warning** Cheetah entities, which are specified in cht::documentmessages::warning (section 2.2.18). This value contains warnings for the operations that are specified in the collection of item operations.

2.2.22 cht::documentmessages::key_value_pair

The **key_value_pair** Cheetah entity is a common superclass that associates a key with a value that can be one of various types. Cheetah entity specification for **key_value_pair**:

```
entity key_value_pair {
    attribute string key;
};
```

key: A string that contains the key.

Copyright © 2012 Microsoft Corporation.

2.2.23 cht::documentmessages::key_value_collection

The **key_value_collection** Cheetah entity forms an association between a single key and a **key_value_pair** collection. Cheetah entity specification for **key_value_collection**:

The **key_value_collection** Cheetah entity is a subclass of the **key_value_pair** Cheetah entity, which is specified in cht::documentmessages::key_value_pair (section 2.2.22). Cheetah entity specification for **key_value_collection**:

```
entity key_value_collection : key_value_pair {
    collection key_value_pair values;
};
```

values: A collection of **key_value_pair** Cheetah entities.

2.2.24 cht::documentmessages::document_id

The **document_id** Cheetah entity uniquely identifies an item by representing the **document identifier (3)** of the item. Cheetah entity specification for **document_id**:

```
entity document_id {
    attribute string id;
    collection key_value_pair routing_attributes;
};
```

id: A string that uniquely identifies the item.

routing_attributes: Unused. The value MUST be an empty Cheetah collection.

2.2.25 cht::documentmessages::string_attribute

The **string_attribute** Cheetah entity forms an association between a key and a string value.

The **string_attribute** Cheetah entity is a subclass of the **key_value_pair** Cheetah entity, which is specified in cht::documentmessages::key_value_pair (section 2.2.22). Cheetah entity specification for **string_attribute**:

```
entity string_attribute : key_value_pair {
    attribute string value;
};
```

value: A string that contains the value.

2.2.26 cht::documentmessages::integer_attribute

The **integer_attribute** Cheetah entity forms an association between a key and an integer value.

The **integer_attribute** Cheetah entity is a subclass of the **key_value_pair** Cheetah entity, which is specified in cht::documentmessages::key_value_pair (section 2.2.22). Cheetah entity specification for **integer_attribute**:

```
entity integer_attribute : key_value_pair {
    attribute integer value;
```


Copyright © 2012 Microsoft Corporation.

```
};
```

value: An integer that contains the value.

2.2.27 cht::documentmessages::bytearray_attribute

The **bytearray_attribute** Cheetah entity forms an association between a key and a value that is contained in a byte array.

The **bytearray_attribute** Cheetah entity is a subclass of the **key_value_pair** Cheetah entity, which is specified in cht::documentmessages::key_value_pair (section 2.2.22). Cheetah entity specification for **bytearray_attribute**:

```
entity bytearray_attribute : key_value_pair {
    attribute bytearray value;
};
```

value: A byte array that contains the value.

2.2.28 cht::documentmessages::document

The **document** Cheetah entity contains information about a single item. Cheetah entity specification for **document**:

```
entity document {
    attribute document_id doc_id;
    collection key_value_pair document_attributes;
};
```

doc_id: A **document_id** Cheetah entity, as specified in cht::documentmessages::document_id (section 2.2.24), that uniquely identifies the item.

document_attributes: A collection of **key_value_pair** Cheetah entities, as specified in cht::documentmessages::key_value_pair (section 2.2.22), that contains the **attributes (1)** of the item.

2.2.29 cht::documentmessages::update_operation

The **update_operation** Cheetah entity either adds a specific item to the index or replaces that item. If an item with the specified document identifier (3) already exists in the **search index**, it is replaced.

The **update_operation** Cheetah entity is a subclass of the **operation** Cheetah entity, which is specified in cht::documentmessages::operation (section 2.2.19).

```
entity update_operation : operation {
    attribute document doc;
};
```

doc: A **document** Cheetah entity, as specified in cht::documentmessages::document (section 2.2.28), that represents the item to add or replace. The **document_attributes** collection of the **document** MUST contain a **string_attribute** Cheetah entity, as specified in

Copyright © 2012 Microsoft Corporation.

cht::documentmessages::string_attribute (section 2.2.25), where the key is "fixml" and the value is a string that contains **FAST Index Markup Language (FIXML)**, as specified in [MS-FSFIXML].

2.2.30 cht::documentmessages::internal_partial_update

The **internal_partial_update** Cheetah entity updates a specific item in the search index.

The **internal_update_operation** Cheetah entity is a subclass of the **operation** Cheetah entity, which is specified in cht::documentmessages::operation (section 2.2.19). Cheetah entity specification for **internal_partial_update**:

```
entity internal_partial_update : operation {
    attribute document_id doc_id;
    collection internal_partial_update_operation operations;
};
```

doc_id: A **document_id** Cheetah entity, as specified in cht::documentmessages::document_id (section 2.2.24), that uniquely identifies the item.

operations: A collection of attributes (1) to update for the item.

2.2.31 cht::documentmessages::remove_operation

The **remove_operation** Cheetah entity removes a specific item from the index.

The **remove_operation** Cheetah entity is a subclass of the **operation** Cheetah entity, which is specified in cht::documentmessages::operation (section 2.2.19). Cheetah entity specification for **remove_operation**:

```
entity remove_operation : operation {
    attribute document_id doc_id;
};
```

doc_id: A **document_id** Cheetah entity, as specified in cht::documentmessages::document_id (section 2.2.24), that uniquely identifies the item.

2.2.32 cht::documentmessages::failed_operation

The content distributor and the item processor use the **failed_operation** Cheetah entity to notify the indexing nodes that an item operation has failed during **item processing**.

The **failed_operation** Cheetah entity is a subclass of the **operation** Cheetah entity, which is specified in cht::documentmessages::operation (section 2.2.19).

```
entity failed_operation : operation {
    attribute string subsystem;
    attribute operation_state state;
    attribute string operation_type;
    attribute document_id doc_id;
    attribute error err;
};
```

subsystem: A string that MUST have the value "processing".

Copyright © 2012 Microsoft Corporation.

state: An **operation_state** Cheetah entity, as specified in cht::documentmessages::operation_state (section 2.2.5), that contains the state of the operation.

operation_type: A string that contains a description of the type of operation.

doc_id: A **document_id**, as specified in cht::documentmessages::document_id (section 2.2.24), that uniquely identifies the item.

err: An **error** Cheetah entity, as specified in cht::documentmessages::error (section 2.2.6), that contains information about the error that caused the operation to fail.

2.2.33 cht::documentmessages::clear_collection

The **clear_collection** Cheetah entity removes a **content collection** from the search index.

The **clear_collection** Cheetah entity is a subclass of the **operation** Cheetah entity, which is specified in cht::documentmessages::operation (section 2.2.19). Cheetah entity specification for **clear_collection**:

```
entity clear_collection : operation {
};
```

2.2.34 cht::documentmessages::subsystem_id_set

The **subsystem_id_set** Cheetah entity contains a collection of names. Cheetah entity specification for **subsystem_id_set:**

```
entity subsystem_id_set {
    collection string ids;
};
```

ids: A collection that MUST consist of either an empty Cheetah collection or a single element that contains the string "indexing".

2.2.35 cht::documentmessages::internal_partial_update_operation

The **internal_partial_update_operation** Cheetah entity is a superclass for:

- The **remove_nodes** Cheetah entity, which is specified in cht::documentmessages::remove_nodes (section 2.2.36).
- The **insert_xml** Cheetah entity, which is specified in cht::documentmessages::insert_xml (section 2.2.37).
- The **string_replace** Cheetah entity, which is specified in cht::documentmessages::string_replace (section 2.2.38).

Cheetah entity specification for **internal_partial_update_operation**:

```
entity internal_partial_update_operation {
};
```


Copyright © 2012 Microsoft Corporation.

2.2.36 cht::documentmessages::remove_nodes

The **remove_nodes** Cheetah entity removes XML nodes from a FIXML structure, as specified in [MS-FSFIXML].

The **remove_nodes** Cheetah entity is a subclass of the **internal_partial_update_operation** Cheetah entity, which is specified in cht::documentmessages::internal_partial_update (section 2.2.35). Cheetah entity specification for **remove_nodes**:

```
entity remove_nodes : internal_partial_update_operation {
    attribute string node_selection;
};
```

node_selection: A string that contains the **XML Path Language (XPath)** expression of the XML nodes to remove. The XPath expression MUST refer to a valid node in the FIXML structure, as specified in [MS-FSFIXML].

2.2.37 cht::documentmessages::insert_xml

The **insert_xml** Cheetah entity inserts a **string_attribute** which is specified in cht::documentmessages::string_attribute (section 2.2.25), into a FIXML structure, as specified in [MS-FSFIXML].

The **insert_xml** Cheetah entity is a subclass of the **internal_partial_update_operation** Cheetah entity, which is specified in cht::documentmessages::internal_partial_update (section 2.2.35). Cheetah entity specification for **insert_xml**:

```
entity insert_xml : internal_partial_update_operation {
    attribute string_attribute key_value;
};
```

key_value: The **string_attribute** Cheetah entity to insert into the FIXML structure. The **key** attribute of the **string_attribute** Cheetah entity specifies an XPath expression for the XML node where the insertion is to be performed, and the **value** attribute of the **string_attribute** Cheetah entity specifies the **CDATA section** to insert. The XPath expression MUST refer to a valid node in the FIXML structure.

2.2.38 cht::documentmessages::string_replace

The **string_replace** Cheetah entity replaces the values in a FIXML structure.

The **string_replace** Cheetah entity is a subclass of the **internal_partial_update_operation** Cheetah entity, which is specified in cht::documentmessages::internal_partial_update (section 2.2.35).Cheetah entity specification for **string_replace**:

```
entity string_replace : internal_partial_update_operation {
    attribute string_attribute key_value;
};
```

key_value: The **string_attribute** Cheetah entity, as specified in cht::documentmessages::string_attribute (section 2.2.25), to replace in the FIXML structure, as specified in [MS-FSFIXML]. The **key** attribute of the **string_attribute** Cheetah entity specifies an XPath expression for the XML node where the insertion is to be performed, and the **value** attribute

Copyright © 2012 Microsoft Corporation.

of the **string_attribute** Cheetah entity specifies the CDATA section to insert. The XPath expression MUST refer to a valid node in the FIXML structure.

2.2.39 core::unsupported_guarantee_set

The **unsupported_guarantee_set** exception specifies that the protocol server was unable to create or re-create a session. The **unsupported_guarantee_set** exception is specified by the following FSIDL specification:

```
exception unsupported_guarantee_set {
    string what;
};
```

what: A string that explains the cause of the exception.

2.2.40 coreprocessing::timed_out

The **timed_out** exception specifies that the protocol server was unable to find an available item processor before a given timeout occurred. The **timed_out** exception is specified by the following FSIDL specification:

```
exception timed_out {
    long id;
    string message;
};
```

id: A long that contains the identifier for the exception.

message: A string that explains the cause of the exception.

2.2.41 coreprocessing::service_unavailable

The **service_unavailable** exception specifies that the protocol server was unable to perform a method invocation. The **service_unavailable** exception is specified by the following FSIDL specification:

```
exception service_unavailable {
    long id;
    string message;
};
```

id: A long variable that contains the identifier for the exception.

message: A string that explains the cause of the exception.

2.2.42 coreprocessing::format_error

The **format_error** exception specifies that a parameter to a method invocation has an invalid format. The **format_error** exception is specified by the following FSIDL specification:

```
exception format_error {
    long id;
    string message;
```


Copyright © 2012 Microsoft Corporation.

```
};
```

id: A long variable that contains the identifier for the exception.

message: A string that explains the cause of the exception.

2.2.43 coreprocessing::no_resources

The **no_resources** exception specifies that the protocol server did not have any resources available to process a method invocation. The **no_resources** exception is specified by the following FSIDL specification:

```
exception no_resources {
    long id;
    string message;
};
```

id: A long variable that contains the identifier for the exception.

message: A string that explains the cause of the exception.

2.2.44 coreprocessing::unknown_collection_error

The **unknown_collection_error** specifies that a content collection was unknown.

```
exception unknown_collection_error {
};
```

2.3 Directory Service Schema Elements

None.

Copyright © 2012 Microsoft Corporation.

3 Protocol Details

This protocol consists of three interfaces: **coreprocessing::session_factory**, **coreprocessing::session**, and **coreprocessing::operation_callback**. For the **coreprocessing::session_factory** interface, the content distributor acts as the protocol client, and the indexing dispatcher acts as the protocol server. For the **coreprocessing::session** interface, the content distributor and the item processor act as the protocol clients, and the indexing dispatcher acts as the protocol server. For the **coreprocessing::operation_callback** interface, the indexing dispatcher acts as the protocol client, and the content distributor acts as the protocol server.

The content distributor communicates synchronously with the indexing dispatcher, setting up a new session by using the **coreprocessing::session_factory** interface. The indexing dispatcher sends asynchronous status messages about item operations to the content distributor by using the **coreprocessing::operation_callback** interface. The content distributor and the item processor submit item operations to the indexing dispatcher by using the **coreprocessing::session** interface.

The indexing dispatcher MUST implement the **coreprocessing::session_factory** interface, as specified in coreprocessing::session_factory Server Details (section 3.1), and the **coreprocessing::session** interface, as specified in coreprocessing::session Server Details (section 3.3). The content distributor MUST implement the **coreprocessing::operation_callback** interface, as specified in coreprocessing::operation_callback Server Details (section 3.4).

The client side of the **coreprocessing::session_factory** interface is specified in coreprocessing::session_factory Client Details (section 3.2). The client side of the **coreprocessing::operation_callback** interface is specified in coreprocessing::operation_callback Client Details section (3.5). The client side of the **coreprocessing::session** interface is simply a pass-through. That is, no additional timers or other states are needed on the client side of this protocol. Calls made by the higher-layer protocol or application are passed directly to the transport, and the results returned by the transport are passed directly back to the higher-layer protocol or application.

3.1 coreprocessing::session_factory Server Details

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 specified in this document.

The protocol server maintains the following states:

session holder: A container for a set of **coreprocessing::session** server objects, each of which can be referenced by a **session identifier**.

For each **coreprocessing::session** server object in the **session holder** state, the protocol server maintains the following states:

session identifier: An integer that contains the identifier of a **coreprocessing::session** server object.

callback client: A state that contains a **coreprocessing::operation_callback** **client proxy**, enabling a protocol server to send asynchronous callback messages via the **coreprocessing::operation_callback** interface.

Copyright © 2012 Microsoft Corporation.

3.1.2 Timers

None.

3.1.3 Initialization

The protocol server MUST use the **bind** method of the Middleware Protocol, as specified in [MS-FSMW] section 3, to bind to the **server object** in the name server. An **abstract object reference (AOR)**, as specified in [MS-FSMW] section 2, encapsulates the input values for the **bind** method:

name: A string value that MUST be "esp/clusters/webcluster/indexing/dispatcher/sessionfactory".

object_id: A value that is implementation specific— that is, determined by the higher-level application.

host: A string that contains the **host name** of the server object on the protocol server. The value is implementation specific—that is, determined by the higher-level application.

port: The **base port** + 390.

interface_type: A string value that MUST be "coreprocessing::session_factory".

interface_version: A string value that MUST be "5.1".

3.1.4 Message Processing Events and Sequencing Rules

The **coreprocessing::session_factory** interface specifies the methods that are listed in the following table.

Method	Description
create	Creates a **coreprocessing::session** server object on the protocol server and returns a client proxy for the session.
recreate	Re-creates a **coreprocessing::session** server object on the protocol server and returns a client proxy for the session.
close	Closes a **coreprocessing::session** server object on the protocol server.
get_highest_session_id	Returns the session identifier that is numerically the highest.

3.1.4.1 coreprocessing::session_factory::create

The **create** method creates a session server object and returns a client proxy for the session. The method is specified by the following FSIDL specification:

```
coreprocessing::session create(
    in long id,
    in string collection,
    in coreprocessing::operation_callback callback,
    in cht::core::guarantee_set guarantees)
raises(coreprocessing::unknown_collection_error,
       core::unsupported_guarantee_set);
```

id: The identifier of the new **coreprocessing::session** server object. The value MUST be a long that is equal to or greater than 0.

Copyright © 2012 Microsoft Corporation.

collection: A string that contains the name of the content collection for which to create the session.

callback: A client proxy that implements the **coreprocessing::operation_callback** interface, as specified in coreprocessing::operation_callback Server Details (section 3.4). This client proxy is used to receive callback messages from the protocol server.

guarantees: A guarantee set in which the **guarantees** attribute MUST contain either a **cht::core::feeding_priority** object that specifies the priority for this session or an empty collection.

Return value: A **coreprocessing::session** client proxy instantiated with an AOR as specified in Initialization (section 3.3.3).

Exceptions raised:

coreprocessing::unknown_collection_error: This exception is not used in this protocol.

core::unsupported_guarantee_set: Raised if the protocol server is unable to create the session.

When the protocol server receives a **create** method invocation, it MUST create and return a new session client proxy to the protocol client and then activate the new **coreprocessing::session** server object. The protocol server MUST instantiate the returned client proxy with an AOR as specified in Initialization (section 3.3.3).

The protocol server MUST store the **coreprocessing::session** server object in the **session holder** state, with the **session identifier** as the unique key.

The protocol server MUST store the **callback** input value in the **callback client** state that is associated with the newly created **coreprocessing::session** server object. Doing so enables the protocol server to communicate status information asynchronously with the protocol client for items that are received by the **session::process** method. For more information, see coreprocessing::session::process (section 3.3.4.3).

3.1.4.2 coreprocessing::session_factory::recreate

The **recreate** method re-creates a specified session, which MUST have been previously created through an invocation of the **coreprocessing::session_factory::create** method, as specified in coreprocessing::session_factory::create (section 3.1.4.1). The method is specified by the following FSIDL specification:

```
coreprocessing::session recreate(
    in long id,
    in string collection,
    in coreprocessing::operation_callback callback,
    in cht::core::guarantee_set guarantees)
raises (coreprocessing::unknown_collection_error,
        core::unsupported_guarantee_set);
```

id: The session identifier of the **coreprocessing::session** server object. The value MUST be equal to or greater than 0.

collection: A string that contains the name of the content collection for which to re-create the session.

Copyright © 2012 Microsoft Corporation.

callback: A client proxy that implements the **coreprocessing::operation_callback** interface, as specified in coreprocessing::operation_callback Server Details (section 3.4). This client proxy is used to receive callback messages from the protocol server.

guarantees: A guarantee set in which the **guarantees** attribute MUST contain either a **cht::core::feeding_priority** object that specifies the priority for this session or an empty collection.

Return value: A **coreprocessing::session** client proxy instantiated with an AOR as specified in Initialization (section 3.3.3).

Exceptions raised:

coreprocessing::unknown_collection_error: This exception is not used in this protocol.

core::unsupported_guarantee_set: Raised if the protocol server is unable to create the session.

When the protocol server receives a **recreate** method invocation, it MUST verify the **session holder** state. If the **session holder** state contains a **coreprocessing::session** server object with the specified session identifier, the protocol server MUST return a client proxy to the existing **coreprocessing::session** server object. If no session with the specified session identifier exists, the protocol server MUST create and return a new session client proxy to the protocol client and then activate the new **coreprocessing::session** server object by using an AOR as specified in Initialization (section 3.3.3).

The protocol server MUST store the **coreprocessing::session** server object in the **session holder** state, with the session identifier as a unique key.

The protocol server MUST store the **callback** client proxy in the **callback** state.

3.1.4.3 coreprocessing::session_factory::close

The **close** method closes a session on the protocol server. The session MUST have been previously created through an invocation of the **coreprocessing::session_factory::create** method, as specified in coreprocessing::session_factory::create (section 3.1.4.1). The method is specified by the following FSIDL specification:

```
void close(in long id);
```

id: The session identifier of the session to close. The value MUST equal that of a **session identifier** in the **session holder** state.

Return value: None.

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

The protocol server MUST remove the **coreprocessing::session** server object with the specified **session identifier** from the **session holder** state.

3.1.4.4 coreprocessing::session_factory::get_highest_session_id

The **get_highest_session_id** method returns the session identifier with the highest number that is registered with the protocol server. The method is specified by the following FSIDL specification:

Copyright © 2012 Microsoft Corporation.

```
long get_highest_session_id();
```

Return value: The value of the **session identifier** state that has the highest value of all the session instances in the **session holder** state. This value MUST be a long that that is equal to or greater than 0.

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

3.1.5 Timer Events

None.

3.1.6 Other Local Events

None.

3.2 coreprocessing::session_factory Client Details

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

The client side of the **coreprocessing::session_factory** interface MUST use the **resolve** method of the underlying protocol, as specified in [MS-FSMW] section 3, to get the client proxy of the **coreprocessing::session_factory** server object that is bound to the name server. The input values for the **resolve** method are:

name: A string value that MUST be "esp/clusters/webcluster/indexing/dispatcher/sessionfactory".

interface_type: A string value that MUST be "coreprocessing::session_factory".

interface_version: A string value that MUSE be "5.1".

3.2.4 Message Processing Events and Sequencing Rules

The **processing::session_factory** interface specifies the methods that are listed in the following table.

Method	Description
create	Creates a **coreprocessing::session** server object on the protocol server and returns a client proxy for the session.
recreate	Re-creates a **coreprocessing::session** server object on the protocol server and returns a client proxy for the session.
close	Closes a **coreprocessing::session** server object on the protocol server.

Copyright © 2012 Microsoft Corporation.

Method	Description
get_highest_session_id	Returns the session identifier that is numerically the highest.

3.2.4.1 coreprocessing::session_factory::create

The **create** method is specified in coreprocessing::session_factory::create (section 3.1.4.1).

3.2.4.2 coreprocessing::session_factory::recreate

The **recreate** method is specified in coreprocessing::session_factory::recreate (section 3.1.4.2).

3.2.4.3 coreprocessing::session_factory::close

The **close** method is specified in coreprocessing::session_factory::close (section 3.1.4.3).

3.2.4.4 coreprocessing::session_factory::get_highest_session_id

The **get_highest_session_id** method is specified in coreprocessing::session_factory::get_highest_session_id (section 3.1.4.4).

3.2.5 Timer Events

None.

3.2.6 Other Local Events

None.

3.3 coreprocessing::session Server Details

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 specified in this document.

The protocol server maintains the following state:

session identifier: An integer that contains the identifier of a **coreprocessing::session** server object.

3.3.2 Timers

None.

3.3.3 Initialization

The **coreprocessing::session** server object MUST be created by either the **coreprocessing::session_factory::create** method, as specified in coreprocessing::session_factory::create (section 3.1.4.1), or the

Copyright © 2012 Microsoft Corporation.

coreprocessing::session_factory::recreate method, as specified in coreprocessing::session_factory::recreate (section 3.1.4.2).

The protocol server MUST initialize the **coreprocessing::session** server object by using an AOR, as specified in [MS-FSMW] section 3, that contains the following values:

object id: A value that is implementation specific— that is, determined by the higher-level application.

host: A value that is implementation specific—that is, determined by the higher-level application.

port: The base port + 390.

interface_type: A string value that MUST be "coreprocessing::session".

interface_version: A string value that MUST be "5.2".

3.3.4 Message Processing Events and Sequencing Rules

The **coreprocessing::session** interface specifies the methods that are listed in the following table.

Method	Description
get_session_id	Returns the identifier of the **session** interface.
get_system_ids	Returns the system identifiers.
process	Processes a set of item operations.

3.3.4.1 coreprocessing::session::get_session_id

The **get_session_id** method returns the session identifier of the session. The method is specified by the following FSIDL specification:

```
long get_session_id();
```

Return value: A long variable that contains the identifier of the session.

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

When the protocol server receives a **get_session_id** method invocation on the **coreprocessing::session** server object, the protocol server MUST return the value of the **session identifier** state.

3.3.4.2 coreprocessing::session::get_system_ids

The **get_system_ids** method returns a description of the callback messages generated by the indexing nodes. The method is specified by the following FSIDL specification:

```
cht::documentmessages::subsystem_id_set
    coreprocessing::session::get_system_ids();
```


Copyright © 2012 Microsoft Corporation.

Return value: A **cht::documentmessages::subsystem_id_set** Cheetah entity in which the **ids** collection MUST contain one string that contains the value "indexing:1:1". This string means that the indexing nodes generate two callback messages, as specified in coreprocessing::operation_callback::status_changed (section 3.4.4.1).

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

3.3.4.3 coreprocessing::session::process

The **process** method sends a set of item operations to the indexing nodes. The method is specified by the following FSIDL specification:

```
long long process(in cht::documentmessages::operation_set batch,
                  in cht::documentmessages::subsystem_id_set subsystems)
raises (coreprocessing::timed_out,
        coreprocessing::service_unavailable,
        coreprocessing::format_error,
        coreprocessing::no_resources);
```

batch: A sequence of item operations to send to the protocol server.

subsystems: A **cht::documentmessages::subsystem_id_set** object for which the **ids** MUST be an empty collection.

Return value: A **long long** value representing the item operation identifier that has the numerically lowest value among the item operations in the **batch** input value.

Exceptions raised:

coreprocessing::timed_out: This exception is not used in this protocol.

coreprocessing::service_unavailable: This exception is not used in this protocol.

coreprocessing::format_error: The protocol server MUST raise this exception if the submitted sequence of item operations contains content that is not valid.

coreprocessing::no_resources: The protocol server MUST raise this exception if the protocol server is out of resources to handle the submitted item operations.

3.3.5 Timer Events

None.

3.3.6 Other Local Events

If the **coreprocessing::session::process** method raises a system exception, the protocol client, which is the item processor, MUST create a **cht::documentmessages**::**error** Cheetah entity, as specified in cht::documentmessages::error (section 2.2.6), for each item operation in the **batch** input value to the **coreprocessing::session::process** method. (For more details, see [MS-FSMW].) The item processor MUST return the generated **cht::documentmessages::error** Cheetah entities to the content distributor through the **processing::procserver_handler::processed** method, as specified in [MS-FSDPD] section 3.

Copyright © 2012 Microsoft Corporation.

3.4 coreprocessing::operation_callback Server Details

A protocol server hosting the **coreprocessing::operation_callback** server object receives asynchronous callback messages from the indexing nodes.

3.4.1 Abstract Data Model

None.

3.4.2 Timers

None.

3.4.3 Initialization

The protocol server MUST initialize the **coreprocessing::operation_callback** server object by using an AOR, as specified in [MS-FSMW] section 3, that contains the following values:

object id: A value that is implementation specific—that is, determined by the higher-level application.

host: A value that is implementation specific—that is, determined by the higher-level application.

port: The base port + 390.

interface_type: A string value that MUST be "coreprocessing::operation_callback".

interface_version: A string value that MUST be "1.0".

3.4.4 Message Processing Events and Sequencing Rules

The **coreprocessing::operation_callback** interface specifies the method that is listed in the following table.

Method	Description
status_changed	Makes information available about a change in the status of item operations.

3.4.4.1 coreprocessing::operation_callback::status_changed

Makes information available about a change in the status of item operations. The method is specified by the following FSIDL specification:

```
void status_changed(in cht::documentmessages::operation_status_info status);
```

status: A **cht::documentmessages::operation_status_info** Cheetah entity, as specified in cht::documentmessages::operation_status_info (section 2.2.20), containing status information about a range of operations that have been stored to disk or made searchable by indexing nodes, as specified in [MS-FSID] section 3.

Return value: None.

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

Copyright © 2012 Microsoft Corporation.

The indexing dispatcher invokes this method when a set of operations has been either saved to disk or indexed.

When the protocol server receives a **coreprocessing::operation_callback::status_changed** method call on the **callback** server object, it MUST make the information available to the content client, as specified in [MS-FSCF] section 3.

For each **cht::documentmessages::operation_set** Cheetah entity, as specified in cht::documentmessages::operation_set (section 2.2.20) **session** protocol client has submitted by using **coreprocessing::session::process** method, as specified in coreprocessing::session::process (section 3.3.4.3), the indexing dispatcher generates two callback messages: **Secured by indexing** and **Completed by indexing**.

The protocol server receives the **Secured by indexing** callback message when the indexing nodes have saved item operations to disk. The **status** input value to the **coreprocessing::session::status_changed** method will contain the following attributes:

first_op_id: The item operation identifier that is numerically the lowest in the **cht::documentmessages::operation_set** Cheetah entity that was submitted through the **coreprocessing::session::process** method.

last_op_id: The item operation identifier that is numerically the highest in the **cht::documentmessages::operation_set** Cheetah entity that was submitted through the **coreprocessing::session::process** method.

state: A value that MUST be the Cheetah enumeration value **cht::documentmessages::secured**, as specified in cht::documentmessages::operation_state (section 2.2.5).

subsystem: A string that MUST have the value "indexing".

errors: The errors provided by the indexing nodes for item operations. The **operation_id** attribute of the **cht::documentmessages::error** Cheetah entity identifies the item operation that an error refers to.

warnings: The warnings provided by the indexing nodes for item operations. The **operation_id** attribute of the **cht::documentmessages::warning** Cheetah entity identifies the item operation that a warning refers to.

The protocol server receives the **Completed by indexing** callback message when the indexing nodes have processed the item operations and the actions that were triggered by the item operations are visible in the search index. The **status** input value to the **coreprocessing::session::status_changed** method will contain the following attributes:

first_op_id: The item operation identifier that is numerically the lowest in the **cht::documentmessages::operation_set** Cheetah entity that was submitted through the **coreprocessing::session::process** method**.**

last_op_id: The item operation identifier that is numerically the highest in the **cht::documentmessages::operation_set** Cheetah entity that was submitted through the **coreprocessing::session::process** method**.**

state: A value that MUST be the Cheetah enumeration value **cht::documentmessages::completed**, as specified in cht::documentmessages::operation_state (section 2.2.5).

subsystem: A string that MUST have the value "indexing".

Copyright © 2012 Microsoft Corporation.

errors: The errors provided by the indexing nodes for item operations. The **operation_id** attribute of the **cht::documentmessages::error** Cheetah entity identifies the item operation that an error refers to.

warnings: The warnings provided by the indexing nodes for item operations. The **operation_id** attribute of the **cht::documentmessages::warning** Cheetah entity identifies the item operation that a warning refers to.

3.4.5 Timer Events

None.

3.4.6 Other Local Events

None.

3.5 coreprocessing::operation_callback Client Details

The indexing nodes send asynchronous callback messages to the content distributor about operations that are received in a **coreprocessing::session::process**.

3.5.1 Abstract Data Model

None.

3.5.2 Timers

None.

3.5.3 Initialization

The protocol client that uses the **coreprocessing::operation_callback** interface MUST use the **callback** client proxy that is sent as both an input value to the **coreprocessing::session_factory::create** method, as specified in coreprocessing::session_factory::create (section 3.1.4.1), and as an input value to the **coreprocessing::session_factory::recreate** method, as specified in coreprocessing::session_factory::recreate (section 3.1.4.2).

3.5.4 Message Processing Events and Sequencing Rules

The **coreprocessing::operation_callback** interface specifies the method that is listed in the following table.

Method	Description
status_changed	Makes information available about a change in the status of item operations.

3.5.4.1 coreprocessing::operation_callback::status_changed

The protocol client MUST call the **coreprocessing::operation_callback::status_changed** method when all the item operations received in a **coreprocessing::session::process** method invocation, as specified in coreprocessing::session::process (section 3.3.4.3), have been saved to disk and when the actions that were triggered by the item operations are visible in the search index, as specified in [MS-FSID] section 3.

Copyright © 2012 Microsoft Corporation.

3.5.5 Timer Events

None.

3.5.6 Other Local Events

If the **status_changed** method in the **coreprocessing::operation_callback** interface raises a system exception, the protocol client MUST perform the following steps:

1. Find the **coreprocessing::session** server object that corresponds to the **coreprocessing::operation_callback** client proxy in the **session holder** state.
2. Deactivate the **coreprocessing::session** server object.
3. Remove the **coreprocessing::session** server object from the **session holder** state.

For more details about system exception, see [MS-FSMW].

Copyright © 2012 Microsoft Corporation.

4 Protocol Examples

4.1 Processing Item Operations

This example describes how to create and set up a session, feed 10 item operations, receive callback messages about the status of the item operations, and then close the session.

Initializing the session

The **coreprocessing::session_factory** protocol server first creates a server object that implements the **coreprocessing::session_factory** interface and then registers that object in the name server. The **coreprocessing::session_factory** protocol client acquires a client proxy for this **coreprocessing::session_factory** interface by resolving the server object in the name server. Doing so is possible because the protocol client and protocol server have already agreed on the location of the shared name server and the symbolic name of the server object.

Setting up the session

The **coreprocessing::session_factory** protocol client creates and activates a **coreprocessing::operation_callback** server object and then uses the **create** method to send a client proxy for this server object to the **coreprocessing::session_factory** protocol server.

The **coreprocessing::session_factory** protocol server receives the **create** method invocation and then creates, activates, and returns a **coreprocessing::session** client proxy. The **coreprocessing::session_factory** protocol server then stores the received session identifier in the **session identifier** state, stores the created **coreprocessing::session** server object in the **session holder** state, and stores the received **coreprocessing::operation_callback** client proxy in the **callback client** state.

The **coreprocessing::session_factory** protocol client stores the returned **coreprocessing::session** client proxy in the **session client holder** state.

Using the session

The **session** protocol client uses the **coreprocessing::session** client proxy (which was returned from the **coreprocessing::session_factory::create** method) to call the **process** method to send the item operations to the **coreprocessing::session** protocol server.

Sending callback messages

The **coreprocessing::session** protocol server receives the **process** method invocation and processes the item operations that are included as an input value. When the operations have been stored to disk, the **coreprocessing::session** protocol server looks up the **coreprocessing::operation_callback** client proxy from the **callback client** state and then invokes the **coreprocessing::operation_callback::status_changed** method in the **coreprocessing::operation_callback** protocol server.

The **coreprocessing::session** protocol server also invokes the **coreprocessing::operation_callback::status_changed** method in the **coreprocessing::operation_callback** protocol server when the item operations have been processed and the actions triggered by the item operations are visible in the search index.

Receiving callback messages

The **coreprocessing::operation_callback** protocol server receives the **status_changed** method invocations and makes them visible to the content client, as described in [MS-FSCF] section 3.

Copyright © 2012 Microsoft Corporation.

Closing the session

The **coreprocessing::session_factory** protocol client closes the session by invoking the **close** method on the **coreprocessing::session_factory** protocol server.

4.1.1 Code: Initializing the session_factory Protocol Server

Note: This code includes additional line breaks to facilitate online and printed viewing. Remove the line breaks before running the code.

```
SET session_factory_server_object_instance TO INSTANCE OF coreprocessing::session_factory
SERVER OBJECT

SET session_factory_server_object_host TO "myserver"

SET session_factory_server_object_port TO 13328

SET session_factory_server_object_interface_type TO "coreprocessing::session_factory"

SET session_factory_server_object_interface_version TO "5.1"

SET session_factory_server_object_name TO
"esp/clusters/webcluster/indexing/dispatcher/sessionfactory"

SET session_factory_server_object_aor TO
session_factory_server_object_host,
session_factory_server_object_port, session_factory_server_object_interface_type,
session_factory_server_object_interface_version AND session_factory_server_object_name

CALL nameserver.bind WITH
session_factory_server_object_instance
AND session_factory_server_object_aor
```

4.1.2 Code: Initializing the session_factory Protocol Client

```
SET session_factory_server_object_name TO
"esp/clusters/webcluster/indexing/dispatcher/sessionfactory"

SET session_factory_server_object_type TO
"coreprocessing::session_factory"

SET session_factory_server_object_version TO "5.1"

CALL nameserver.resolve WITH
session_factory_server_object_name,
session_factory_server_object_type AND
session_factory_server_object_version
RETURNING session_factory_client_proxy
```

4.1.3 Code: Sending a Message from the session_factory Protocol Client

```
SET session_id TO "1"

SET guarantees to cht::core::guarantee_set

SET collection TO "mycollection"
```


Copyright © 2012 Microsoft Corporation.

```
SET callback_server_object_instance
TO INSTANCE OF operation_callback SERVER OBJECT

CALL session_factory_client_proxy.create WITH
session_id AND collection
AND callback_server_object_instance
AND guarantees RETURNING session_client_proxy

ADD session_client_proxy TO session_client_holder_state
```

4.1.4 Code: Sending a Response from the session_factory Protocol Server

```
SET session_id_state TO session_id

SET callback_client_state TO callback_server_object_instance

SET session_server_object_instance TO INSTANCE OF
coreprocessing::session SERVER OBJECT

SET session_server_state TO session_server_object_instance

RETURN session_server_object_instance
```

4.1.5 Code: Initializing the session Protocol Client

```
GET session_client_proxy FROM
session_factory_client_proxy.create RETURN value
```

4.1.6 Code: Invoking the process Method from the session Protocol Client

```
SET last_operation_in_sequence TO "9"

SET operations TO OPERATION_SET_OBJECT_WITH_10_OPERATIONS

SET operations.completed_op_id TO 0

SET subsystem_id_set TO subystem_id_set_object

SET subysystem_id_set.ids to EMPTY COLLECTION

CALL session_client_proxy.process WITH operations
AND subystem_id_set RETURNING first_operation_id
```

4.1.7 Code: Sending a Response from the session Protocol Server

```
PROCESS ALL operations
SET callback_client_proxy TO callback_client_state

REPEAT
  IF operation RANGE IN operations IS PERSISTED TO DISK,
    THEN CALL callback_client_proxy.status_changed WITH
        OPERATION_STATUS_INFO FOR RANGE
```


Copyright © 2012 Microsoft Corporation.

```
  IF operation RANGE IN operations IS SEARCHABLE,
    THEN CALL callback_client_proxy.status_changed WITH
        OPERATION_STATUS_INFO FOR RANGE
UNTIL complete callback HAS BEEN SENT FOR ALL operations
```

4.1.8 Code: Sending a Response from the session Protocol Client

```
REPEAT
RECEIVE CALLBACKS and RETURN to content client
```

4.1.9 Code: Closing the session_factory Protocol Client

```
CALL session_factory_client_proxy.close WITH session_id

DEACTIVATE callback_server_object_instance
```

4.1.10 Code: Closing the session_factory Protocol Server

```
GET session_server_object_instance FROM session_server_state
FOR session_id

REMOVE session_server_object_instance FROM
session_server_state

DEACTIVATE session_server_object_instance
```


Copyright © 2012 Microsoft Corporation.

5 Security

5.1 Security Considerations for Implementers

This protocol provides neither endpoint authentication nor data encryption for the communications between the protocol client and the protocol server.

5.2 Index of Security Parameters

None.

Copyright © 2012 Microsoft Corporation.

6 Appendix A: Full FSIDL

For ease of implementation, the full FSIDL is provided in the following code:

```
module interfaces {
  module core {

    exception unsupported_guarantee_set {
      string message;
    };
  };

  module coreprocessing {

    exception format_error {
      long id;
      string message;
    };
    exception no_resources {
      long id;
      string message;
    };

    exception service_unavailable {
      long id;
      string message;
    };

    exception timed_out {
      long id;
      string message;
    };

    exception unknown_collection_error {
    };

    interface operation_callback {
      #pragma version operation_callback 1.0

      void status_changed(
        in cht::documentmessages::operation_status_info status);

    };

    interface session {
      #pragma version session 5.2

      long get_session_id();

      long long
      process(in cht::documentmessages::operation_set batch,
              in cht::documentmessages::subsystem_id_set subsystem_ids)
        raises (timed_out, service_unavailable, format_error, no_resources);

      cht::documentmessages::subsystem_id_set get_system_ids();

    };
```


Copyright © 2012 Microsoft Corporation.

```
    interface session_factory {
       #pragma version session_factory 5.1

       coreprocessing::session create(
         in long id,
         in string collection,
         in coreprocessing::operation_callback callback,
         in cht::core::guarantee_set guarantees)
         raises (unknown_collection_error, core::unsupported_guarantee_set);

       coreprocessing::session recreate(
         in long id,
         in string collection,
         in coreprocessing::operation_callback callback,
         in cht::core::guarantee_set guarantees)
         raises (unknown_collection_error, core::unsupported_guarantee_set);

       void close(in long id);

       long get_highest_session_id();

     };
   };
 };
```


Copyright © 2012 Microsoft Corporation.

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.

Copyright © 2012 Microsoft Corporation.

8 Change Tracking

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

Copyright © 2012 Microsoft Corporation.

9 Index

Copyright © 2012 Microsoft Corporation.

E

F

G

I

Copyright © 2012 Microsoft Corporation.

Copyright © 2012 Microsoft Corporation.

T

V

Copyright © 2012 Microsoft Corporation.