

# [MS-FSIFT]: Indexer Fault Tolerance Protocol Specification

---

## Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

## Revision Summary

Date	Revision History	Revision Class	Comments
02/19/2010	1.0	Major	Initial Availability
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.05	Minor	Clarified the meaning of the technical content.
03/18/2011	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	1.05	No change	No changes to the meaning, language, or formatting of the technical content.

# Table of Contents

<b>1</b>	<b>Introduction .....</b>	<b>6</b>
1.1	Glossary .....	6
1.2	References.....	6
1.2.1	Normative References.....	6
1.2.2	Informative References .....	7
1.3	Protocol Overview (Synopsis) .....	7
1.3.1	Column Backup.....	8
1.3.2	Column Master.....	8
1.3.3	Content Operation Sequence Store .....	9
1.3.4	Sequence Receptor.....	9
1.4	Relationship to Other Protocols.....	9
1.5	Prerequisites/Preconditions .....	10
1.6	Applicability Statement.....	10
1.7	Versioning and Capability Negotiation.....	10
1.8	Vendor-Extensible Fields.....	10
1.9	Standards Assignments .....	10
<b>2</b>	<b>Messages.....</b>	<b>11</b>
2.1	Transport.....	11
2.2	Common Data Types .....	11
2.2.1	sequence_log_info.....	11
2.2.2	sequence_operation.....	12
2.2.3	empty_operation.....	12
2.2.4	fixml_invalidation .....	12
2.2.5	remdoclist .....	13
2.2.6	exclusionlist.....	13
2.2.7	remove_collection .....	13
2.2.8	fixml_append.....	14
2.2.9	document_error .....	14
2.2.10	content_operation_sequence .....	15
2.2.11	cheetah.....	16
<b>3</b>	<b>Protocol Details.....</b>	<b>17</b>
3.1	column_master and content_operation_sequence_store Server Details .....	17
3.1.1	Abstract Data Model .....	17
3.1.2	Timers .....	17
3.1.3	Initialization .....	18
3.1.4	Message Processing Events and Sequencing Rules.....	20
3.1.4.1	column_master::get_row_id .....	21
3.1.4.2	column_master::register_backup_node.....	21
3.1.4.3	column_master::has_backup_node.....	22
3.1.4.4	column_master::check_backup_nodes .....	22
3.1.4.5	column_master::abdicate .....	22
3.1.4.6	column_master::connect_receiver .....	22
3.1.4.7	column_master::disconnect_receiver .....	23
3.1.4.8	content_operation_sequence_store::is_master.....	23
3.1.4.9	content_operation_sequence_store::get_stored_sequences .....	23
3.1.4.10	content_operation_sequence_store::has_sequence_id .....	23
3.1.4.11	content_operation_sequence_store::request_sequences.....	24
3.1.4.12	content_operation_sequence_store::get_row_id.....	24

3.1.4.13	content_operation_sequence_store::get_hostname .....	24
3.1.4.14	content_operation_sequence_store::get_highest_sequence_id .....	24
3.1.4.15	content_operation_sequence_store::get_lowest_sequence_id .....	24
3.1.5	Timer Events .....	25
3.1.6	Other Local Events .....	25
3.2	column_master and content_operation_sequence_store Client Details .....	25
3.2.1	Abstract Data Model .....	25
3.2.2	Timers .....	25
3.2.3	Initialization .....	25
3.2.4	Message Processing Events and Sequencing Rules .....	26
3.2.4.1	column_master::get_row_id .....	27
3.2.4.2	column_master::register_backup_node .....	27
3.2.4.3	column_master::has_backup_node .....	27
3.2.4.4	column_master::check_backup_nodes .....	27
3.2.4.5	column_master::abdicate .....	27
3.2.4.6	column_master::connect_receiver .....	27
3.2.4.7	column_master::disconnect_receiver .....	27
3.2.4.8	content_operation_sequence_store::is_master .....	27
3.2.4.9	content_operation_sequence_store::get_stored_sequences .....	28
3.2.4.10	content_operation_sequence_store::has_sequence_id .....	28
3.2.4.11	content_operation_sequence_store::request_sequences .....	28
3.2.4.12	content_operation_sequence_store::get_row_id .....	28
3.2.4.13	content_operation_sequence_store::get_hostname .....	28
3.2.4.14	content_operation_sequence_store::get_highest_sequence_id .....	28
3.2.4.15	content_operation_sequence_store::get_lowest_sequence_id .....	28
3.2.5	Timer Events .....	28
3.2.6	Other Local Events .....	28
3.3	column_backup and sequence_receptor Server Details .....	28
3.3.1	Abstract Data Model .....	28
3.3.2	Timers .....	29
3.3.3	Initialization .....	29
3.3.4	Message Processing Events and Sequencing Rules .....	29
3.3.4.1	column_backup::get_row_id .....	30
3.3.4.2	column_backup::get_hostname .....	30
3.3.4.3	column_backup::submit_sequence .....	30
3.3.4.4	column_backup::commit_sequence .....	30
3.3.4.5	column_backup::abort_sequence .....	30
3.3.4.6	column_backup::activate_index_set .....	31
3.3.4.7	sequence_receptor::submit_sequence .....	31
3.3.4.8	sequence_receptor::finished .....	31
3.3.4.9	sequence_receptor::get_hostname .....	31
3.3.5	Timer Events .....	31
3.3.6	Other Local Events .....	32
3.4	column_backup and sequence_receptor Client Details .....	32
3.4.1	Abstract Data Model .....	32
3.4.2	Timers .....	32
3.4.3	Initialization .....	32
3.4.4	Message Processing Events and Sequencing Rules .....	32
3.4.4.1	column_backup::get_row_id .....	32
3.4.4.2	column:backup::get_hostname .....	32
3.4.4.3	column_backup::submit_sequence .....	33
3.4.4.4	column_backup::commit_sequence .....	33
3.4.4.5	column_backup::abort_sequence .....	33

3.4.4.6	column_backup::activate_index_set .....	33
3.4.4.7	sequence_receptor::submit_sequence.....	33
3.4.4.8	sequence_receptor::finished .....	33
3.4.4.9	sequence_receptor::get_hostname .....	33
3.4.5	Timer Events .....	33
3.4.6	Other Local Events .....	33
<b>4</b>	<b>Protocol Examples.....</b>	<b>34</b>
4.1	Recover a Backup Indexing Node.....	34
4.1.1	Recovery Code.....	34
4.1.2	Recovery Sequence Diagram .....	35
<b>5</b>	<b>Security.....</b>	<b>36</b>
5.1	Security Considerations for Implementers.....	36
5.2	Index of Security Parameters .....	36
<b>6</b>	<b>Appendix A: Full FSIDL.....</b>	<b>37</b>
6.1	FSIDL.....	37
6.2	Cheetah.....	38
<b>7</b>	<b>Appendix B: Product Behavior .....</b>	<b>40</b>
<b>8</b>	<b>Change Tracking.....</b>	<b>41</b>
<b>9</b>	<b>Index .....</b>	<b>42</b>

# 1 Introduction

This document specifies the Indexer Fault Tolerance Protocol, which is used for distributing and synchronizing data structures between indexing nodes, through which functionality the index column achieves fault-tolerant behavior.

## 1.1 Glossary

The following terms are defined in [\[MS-GLOS\]](#):

**fault-tolerant**  
**fully qualified domain name (FQDN)**  
**unmarshal**

The following terms are defined in [\[MS-OFCGLOS\]](#):

**abstract object reference (AOR)**  
**base port**  
**Boolean**  
**Cheetah**  
**Cheetah checksum**  
**Cheetah entity**  
**client proxy**  
**content collection**  
**document identifier**  
**exclusion list**  
**FAST Index Markup Language (FIXML)**  
**FAST Search Interface Definition Language (FSIDL)**  
**index column**  
**index partition**  
**indexer row**  
**indexing node**  
**inverted index**  
**item**  
**master indexer node**  
**name server**  
**query matching node**

The following terms are specific to this document:

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

## 1.2 References

### 1.2.1 Normative References

We conduct frequent surveys of the normative references to assure their continued availability. If you have any issue with finding a normative reference, please contact [dochelp@microsoft.com](mailto: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](#)"

[MS-FSID] Microsoft Corporation, "[Indexing Distribution Protocol Specification](#)"

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

[MS-FSRFCO] Microsoft Corporation, "[Remote File Copy Orchestration 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-FSFXML] Microsoft Corporation, "[FXML Data Structure](#)"

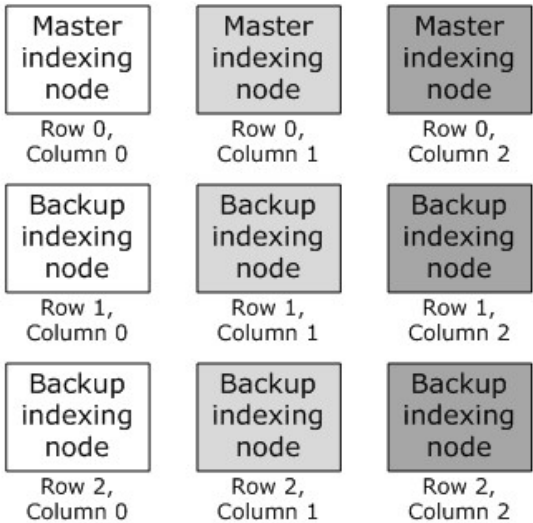
[MS-GLOS] Microsoft Corporation, "[Windows Protocols Master Glossary](#)".

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

### 1.3 Protocol Overview (Synopsis)

On the highest level, the index is partitioned across several **index columns**. On the level of each **indexing node**, the index is partitioned into a disjointed set of **index partitions**. These index partitions are denoted by integers from 0 to n-1, where n is the number of index partitions on the indexing node. A full set of disjoint index partitions is called an index set, and the set of index partitions currently used by **query matching nodes** to facilitate search queries is called the active index set.

In a **fault-tolerant** system setup there are several indexing nodes in the same index column, with indexing nodes assuming different column roles. In every index column one of the indexing nodes assumes the role of the **master indexer node**, while the rest are referred to as backup indexing nodes. The different indexing nodes are identified by their **indexer row** identifier, as shown in the following figure.



**Figure 1: Matrix of indexing nodes in a fault-tolerant topology**

The backup indexing nodes have all of the **item** data needed to build indexes, but it is only the master indexer node that performs the actual indexing.

Item operations, such as adding new items or removing old items, result from Middleware calls, as described in [\[MS-FSMW\]](#). Supported item operations include the **update\_operation**, which adds a new item to the **inverted index**, and the **remove\_operation**, which removes an item from the inverted index. All supported item operations are described in [\[MS-FSID\]](#) section 2.2.

The high-level item operations are converted by the master indexer node into low-level **sequence operations**, as described in section [2.2.2](#). The set of indexable items are stored in the intermediate **FAST Index Markup Language (FIXML)** format, as described in [\[MS-FSFIXML\]](#), and the sequence operations manipulate both the FIXML files and the inverted index structures. FIXML files are identified by using a **file identifier**, an increasing integer sequence number. A single FIXML file potentially contains several items.

The master indexer node also keeps a fault-tolerant storage, a data structure that contains a backlog of processed sequence operations. The sequence operations are numbered with an increasing integer sequence identifier. The fault-tolerant storage enables a restarted backup indexing node to quickly become fully synchronized, as the master indexer node only has to resend the sequence operations that were not delivered during the backup indexing node's downtime.

Using the protocol described in this document, the backup indexing nodes register a file receiver with the master indexer node. The file receiver is a server object implementing the `file_receiver` interface, as described in [\[MS-FSRFCO\]](#). The file receiver interface is used for transferring the inverted index from the master indexer node to the backup indexing nodes.

The Indexer Fault Tolerance Protocol uses four interrelated **FAST Search Interface Definition Language (FSIDL)** interfaces, with both the master indexer node and the backup indexing nodes taking on the role of protocol client in some transactions, and protocol server in others. The interfaces are described in the following sections.

### 1.3.1 Column Backup

The Column Backup interface is used by the master indexer node in its communication with the backup indexing nodes. The provided functionality includes the following:

- Retrieving the indexer row identifier of the backup indexing node.
- Retrieving the **fully qualified domain name (FQDN)** of the backup indexing node.
- Submitting a set of sequence operations to the backup indexing node.
- Forcing the backup indexing node to re-resolve the master indexer node.

### 1.3.2 Column Master

The Column Master interface is used by the backup indexing nodes in their communication with the master indexer node. The provided functionality includes the following:

- Retrieving the indexer row identifier of the master indexer node.
- Registering the existence of a backup indexing node.
- Unregistering unresponsive backup indexing nodes.
- Forcing the master indexer node to relinquish status as master indexer node.
- Registering a subscribing file receiver.

### 1.3.3 Content Operation Sequence Store

The Content Operation Sequence Store interface is used by the backup indexing nodes to request missing item data from other indexing nodes in the index column. The provided functionality includes the following:

- Retrieving the indexer row identifier of the remote indexing node.
- Confirming whether the remote indexing node is the master indexer node.
- Querying whether the remote indexing node has a specific sequence operation available.
- Querying the range of sequence operations available from the remote indexing node.
- Requesting an asynchronous transfer of sequence operations.
- Retrieving the fully qualified domain name (FQDN) of the remote indexing node.
- Retrieving the sequence identifier of the newest sequence operation stored in the remote indexing node's fault-tolerant storage.
- Retrieving the sequence identifier of the oldest sequence operation stored in the remote indexing node's fault-tolerant storage.

### 1.3.4 Sequence Receptor

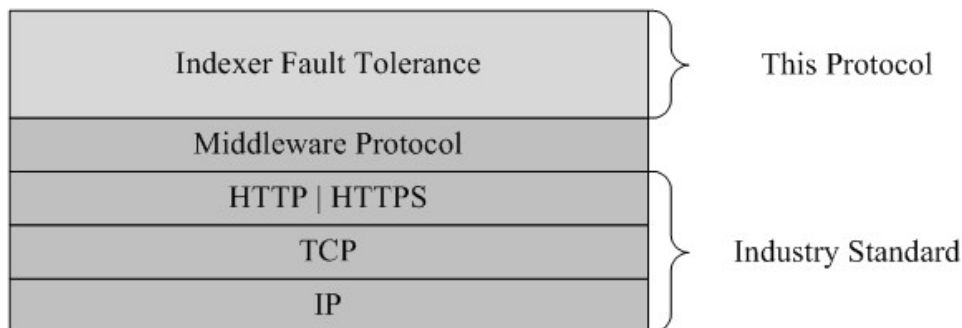
The Sequence Receptor interface is used by indexing nodes to respond to requests issued using the Content Operation Sequence Store interface. The provided functionality includes the following:

- Retrieving the fully qualified domain name (FQDN) of the remote indexing node.
- Submitting a set of sequence operations as a response to requests issued using the Content Operation Sequence Store interface.

## 1.4 Relationship to Other Protocols

This 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 used by this protocol:



**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**

This protocol is applicable where there is a need for distributing and synchronizing data structures between indexing nodes.

## **1.7 Versioning and Capability Negotiation**

Capability Negotiation: The Middleware Protocol is connectionless, but the correct interface version is to be specified in every message passed using the Middleware Protocol. See sections [3.1.3](#) and [3.3.3](#) for the specific version numbers.

## **1.8 Vendor-Extensible Fields**

None.

## **1.9 Standards Assignments**

None.

## 2 Messages

### 2.1 Transport

The messages supported by the interfaces specified in sections [3.1.4](#) and [3.3.4](#) MUST be sent as HTTP POST messages, as specified in [\[MS-FSMW\]](#).

### 2.2 Common Data Types

The allowed FSIDL data types are specified in [\[MS-FSMW\]](#). This protocol also uses custom Cheetah data types that are marshaled and embedded in a generic collection of bytes.

**Cheetah entities** MUST be encoded as specified in [\[MS-FSCHT\]](#) section 2.2. The **Cheetah checksum** of the Cheetah messages MUST be -2127454238. The Cheetah type identifier for the Cheetah entities MUST be as specified in the following table.

Cheetah entity	Cheetah type identifier
<b>sequence_log_info</b>	5
<b>sequence_operation</b>	6
<b>empty_operation</b>	7
<b>fixml_invalidation</b>	8
<b>Remdoclist</b>	9
<b>Exclusionlist</b>	10
<b>remove_collection</b>	11
<b>fixml_append</b>	12
<b>document_error</b>	15
<b>content_operation_sequence</b>	16

In addition to the preceding Cheetah entities, the protocol also makes use of data types that are aliases for standard FSIDL data types. The aliased data types are not custom data types, but rather standard FSIDL data types that have been given more convenient or verbose names to increase code readability.

#### 2.2.1 sequence\_log\_info

The **sequence\_log\_info** data type is used to encapsulate information regarding the content of the fault-tolerant storage. The structure of this data type, as specified in section [6.2](#), is as follows:

```
entity sequence_log_info {
    attribute longint low_sequence_id;
    attribute longint high_sequence_id;
    attribute longint processed_sequence_id;
};
```

**low\_sequence\_id:** An integer value containing the sequence identifier of the oldest sequence operation stored in the fault-tolerant storage. The value MUST be greater than or equal to zero, and the value MUST be less than or equal to **high\_sequence\_id**.

**high\_sequence\_id:** An integer value containing the sequence identifier of the newest sequence operation stored in the fault-tolerant storage. The value MUST be greater than or equal to zero, and the value MUST be greater than or equal to **low\_sequence\_id**.

**processed\_sequence\_id:** An integer value containing the sequence identifier of the newest operation the indexing node has executed. The value MUST be greater than or equal to zero, and the value MUST be less than or equal to **high\_sequence\_id**.

### 2.2.2 sequence\_operation

The **sequence\_operation** data type is the base type for operations manipulating the set of indexed items. The structure of this data type, as specified in section [6.2](#), is as follows:

```
entity sequence_operation {
    attribute longint sequence_number;
    attribute longint operation_id;
};
```

**sequence\_number:** An integer value containing the sequence identifier of the sequence operation. The value MUST be greater than or equal to zero.

**operation\_id:** The identifier of the item operation from which the sequence operation originated.

### 2.2.3 empty\_operation

The **empty\_operation** data type is a subtype of **sequence\_operation** specified in section [2.2.2](#). The **empty\_operation** MUST NOT have any effect. The structure of this data type, as specified in section [6.2](#), is as follows:

```
entity empty_operation : sequence_operation {
};
```

### 2.2.4 fixml\_invalidiation

The **fixml\_invalidiation** data type is a sub-type of **sequence\_operation** specified in section [2.2.2](#). The **fixml\_invalidiation** data type is used to invalidate an item. The structure of this data type, as specified in section [6.2](#), is as follows:

```
entity fixml_invalidiation : sequence_operation {
    attribute string document_id;
    attribute int file_id;
    attribute int magic_idx;
    attribute bool is_update;
};
```

**document\_id:** The **document identifier (3)**.

**file\_id:** The file identifier of the FIXML file containing the item to be removed.

**magic\_idx:** An integer value from 0 to n-1, where n is the number of items in the FIXML file, describing the position of the item in the FIXML file.

**is\_update:** MUST be **true** if the sequence operation was derived from a high-level **update\_operation** as specified in [\[MS-FSID\]](#) section 2.2.36; otherwise **false**.

### 2.2.5 remdoclist

The **remdoclist** data type is a subtype of **sequence\_operation** specified in section [2.2.2](#). The **remdoclist** data type is used to add an item to the **remove list**, a list of items that are to be removed from the index the next time it is re-indexed. Instead of removing the item from the index directly, it is removed from query results by being added to the query matching nodes' **exclusion lists**. The structure of this data type, as specified in section [6.2](#), is as follows:

```
entity remdoclist : sequence_operation {
    attribute string  document_id;
    attribute int     old_file_id;
    attribute int     new_file_id;
};
```

**document\_id:** The document identifier (3) of the item.

**old\_file\_id:** The file identifier of the FIXML file containing the item to remove.

**new\_file\_id:** If the sequence operation was derived from a high-level **update\_operation**, as specified in [\[MS-FSID\]](#) section 2.2.36, the value MUST be the file identifier of the FIXML file containing the updated item. If the sequence operation was not derived from a high-level **update\_operation**, the value MUST be the same as **old\_file\_id**.

### 2.2.6 exclusionlist

The **exclusionlist** data type is a subtype of **sequence\_operation** specified in section [2.2.2](#). The **exclusionlist** data type is used to add an item to the exclusion list. An item that has been added to the exclusion list will be removed from query results, even if it is present in the index. The structure of this data type, as specified in section [6.2](#), is as follows:

```
entity exclusionlist : sequence_operation {
    attribute string  document_id;
    attribute int     old_file_id;
};
```

**document\_id:** The document identifier (3) of the item.

**old\_file\_id:** The file identifier of the FIXML file containing the item to exclusion list.

### 2.2.7 remove\_collection

The **remove\_collection** data type is a subtype of **sequence\_operation** specified in section [2.2.2](#). The **remove\_collection** data type is used to remove a **content collection** and all of its indexed items. The content collection concerned is implicitly derived from the higher level session set up before item operations are fed, as specified in [\[MS-FSID\]](#) section 3.1.4.1. The structure of this data type, as specified in section [6.2](#), is as follows:

```
entity remove_collection : sequence_operation {
```

```
};
```

### 2.2.8 fixml\_append

The **fixml\_append** data type is a subtype of **sequence\_operation** specified in section [2.2.2](#). The **fixml\_append** data type is used to add an item to a FIXML file. The structure of this data type, as specified in section [6.2](#), is as follows:

```
entity fixml_append : sequence_operation {  
    attribute string document_id;  
    attribute string document_content;  
    attribute int file_id;  
    attribute int magic_idx;  
    attribute bool is_update;  
};
```

**document\_id:** The document identifier (3) of the item.

**document\_content:** The content of the item to append.

**file\_id:** The file identifier of the FIXML file in which to place the item.

**magic\_idx:** An integer describing the position of the item in the FIXML file. This MUST be between 0 and n-1, where n is the number of items in the FIXML file.

**is\_update:** A **Boolean** value that MUST be **true**.

### 2.2.9 document\_error

The **document\_error** data type is a subtype of **sequence\_operation** specified in section [2.2.2](#). The **document\_error** data type contains errors relating to an item. The structure of this data type, as specified in section [6.2](#), is as follows:

```
entity document_error : sequence_operation {  
    attribute string document_id;  
    attribute int error_code;  
    attribute int action;  
    attribute string subsystem;  
    attribute string error_message;  
}
```

**document\_id:** The document identifier (3) of the item.

**error\_code:** An integer value containing an error code. The value MUST be one of the values described in the following table.

Error code	Description
1	Missing attribute.
2	Generic error.
3	Unknown item.

Error code	Description
4	Indexer suspended.
5	FIXML write error. Failed to persist item operation.
6	Unknown content collection.
7	Partial update error.

**action:** An integer value containing a suggested action code. The value **MUST** be one of the values described in the following table.

Action code	Description
1	Resubmit.
2	Limited resubmit.
3	Drop operation.
4	Terminate.

**subsystem:** A string value that **MUST** be "indexing".

**error\_message:** A string that contains the error message.

## 2.2.10 content\_operation\_sequence

The **content\_operation\_sequence** data type is a collection of sequence operations. The structure of this data type, as specified in section [6.2](#), is as follows:

```
entity content_operation_sequence {
    attribute int session_id;
    attribute string document_collection;
    attribute longint low_sequence_id;
    attribute longint high_sequence_id;
    collection sequence_operation operations;
};
```

**session\_id:** The session identifier of the session used to feed the high level operations from which the sequence operations were deduced. The sessions are created using the **create\_session** method, as specified in [\[MS-FSID\]](#) section 3.2.4.1.

**document\_collection:** The content collection associated with the sequence operations.

**low\_sequence\_id:** The lowest sequence identifier of the operations contained in the structure.

**high\_sequence\_id:** The highest sequence identifier of the operations contained in the structure.

**operations:** A collection of **sequence\_operations**, as specified in section [2.2.2](#).

### 2.2.11 cheetah

The **cheetah** data type is an alias for a collection of bytes. It is used for all custom data types that are marshaled using Cheetah. The structure of this data type, as specified in section [6.2](#), is as follows:

```
typedef sequence<octet> cheetah;
```

## 3 Protocol Details

This protocol consists of four different protocol interfaces: **column\_master**, **column\_backup**, **content\_operation\_sequence\_store**, and **sequence\_receptor**. For **column\_master** and **content\_operation\_sequence\_store** the master indexer node acts as a protocol server, and the backup indexing nodes act as the protocol client. For **column\_backup** and **sequence\_receptor**, the roles are reversed, and the master indexer node is the protocol client, while the backup indexing nodes are the protocol servers.

### 3.1 column\_master and content\_operation\_sequence\_store Server Details

A master indexer node implementing the **column\_master** and **content\_operation\_sequence\_store** interfaces receives messages from the backup indexing nodes.

#### 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 following data structures are needed by the master indexer node in the role of protocol server:

**fault\_tolerance\_storage:** a data structure containing a backlog of sequence operations.

**lowest\_sequence\_id:** an integer value containing the sequence identifier of the oldest sequence operation stored in the **fault\_tolerance\_storage**.

**highest\_sequence\_id:** an integer value containing the sequence identifier of the newest sequence operation stored in the **fault\_tolerance\_storage**.

**highest\_processed\_id:** an integer value containing the sequence identifier of the newest sequence operation that the indexing node has processed.

**column\_role:** the current column role of the indexing node in the index column, either "BACKUP," "MASTER," or "UNKNOWN." See section [3.1.3](#) for details on establishing the column role.

**backups:** a set of client proxies implementing the **column\_backup** interface. The client proxies are not resolved using the **name server**, rather they are registered using the **register\_backup\_node** method of the **column\_master** interface.

**file\_receivers:** a set of client proxies implementing the **file\_receiver** interface, specified in [\[MS-FSRFCO\]](#). The file receivers are registered and deregistered by using the Indexer Fault Tolerance protocol; however, the actual use of the client proxies are in relation with the protocol specified in [\[MS-FSRFCO\]](#).

#### 3.1.2 Timers

None.

### 3.1.3 Initialization

Unless the column role of an indexing node is decided through static and implementation-specific configuration options, the decision about which column role to assume is established through the following steps, performed at system initialization:

The column role is set to "UNKNOWN".

The indexing node tries to resolve the **column\_master** interface in the name server, using the **resolve** method, as specified in [\[MS-FSMW\]](#). There are two possible outcomes:

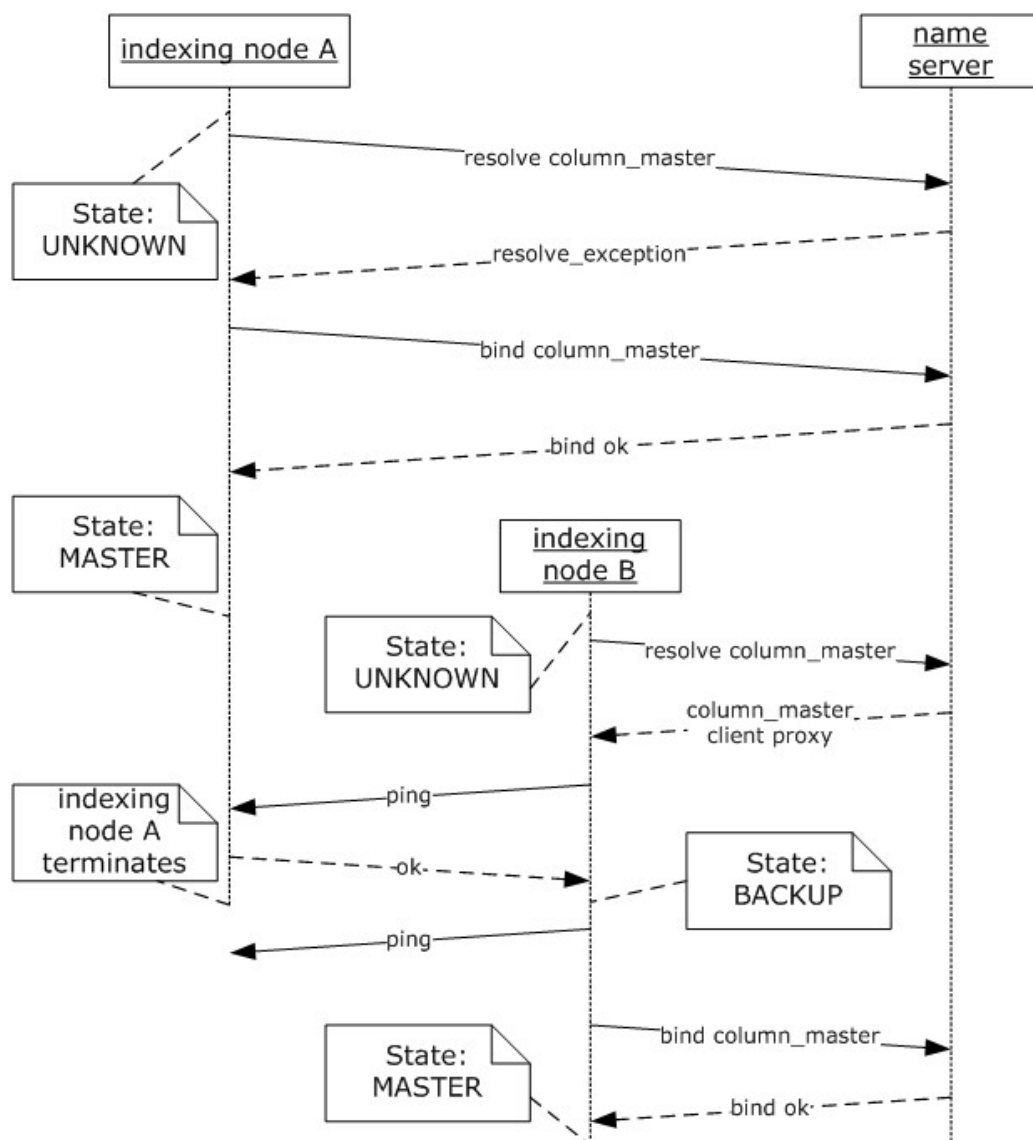
The interface is successfully resolved, and the indexing node tries to ascertain the availability of the current master indexer node using the standard Middleware **\_\_ping** method, as specified in [\[MS-FSMW\]](#) section 4.2. There are two possible outcomes:

The remote indexing node responds to the **\_\_ping**, and the local indexing node assumes the role of "BACKUP".

The remote indexing node is unavailable, and the local indexing node tries to bind the **column\_master** interface itself. If it succeeds in binding to the **column\_master** interface, it assumes the role of "MASTER;" otherwise, the local indexing node repeats step 2.

The interface cannot be resolved, and the local indexing node tries to bind the **column\_master** interface itself. If it succeeds in binding to the **column\_master** interface, it assumes the role of "MASTER;" otherwise, the local indexing node repeats step 2.

When the backup indexing nodes are established, they will periodically ascertain the availability of the master indexer node, and if the master indexer node fails to respond to a **\_\_ping**, the backup indexing nodes will try to bind to the **column\_master** interface and assume the role of master indexer node. A typical scenario is described in the following figure.



**Figure 3: Establishing column role**

If the indexing nodes have predefined roles, a backup indexing node will not try to register the **column\_master** interface in the absence of a master indexer node, rather it will remain a backup indexing node and continually try to resolve the predefined master indexer node.

The master indexer node MUST use the Middleware **bind** method to register an **rtsearch::column\_master** server object in the name server, as specified in [MS-FSMW] section 2.2. The parameters for the **bind** method are encapsulated in an **abstract object reference (AOR)**, as specified in [MS-FSMW] section 3.4.4.2:

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

**port:** An integer specifying the port number of the server object on the protocol server. The value is **base port** plus 390.

**interface\_type:** A string that MUST be "rtsearch::column\_master".

**interface\_version:** A string that MUST be "5.9".

**object\_id:** An integer value that MUST be unique for each server object.

**name:** A string that MUST be "esp/clusters/webcluster/indexing/indexer-C/columnmaster", where C is the index column identifier.

If fault-tolerant behavior is enabled, all indexing nodes MUST use the Middleware **bind** method to register an **rtsearch::content\_operation\_sequence\_store** server object in the name server, as specified in [\[MS-FSMW\]](#) section 2.2. The parameters for the **bind** method are encapsulated in an abstract object reference (AOR), as specified in [\[MS-FSMW\]](#) section 3.4.4.2:

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

**port:** An integer specifying the port number of the server object on the protocol server. The value is base port plus 390.

**interface\_type:** A string that MUST be "rtsearch::content\_operation\_sequence\_store".

**interface\_version:** A string that MUST be "5.6".

**object\_id:** An integer value that MUST be unique for each server object.

**name:** A string that MUST be "esp/clusters/webcluster/indexing/indexer-C-R/opr\_seq\_store", where C is the index column identifier and R is the indexer row identifier.

### 3.1.4 Message Processing Events and Sequencing Rules

The message type is determined at the Middleware level. The Middleware MUST call the correct method of a server object implementing an interface. If custom data types are present in the signature of the method being called, the Middleware MUST **unmarshal (1)** the Cheetah data before passing the arguments to the server object.

In accordance with the Middleware specification, the generic Middleware exceptions may be thrown from any method, and are thus not defined in the FSIDL method signatures.

The available methods are specified in the following table.

Method	Description
<b>column_master::get_row_id</b>	Returns the indexer row of the indexing node.
<b>column_master::register_backup_node</b>	Registers a backup indexing node.
<b>column_master::has_backup_node</b>	Returns whether a specific indexing node is registered with the master indexer node.
<b>column_master::check_backup_nodes</b>	Unregisters unresponsive backup indexing nodes.
<b>column_master::abdicate</b>	Relinquishes status as master indexer node.
<b>column_master::connect_receiver</b>	Adds a file receiver to the

Method	Description
	<b>file_receivers</b> state variable.
<b>column_master::disconnect_receiver</b>	Removes a file receiver from the <b>file_receivers</b> state variable.
<b>content_operation_sequence_store::is_master</b>	Returns whether the indexing node is the master indexer node of the index column.
<b>content_operation_sequence_store::get_stored_sequences</b>	Returns the range of sequence operations stored in the fault-tolerant storage.
<b>content_operation_sequence_store::has_sequence_id</b>	Returns whether the fault-tolerant storage contains a specified sequence operation.
<b>content_operation_sequence_store::request_sequences</b>	Requests an asynchronous transfer of sequence operations.
<b>content_operation_sequence_store::get_row_id</b>	Returns the indexer row of the indexing node.
<b>content_operation_sequence_store::get_hostname</b>	Returns the host name of the indexing node.
<b>content_operation_sequence_store::get_highest_sequence_id</b>	Returns the value of the state variable <b>highest_sequence_id</b> .
<b>content_operation_sequence_store::get_lowest_sequence_id</b>	Returns the value of the state variable <b>lowest_sequence_id</b> .

#### 3.1.4.1 column\_master::get\_row\_id

The **get\_row\_id** method MUST return the indexer row of the indexing node. The structure of this method, as specified in section [6.1](#), is as follows:

```
long get_row_id();
```

**Return values:** MUST return the indexer row of the indexing node.

#### 3.1.4.2 column\_master::register\_backup\_node

The **register\_backup\_node** method MUST add the backup indexing node **client proxy** to the state variable **backups**. The structure of this method, as specified in section [6.1](#), is as follows:

```
void register_backup_node(in column_backup backup_interface,
                          in long row_id);
```

**backup\_interface:** The client proxy of the backup indexing node registering.

**row\_id:** The indexer row of the backup indexing node registering.

### 3.1.4.3 column\_master::has\_backup\_node

The **has\_backup\_node** method MUST return whether the state variable **backups** contains the backup indexing node of a specific indexer row. The structure of this method, as specified in section [6.1](#), is as follows:

```
boolean has_backup_node(in long row_id);
```

**row\_id:** The indexer row identifier of the backup indexing node.

**Return values:** The method MUST return **true** if the state variable **backups** contains the specified backup indexing node; otherwise **false**.

### 3.1.4.4 column\_master::check\_backup\_nodes

The **check\_backup\_nodes** method MUST remove all unavailable backup indexing nodes from the state variable **backups**. The availability of the backup indexing nodes MUST be ascertained using the **\_\_ping** method on the client proxies, as specified in [\[MS-FSMW\]](#) section 4.2. The structure of this method, as specified in section [6.1](#), is as follows:

```
void check_backup_nodes();
```

### 3.1.4.5 column\_master::abdicate

The **abdicate** method MUST force the master indexer node to relinquish its status as master indexer node by unbinding the **column\_master** interface from the name server, as specified in [\[MS-FSMW\]](#) section 3.4.4.3. The **column\_role** state variable MUST be set to "UNKNOWN", and the column role must be newly established in accordance with the description in section [3.1.3](#). The **abdicate** method MAY return a void value, but SHOULD simply time out. The structure of this method, as specified in section [6.1](#), is as follows:

```
void abdicate();
```

### 3.1.4.6 column\_master::connect\_receiver

The **connect\_receiver** method MUST add the **file\_receiver** client proxy to the state variable **file\_receivers**. The **file\_receiver** interface is specified in [\[MS-FSRFCO\]](#). The availability of the file receiver server object MUST be ascertained using the **\_\_ping** method on the client proxy, as specified in [\[MS-FSMW\]](#) section 4.2. The structure of this method, as specified in section [6.1](#), is as follows:

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

**receiver:** The client proxy of the connecting **file\_receiver**, as specified in [\[MS-FSRFCO\]](#).

**hostname:** The host name of the connecting **file\_receiver**.

**port:** The port number of the connecting **file\_receiver**.

**Return values:** The method MUST return **true** if the **file\_receivers** state variable is successfully updated and the server object is available; otherwise **false**.

#### 3.1.4.7 column\_master::disconnect\_receiver

The **disconnect\_receiver** method MUST remove the **file\_receiver** client proxy associated with the specified host name and port number from the state variable **file\_receivers**. The structure of this method, as specified in section [6.1](#), is as follows:

```
boolean disconnect_receiver(in string hostname, in long port);
```

**hostname:** The host name of the disconnecting **file\_receiver**.

**port:** The port number of the disconnecting **file\_receiver**.

**Return values:** The method MUST return **true**.

#### 3.1.4.8 content\_operation\_sequence\_store::is\_master

The **is\_master** method MUST return whether or not the **column\_role** state variable equals "MASTER". The structure of this method, as specified in section [6.1](#), is as follows:

```
boolean is_master();
```

**Return values:** The method MUST return **true** if the state variable **column\_role** equals "MASTER"; otherwise **false**.

#### 3.1.4.9 content\_operation\_sequence\_store::get\_stored\_sequences

The **get\_stored\_sequences** method MUST return the range of sequence operations stored in the **fault\_tolerance\_storage** and the **highest\_processed\_id** state variables. The structure of this method, as specified in section [6.1](#), is as follows:

```
cht::rtsmessages::sequence_log_info get_stored_sequences();
```

**Return values:** The method MUST return a **sequence\_log\_info** object containing the state variables **highest\_sequence\_id**, **lowest\_sequence\_id** and **highest\_processed\_id**.

#### 3.1.4.10 content\_operation\_sequence\_store::has\_sequence\_id

The **has\_sequence\_id** method MUST return whether the **fault\_tolerance\_storage** contains the specified sequence operation. The structure of this method, as specified in section [6.1](#), is as follows:

```
boolean has_sequence_id(in long long sequence_id);
```

**sequence\_id:** The sequence identifier.

**Return values:** The method MUST return **true** if the **fault\_tolerance\_storage** contains the sequence operation; otherwise **false**.

#### 3.1.4.11 content\_operation\_sequence\_store::request\_sequences

The **request\_sequences** method MUST initiate an asynchronous transfer of a range of sequence operations using **sequence\_receptor::submit\_sequence**. When all of the sequence operations have been submitted, a call to the **sequence\_receptor::finished** method MUST be made. The structure of this method, as specified in section [6.1](#), is as follows:

```
void request_sequences(in sequence_receptor receptor,
                      in long long from_sequence_id,
                      in long long to_sequence_id);
```

**receptor:** The **sequence\_receptor** client proxy through which to send the sequence operations.

**from\_sequence\_id:** The lowest identifier of the range of sequence operations to transfer.

**to\_sequence\_id:** The highest identifier of the range of sequence operations to transfer.

#### 3.1.4.12 content\_operation\_sequence\_store::get\_row\_id

The **get\_row\_id** method MUST return the indexer row identifier of the indexing node. The structure of this method, as specified in section [6.1](#), is as follows:

```
long get_row_id();
```

**Return values:** The method MUST return the indexer row identifier of the indexing node.

#### 3.1.4.13 content\_operation\_sequence\_store::get\_hostname

The **get\_hostname** method MUST return the fully qualified domain name (FQDN) of the indexing node. The structure of this method, as specified in section [6.1](#), is as follows:

```
string get_hostname();
```

**Return values:** The method MUST return the fully qualified domain name of the indexing node.

#### 3.1.4.14 content\_operation\_sequence\_store::get\_highest\_sequence\_id

The **get\_highest\_sequence\_id** method MUST return the value of the state variable **highest\_sequence\_id**. The structure of this method, as specified in section [6.1](#), is as follows:

```
long long get_highest_sequence_id();
```

**Return values:** The method MUST return the value of the state variable **highest\_sequence\_id**.

#### 3.1.4.15 content\_operation\_sequence\_store::get\_lowest\_sequence\_id

The **get\_lowest\_sequence\_id** method MUST return the value of the state variable **lowest\_sequence\_id**. The structure of this method, as specified in section [6.1](#), is as follows:

```
long long get_lowest_sequence_id();
```

**Return values:** The method MUST return the value of the state variable **lowest\_sequence\_id**.

### 3.1.5 Timer Events

None.

### 3.1.6 Other Local Events

None.

## 3.2 column\_master and content\_operation\_sequence\_store Client Details

The backup indexing nodes use the **column\_master** interface to register with and poll information from the master indexer node. The backup indexing nodes use the **content\_operation\_sequence\_store** interface to obtain missing sequence operations from another indexing node.

### 3.2.1 Abstract Data Model

None.

### 3.2.2 Timers

None.

### 3.2.3 Initialization

The protocol client MUST use the Middleware **resolve** method to find the server objects bound in the name server, as specified in [\[MS-FSMW\]](#) section 3.4.4.1.

For resolving the **content\_operation\_sequence\_store** server object, the parameters are:

**name:** A string that MUST be "esp/clusters/webcluster/indexing/indexer-C-R/opr\_seq\_store", where *C* is the index column identifier and *R* is the indexer row identifier.

**interface\_type:** A string that MUST be "rtsearch::content\_operation\_sequence\_store".

**interface\_version:** A string that MUST be "5.6".

For resolving the **column\_master** server object, the parameters are:

**name:** A string that MUST be "esp/clusters/webcluster/indexing/indexer-C/columnmaster", where *C* is the index column identifier.

**interface\_type:** A string that MUST be "rtsearch::column\_master".

**interface\_version:** A string that MUST be "5.9".

The backup indexing nodes MUST create a server object implementing the **column\_backup** interface. The server object must then be registered with the master indexer node using **column\_master::register\_backup\_node**. The server object MUST have the following values set for the abstract object reference (AOR), as specified in [\[MS-FSMW\]](#) section 2.2.14:

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

**port:** An integer specifying the port number of the server object on the protocol server. The value is base port plus 390.

**interface\_type:** A string that MUST be "rtsearch::column\_backup".

**interface\_version:** A string that MUST be "5.14".

**object\_id:** An integer value that MUST be unique for each server object.

To call methods that have parameters of type **sequence\_receptor**, a server object implementing the interface **sequence\_receptor** MUST first be created. The server object MUST have the following values set for the abstract object reference (AOR), as specified in [\[MS-FSMW\]](#) section 2.2.14:

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

**port:** An integer specifying the port number of the server object on the protocol server. The value is base port plus 390.

**interface\_type:** A string that MUST be "rtsearch::sequence\_receptor".

**interface\_version:** A string that MUST be "5.2".

**object\_id:** An integer value that MUST be unique for each server object.

### 3.2.4 Message Processing Events and Sequencing Rules

The available methods are specified in the following table.

Method	Description
<b>column_master::get_row_id</b>	Returns the indexer row of the indexing node.
<b>column_master::register_backup_node</b>	Registers a backup indexing node.
<b>column_master::has_backup_node</b>	Returns whether a specific indexing node is registered with the master indexer node.
<b>column_master::check_backup_nodes</b>	Unregisters unresponsive backup indexing nodes.
<b>column_master::abdicate</b>	Relinquishes status as master indexer node.
<b>column_master::connect_receiver</b>	Adds a file receiver to the <b>file_receivers</b> state variable.
<b>column_master::disconnect_receiver</b>	Removes a file receiver from the <b>file_receivers</b> state variable.
<b>content_operation_sequence_store::is_master</b>	Returns whether the indexing node is the master indexer node of the index column.
<b>content_operation_sequence_store::get_stored_sequences</b>	Returns the range of sequence operations stored in the fault-tolerance storage.

Method	Description
<b>content_operation_sequence_store::has_sequence_id</b>	Returns whether the fault-tolerance storage contains a specified sequence operation.
<b>content_operation_sequence_store::request_sequences</b>	Requests an asynchronous transfer of sequence operations.
<b>content_operation_sequence_store::get_row_id</b>	Returns the indexer row of the indexing node.
<b>content_operation_sequence_store::get_hostname</b>	Returns the host name of the indexing node.
<b>content_operation_sequence_store::get_highest_sequence_id</b>	Returns the value of the state variable <b>highest_sequence_id</b> .
<b>content_operation_sequence_store::get_lowest_sequence_id</b>	Returns the value of the state variable <b>lowest_sequence_id</b> .

#### 3.2.4.1 column\_master::get\_row\_id

The **get\_row\_id** method is specified in section [3.1.4.1](#).

#### 3.2.4.2 column\_master::register\_backup\_node

The **register\_backup\_node** method is specified in section [3.1.4.2](#). Before this call is made, the backup indexing node MUST first create a **column\_backup** server object, as specified in section [3.2.3](#).

#### 3.2.4.3 column\_master::has\_backup\_node

The **has\_backup\_node** method is specified in section [3.1.4.3](#).

#### 3.2.4.4 column\_master::check\_backup\_nodes

The **check\_backup\_nodes** method is specified in section [3.1.4.4](#).

#### 3.2.4.5 column\_master::abdicate

The **abdicate** method is specified in section [3.1.4.5](#). The abdicate method will force the protocol server to deactivate the server object implementing the method itself. The **abdicate** method MAY return a void value, but SHOULD simply time out.

#### 3.2.4.6 column\_master::connect\_receiver

The **connect\_receiver** method is specified in section [3.1.4.6](#).

#### 3.2.4.7 column\_master::disconnect\_receiver

The **disconnect\_receiver** method is specified in section [3.1.4.7](#).

#### 3.2.4.8 content\_operation\_sequence\_store::is\_master

The **is\_master** method is specified in section [3.1.4.8](#).

### 3.2.4.9 `content_operation_sequence_store::get_stored_sequences`

The `get_stored_sequences` method is specified in section [3.1.4.9](#).

### 3.2.4.10 `content_operation_sequence_store::has_sequence_id`

The `has_sequence_id` method is specified in section [3.1.4.10](#).

### 3.2.4.11 `content_operation_sequence_store::request_sequences`

The `request_sequences` method is specified in section [3.1.4.11](#).

### 3.2.4.12 `content_operation_sequence_store::get_row_id`

The `get_row_id` method is specified in section [3.1.4.12](#).

### 3.2.4.13 `content_operation_sequence_store::get_hostname`

The `get_hostname` method is specified in section [3.1.4.13](#).

### 3.2.4.14 `content_operation_sequence_store::get_highest_sequence_id`

The `get_highest_sequence_id` method is specified in section [3.1.4.14](#).

### 3.2.4.15 `content_operation_sequence_store::get_lowest_sequence_id`

The `get_lowest_sequence_id` method is specified in section [3.1.4.15](#).

## 3.2.5 Timer Events

None.

## 3.2.6 Other Local Events

None.

## 3.3 `column_backup` and `sequence_receptor` Server Details

A backup indexing node implementing the `column_backup` and `sequence_receptor` interfaces receive messages from a master indexer node.

### 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 following data structures are needed by the backup indexing node in the role of protocol server:

**`fault_tolerance_storage`:** A data structure containing a backlog of sequence operations.

**`lowest_sequence_id`:** An integer value containing the sequence identifier of the oldest sequence operation stored in the **`fault_tolerance_storage`**.

**highest\_sequence\_id:** An integer value containing the sequence identifier of the newest sequence operation stored in the **fault\_tolerance\_storage**.

**highest\_processed\_id:** An integer value containing the sequence identifier of the newest sequence operation that the indexing node has processed.

**column\_role:** The current column role of the indexing node in the index column, either "BACKUP," "MASTER," or "UNKNOWN." See section [3.1.3](#) for details on establishing the column role.

**active\_index\_set:** The currently active index set.

**sequence\_receptor:** A server object implementing the `sequence_receptor` interface. The server object receives the asynchronous responses to the message

**content\_operation\_sequence\_store::request\_sequences.**

**receptor\_finished:** A Boolean state variable stating whether or not the **sequence\_receptor** has received the submitted set of sequence operations.

### 3.3.2 Timers

None.

### 3.3.3 Initialization

The backup indexing nodes MUST create a server object implementing the **column\_backup**, as specified in section [3.2.3](#).

### 3.3.4 Message Processing Events and Sequencing Rules

The message type is determined at the Middleware level. The Middleware MUST call the correct method of the server object implementing the interface. If custom data types are present in the signature of the method being called, the Middleware MUST unmarshal (1) the Cheetah data before passing the arguments to the server object.

In accordance with the Middleware specification, generic Middleware exceptions may be thrown from any method, and are thus not defined in the FSIDL method signatures.

The available methods are specified in the following table.

Method	Description
<b>column_backup::get_row_id</b>	Returns the indexer row of the indexing node.
<b>column_backup::get_hostname</b>	Returns the host name of the indexing node.
<b>column_backup::submit_sequence</b>	Submits a set of sequence operations.
<b>column_backup::commit_sequence</b>	Persists a set of submitted sequence operations.
<b>column_backup::abort_sequence</b>	Reverts a submitted set of sequence operations.
<b>column_backup::activate_index_set</b>	Updates the active index set.
<b>sequence_receptor::submit_sequence</b>	Submits a set of sequence operations.
<b>sequence_receptor::finished</b>	Sets the state variable <b>receptor_finished</b> to <b>true</b> .

Method	Description
<b>sequence_receptor::get_hostname</b>	Returns the host name of the remote indexing node.

#### 3.3.4.1 column\_backup::get\_row\_id

The **get\_row\_id** method MUST return the indexer row identifier of the indexing node. The structure of this method, as specified in section [6.1](#), is as follows:

```
long get_row_id();
```

**Return values:** The method MUST return the indexer row identifier of the backup indexing node.

#### 3.3.4.2 column\_backup::get\_hostname

The **get\_hostname** method MUST return the fully qualified domain name (FQDN) of the indexing node. The structure of this method, as specified in section [6.1](#), is as follows:

```
string get_hostname();
```

**Return values:** The method MUST return the fully qualified domain name of the indexing node.

#### 3.3.4.3 column\_backup::submit\_sequence

The **submit\_sequence** method MUST update the FIXML files in accordance with the incoming set of sequence operations. The method MUST also update the **highest\_processed\_id** state variable to include the latest set of sequence operations processed. The structure of this method, as specified in section [6.1](#), is as follows:

```
boolean submit_sequence(in cht::rtsmessages::content_operation_sequence seq,
                        in string collection_name)
```

**seq:** The set of sequence operations.

**collection\_name:** The name of the content collection being updated.

**Return values:** The method MUST return **true** if the FIXML files were successfully updated; otherwise, **false**.

#### 3.3.4.4 column\_backup::commit\_sequence

The **commit\_sequence** method MUST update the state variables **lowest\_sequence\_id** and **highest\_sequence\_id** to correspond to the latest set of submitted sequence operations. The structure of this method, as specified in section [6.1](#), is as follows:

```
void commit_sequence();
```

#### 3.3.4.5 column\_backup::abort\_sequence

The **abort\_sequence** method MUST undo the effect of any submitted but not yet committed set of sequence operations. The structure of this method, as specified in section [6.1](#), is as follows:

```
void abort_sequence();
```

#### 3.3.4.6 column\_backup::activate\_index\_set

The **activate\_index\_set** method MUST update the state variable **active\_index\_set** to correspond to the latest index set available on the backup indexing node. The actual index set is transferred to the backup indexing node using a different protocol, as specified in [\[MS-FSRFCO\]](#). The structure of this method, as specified in section [6.1](#), is as follows:

```
void activate_index_set();
```

#### 3.3.4.7 sequence\_receptor::submit\_sequence

The **submit\_sequence** method MUST accept a set of sequence operations, sent as a response to **content\_operation\_sequence\_store::request\_sequences**. The interface is used by indexing nodes recovering from downtime, and the method MUST mirror the functionality of a call to **column\_backup::submit\_sequence** followed by a call to **column\_backup::commit\_sequence**. The structure of this method, as specified in section [6.1](#), is as follows:

```
void submit_sequence(in cht::rtsmessages::content_operation_sequence seq);
```

**seq:** The set of sequence operations submitted from the remote indexing node.

#### 3.3.4.8 sequence\_receptor::finished

The **finished** method is an asynchronous callback sent back to the protocol client as a result of the protocol client issuing a **content\_operation\_sequence\_store::request\_sequences** call. It MUST set the state variable **receptor\_finished** to **true**. The protocol server MUST use it to signal that it has finished submitting the requested sequence operations. The structure of this method, as specified in section [6.1](#), is as follows:

```
void finished();
```

#### 3.3.4.9 sequence\_receptor::get\_hostname

The **get\_hostname** method MUST return the fully qualified domain name (FQDN) of the indexing node. The structure of this method, as specified in section [6.1](#), is as follows:

```
string get_hostname();
```

**Return values:** The method MUST return the fully qualified domain name (FQDN) of the indexing node.

### 3.3.5 Timer Events

None.

### 3.3.6 Other Local Events

None.

## 3.4 column\_backup and sequence\_receptor Client Details

A master indexer node uses the **column\_backup** and **sequence\_receptor** interfaces to interact with a backup indexing node.

### 3.4.1 Abstract Data Model

None.

### 3.4.2 Timers

None.

### 3.4.3 Initialization

The **column\_backup** client proxy is registered with the master indexer node through a call to **column\_master::register\_backup\_node**. The **sequence\_receptor** client proxy is supplied as a parameter to **content\_operation\_sequence\_store::request\_sequences**.

### 3.4.4 Message Processing Events and Sequencing Rules

The available methods are specified in the following table.

Method	Description
<b>column_backup::get_row_id</b>	Returns the indexer row of the indexing node.
<b>column_backup::get_hostname</b>	Returns the host name of the indexing node.
<b>column_backup::submit_sequence</b>	Submits a set of sequence operations.
<b>column_backup::commit_sequence</b>	Persists a set of submitted sequence operations.
<b>column_backup::abort_sequence</b>	Reverts a submitted set of sequence operations.
<b>column_backup::activate_index_set</b>	Updates the active index set.
<b>sequence_receptor::submit_sequence</b>	Submits a set of sequence operations.
<b>sequence_receptor::finished</b>	Sets the state variable <b>receptor_finished</b> to <b>true</b> .
<b>sequence_receptor::get_hostname</b>	Returns the host name of the remote indexing node.

#### 3.4.4.1 column\_backup::get\_row\_id

The **get\_row\_id** method is specified in section [3.3.4.1](#).

#### 3.4.4.2 column:backup::get\_hostname

The **get\_hostname** method is specified in section [3.3.4.2](#).

#### **3.4.4.3 column\_backup::submit\_sequence**

The **submit\_sequence** method is specified in section [3.3.4.3](#).

#### **3.4.4.4 column\_backup::commit\_sequence**

The **commit\_sequence** method is specified in section [3.3.4.4](#).

#### **3.4.4.5 column\_backup::abort\_sequence**

The **abort\_sequence** method is specified in section [3.3.4.5](#).

#### **3.4.4.6 column\_backup::activate\_index\_set**

The **activate\_index\_set** method is specified in section [3.3.4.6](#).

#### **3.4.4.7 sequence\_receptor::submit\_sequence**

The **submit\_sequence** method is specified in section [3.3.4.7](#).

#### **3.4.4.8 sequence\_receptor::finished**

The **finished** method is specified in section [3.3.4.8](#).

#### **3.4.4.9 sequence\_receptor::get\_hostname**

The **get\_hostname** method is specified in section [3.3.4.9](#).

### **3.4.5 Timer Events**

None.

### **3.4.6 Other Local Events**

None.

## 4 Protocol Examples

### 4.1 Recover a Backup Indexing Node

This example illustrates how this protocol is used in a user scenario where a lagging backup indexing node synchronizes with the master indexer node by requesting missing sequence operations.

The recovery process typically involves the following messages:

1. The backup indexing node resolves the master indexer node in the name server.
2. The backup indexing node requests the range endpoint values of the sequence operations stored on the master indexer node.
3. The backup indexing node requests an asynchronous transfer of missing sequence operations from the master indexer node.
4. The master indexer node submits the missing sequence operations to the backup indexing node.
5. The master indexer node submits an end of transmission message to the backup indexing node.

The scenario presupposes that the master indexer node has already created a server object implementing the **operation\_sequence\_store** interface and bound the server object in a name server using a logical name known to the backup indexing node.

When a backup indexing node is started, it first resolves the **content\_operation\_sequence\_store** interface in the name server. The backup indexing node then proceeds by requesting the range of sequence operations the master indexer node has stored in its fault-tolerant storage. If the backup indexing node does not possess all of the sequence operations on the master indexer node, the backup indexing node must request an asynchronous transfer. Before this request can be made, the backup indexing node must create a server object implementing the **sequence\_receptor** interface, so that the backup indexing node can pass a reference to the recipient of the sequence operations. Using the client proxy to the **sequence\_receptor**, the master indexer node submits the missing sequence operations to the backup indexing node. After the transfer of sequence operations, a message indicating that all operations have been submitted is sent to the backup indexing node.

#### 4.1.1 Recovery Code

```
SET server_object_name TO "esp/clusters/webcluster/indexing/indexer-0-0/opr_seq_store"

SET server_object_type TO "rtsearch::content_operation_sequence_store"

SET server_object_version TO 5.6

CALL nameserver.resolve WITH server_object_name, server_object_type AND server_object_version
RETURNING op_seq_store_client_proxy

CALL op_seq_store_client_proxy.get_stored_sequences RETURNING remote_stored_sequences

IF local_stored_sequences.high_sequence_id < remote_stored_sequences.high_sequence_id THEN
  SET sequence_receptor_server_object TO INSTANCE OF sequence_receptor
  SERVER OBJECT

  CALL op_seq_store_client_proxy.request_sequences WITH sequence_receptor_server_object,
    local_stored_sequences.high_sequence_id + 1, remote_stored_sequences.high_sequence
ENDIF
```

```

REPEAT
  PRINT "Waiting for master indexer node to submit missing sequence operations"
UNTIL sequence_receptor_server_object.finished HAS BEEN CALLED

```

#### 4.1.2 Recovery Sequence Diagram

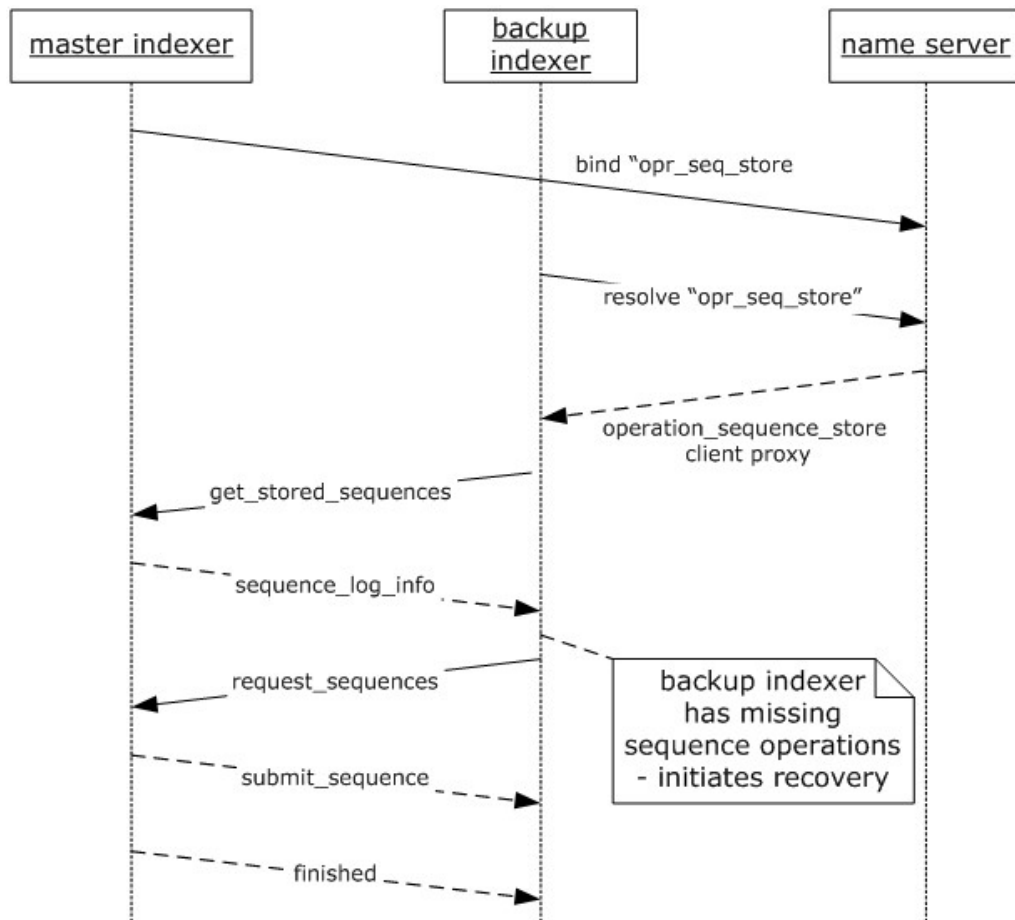


Figure 4: Recovery sequence diagram

## **5 Security**

### **5.1 Security Considerations for Implementers**

None.

### **5.2 Index of Security Parameters**

None.

## 6 Appendix A: Full FSIDL

For ease of implementation, the full FSIDL and Cheetah specifications are provided in the following sections.

### 6.1 FSIDL

```
module cht {
    module rtmessages {
        typedef sequence<octet> cheetah;
        typedef cheetah sequence_log_info;
        typedef cheetah content_operation_sequence;
    };
};

module interfaces {
    module rtsearch {
        interface column_backup {
            long get_row_id();
            string get_hostname();
            boolean submit_sequence(in cht::rtmessages::content_operation_sequence seq,
                                   in string collection_name);
            void commit_sequence();
            void abort_sequence();
            void activate_index_set();
        }

        interface column_master {
            long get_row_id();
            void register_backup_node(in column_backup backup_interface,
                                     in long row_id);
            boolean has_backup_node(in long row_id);
            void check_backup_nodes();
            void abdicate();
            boolean connect_receiver(in file_receiver receiver,
                                    in string hostname,
                                    in long port);
            boolean disconnect_receiver(in string hostname, in long port);
        }

        interface sequence_receptor {
            void submit_sequence(in cht::rtmessages::content_operation_sequence seq);
            void finished();
            string get_hostname();
        }

        interface content_operation_sequence_store {
            boolean is_master();
            cht::rtmessages::sequence_log_info get_stored_sequences();
            boolean has_sequence_id(in long long sequence_id);
            void request_sequences(in sequence_receptor receptor,
                                  in long long from_sequence_id,
                                  in long long to_sequence_id);

            long get_row_id();
            string get_hostname();
            long long get_highest_sequence_id();
            long long get_lowest_sequence_id();
        }
    }
}
```

```
}  
}
```

## 6.2 Cheetah

```
entity sequence_log_info {  
    attribute longint low_sequence_id;  
    attribute longint high_sequence_id;  
    attribute longint processed_sequence_id;  
};  
  
entity sequence_operation {  
    attribute longint sequence_number;  
    attribute longint operation_id;  
};  
  
entity empty_operation : sequence_operation {  
};  
  
entity fixml_invalidation : sequence_operation {  
    attribute string document_id;  
    attribute int file_id;  
    attribute int magic_idx;  
    attribute bool is_update;  
};  
  
entity remove_collection : sequence_operation {  
};  
  
entity fixml_append : sequence_operation {  
    attribute string document_id;  
    attribute string document_content;  
    attribute int file_id;  
    attribute int magic_idx;  
    attribute bool is_update;  
};  
  
entity remdoclist : sequence_operation {  
    attribute string document_id;  
    attribute int old_file_id;  
    attribute int new_file_id;  
};  
  
entity exclusionlist : sequence_operation {  
    attribute string document_id;  
    attribute int old_file_id;  
};  
  
entity document_error : sequence_operation {  
    attribute string document_id;  
    attribute int error_code;  
    attribute int action;  
    attribute string subsystem;  
    attribute string error_message;  
}  
  
entity content_operation_sequence {
```

```
attribute int session_id;  
attribute string document_collection;  
attribute longint low_sequence_id;  
attribute longint high_sequence_id;  
collection sequence_operation operations;  
};
```

## 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

Abstract data model  
  client ([section 3.2.1](#) 25, [section 3.4.1](#) 32)  
  server ([section 3.1.1](#) 17, [section 3.3.1](#) 28)  
[Applicability](#) 10

### C

[Capability negotiation](#) 10  
[Change tracking](#) 41  
[cheetah data type](#) 16  
[Cheetah specification](#) 38  
Client  
  abstract data model ([section 3.2.1](#) 25, [section 3.4.1](#) 32)  
  [column::backup::get\\_hostname method](#) 32  
  [column\\_backup and sequence\\_receptor interface](#) 32  
  [column\\_backup::abort\\_sequence method](#) 33  
  [column\\_backup::activate\\_index\\_set method](#) 33  
  [column\\_backup::commit\\_sequence method](#) 33  
  [column\\_backup::get\\_row\\_id method](#) 32  
  [column\\_backup::submit\\_sequence method](#) 33  
  column\_master and  
    [content\\_operation\\_sequence\\_store interface](#) 25  
  [column\\_master::abdicate method](#) 27  
  [column\\_master::check\\_backup\\_nodes method](#) 27  
  [column\\_master::connect\\_receiver method](#) 27  
  [column\\_master::disconnect\\_receiver method](#) 27  
  [column\\_master::get\\_row\\_id method](#) 27  
  [column\\_master::has\\_backup\\_node method](#) 27  
  [column\\_master::register\\_backup\\_node method](#) 27  
  [content\\_operation\\_sequence\\_store::get\\_highest\\_sequence\\_id method](#) 28  
  [content\\_operation\\_sequence\\_store::get\\_hostname method](#) 28  
  [content\\_operation\\_sequence\\_store::get\\_lowest\\_sequence\\_id method](#) 28  
  [content\\_operation\\_sequence\\_store::get\\_row\\_id method](#) 28  
  [content\\_operation\\_sequence\\_store::get\\_stored\\_sequences method](#) 28  
  [content\\_operation\\_sequence\\_store::has\\_sequence\\_id method](#) 28  
  [content\\_operation\\_sequence\\_store::is\\_master method](#) 27  
  [content\\_operation\\_sequence\\_store::request\\_sequences method](#) 28  
  initialization ([section 3.2.3](#) 25, [section 3.4.3](#) 32)  
  local events ([section 3.2.6](#) 28, [section 3.4.6](#) 33)  
  message processing ([section 3.2.4](#) 26, [section 3.4.4](#) 32)  
  overview ([section 3.2](#) 25, [section 3.4](#) 32)  
  [sequence\\_receptor::finished method](#) 33  
  [sequence\\_receptor::get\\_hostname method](#) 33  
  [sequence\\_receptor::submit\\_sequence method](#) 33

  sequencing rules ([section 3.2.4](#) 26, [section 3.4.4](#) 32)  
  timer events ([section 3.2.5](#) 28, [section 3.4.5](#) 33)  
  timers ([section 3.2.2](#) 25, [section 3.4.2](#) 32)  
  [Column Backup interface](#) 8  
  [Column Master interface](#) 8  
  [column::backup::get\\_hostname method](#) 32  
  column\_backup and sequence\_receptor interface ([section 3.3](#) 28, [section 3.4](#) 32)  
  column\_backup::abort\_sequence method ([section 3.3.4.5](#) 30, [section 3.4.4.5](#) 33)  
  column\_backup::activate\_index\_set method ([section 3.3.4.6](#) 31, [section 3.4.4.6](#) 33)  
  column\_backup::commit\_sequence method ([section 3.3.4.4](#) 30, [section 3.4.4.4](#) 33)  
  [column\\_backup::get\\_hostname method](#) 30  
  column\_backup::get\_row\_id method ([section 3.3.4.1](#) 30, [section 3.4.4.1](#) 32)  
  column\_backup::submit\_sequence method ([section 3.3.4.3](#) 30, [section 3.4.4.3](#) 33)  
  column\_master and  
    content\_operation\_sequence\_store interface ([section 3.1](#) 17, [section 3.2](#) 25)  
  column\_master::abdicate method ([section 3.1.4.5](#) 22, [section 3.2.4.5](#) 27)  
  column\_master::check\_backup\_nodes method ([section 3.1.4.4](#) 22, [section 3.2.4.4](#) 27)  
  column\_master::connect\_receiver method ([section 3.1.4.6](#) 22, [section 3.2.4.6](#) 27)  
  column\_master::disconnect\_receiver method ([section 3.1.4.7](#) 23, [section 3.2.4.7](#) 27)  
  column\_master::get\_row\_id method ([section 3.1.4.1](#) 21, [section 3.2.4.1](#) 27)  
  column\_master::has\_backup\_node method ([section 3.1.4.3](#) 22, [section 3.2.4.3](#) 27)  
  column\_master::register\_backup\_node method ([section 3.1.4.2](#) 21, [section 3.2.4.2](#) 27)  
  [Common data types](#) 11  
  [Content Operation Sequence interface](#) 9  
  [content\\_operation\\_sequence data type](#) 15  
  content\_operation\_sequence\_store::get\_highest\_sequence\_id method ([section 3.1.4.14](#) 24, [section 3.2.4.14](#) 28)  
  content\_operation\_sequence\_store::get\_hostname method ([section 3.1.4.13](#) 24, [section 3.2.4.13](#) 28)  
  content\_operation\_sequence\_store::get\_lowest\_sequence\_id method ([section 3.1.4.15](#) 24, [section 3.2.4.15](#) 28)  
  content\_operation\_sequence\_store::get\_row\_id method ([section 3.1.4.12](#) 24, [section 3.2.4.12](#) 28)  
  content\_operation\_sequence\_store::get\_stored\_sequences method ([section 3.1.4.9](#) 23, [section 3.2.4.9](#) 28)  
  content\_operation\_sequence\_store::has\_sequence\_id method ([section 3.1.4.10](#) 23, [section 3.2.4.10](#) 28)

content\_operation\_sequence\_store::is\_master  
method ([section 3.1.4.8](#) 23, [section 3.2.4.8](#) 27)  
content\_operation\_sequence\_store::request\_sequences method ([section 3.1.4.11](#) 24, [section 3.2.4.11](#) 28)

## D

Data model - abstract  
client ([section 3.2.1](#) 25, [section 3.4.1](#) 32)  
server ([section 3.1.1](#) 17, [section 3.3.1](#) 28)

Data types  
[cheetah](#) 16  
[common - overview](#) 11  
[content\\_operation\\_sequence](#) 15  
[document\\_error](#) 14  
[empty\\_operation](#) 12  
[exclusionlist](#) 13  
[fixml\\_append](#) 14  
[fixml\\_invalidation](#) 12  
[remdoclist](#) 13  
[remove\\_collection](#) 13  
[sequence\\_log\\_info](#) 11  
[sequence\\_operation](#) 12  
[document\\_error data type](#) 14

## E

[empty\\_operation data type](#) 12

### Events

local - client ([section 3.2.6](#) 28, [section 3.4.6](#) 33)  
local - server ([section 3.1.6](#) 25, [section 3.3.6](#) 32)  
timer - client ([section 3.2.5](#) 28, [section 3.4.5](#) 33)  
timer - server ([section 3.1.5](#) 25, [section 3.3.5](#) 31)

### Examples

[recover a backup indexing node](#) 34  
[recovery code](#) 34  
[recovery sequence diagram](#) 35  
[exclusionlist data type](#) 13

## F

[Fields - vendor-extensible](#) 10  
[fixml\\_append data type](#) 14  
[fixml\\_invalidation data type](#) 12  
[FSIDL](#) 37  
[FSIDL specification](#) 37  
[Full FSIDL](#) 37

## G

[Glossary](#) 6

## I

[Implementer - security considerations](#) 36  
[Index of security parameters](#) 36  
[Informative references](#) 7

### Initialization

client ([section 3.2.3](#) 25, [section 3.4.3](#) 32)  
server ([section 3.1.3](#) 18, [section 3.3.3](#) 29)

### Interfaces

[Column Backup](#) 8  
[Column Master](#) 8  
[Content Operation Sequence Store](#) 9  
[Sequence Receptor](#) 9

### Interfaces - client

[column\\_backup and sequence\\_receptor](#) 32  
[column\\_master and content\\_operation\\_sequence\\_store](#) 25

### Interfaces - server

[column\\_backup and sequence\\_receptor](#) 28  
[column\\_master and content\\_operation\\_sequence\\_store](#) 17  
[Interfaces - server and client](#) 17  
[Introduction](#) 6

## L

### Local events

client ([section 3.2.6](#) 28, [section 3.4.6](#) 33)  
server ([section 3.1.6](#) 25, [section 3.3.6](#) 32)

## M

### Message processing

client ([section 3.2.4](#) 26, [section 3.4.4](#) 32)  
server ([section 3.1.4](#) 20, [section 3.3.4](#) 29)

### Messages

[cheetah data type](#) 16  
[common data types](#) 11  
[content\\_operation\\_sequence data type](#) 15  
[document\\_error data type](#) 14  
[empty\\_operation data type](#) 12  
[exclusionlist data type](#) 13  
[fixml\\_append data type](#) 14  
[fixml\\_invalidation data type](#) 12  
[remdoclist data type](#) 13  
[remove\\_collection data type](#) 13  
[sequence\\_log\\_info data type](#) 11  
[sequence\\_operation data type](#) 12  
[transport](#) 11

### Methods

[column::backup::get\\_hostname](#) 32  
[column\\_backup::abort\\_sequence](#) ([section 3.3.4.5](#) 30, [section 3.4.4.5](#) 33)  
[column\\_backup::activate\\_index\\_set](#) ([section 3.3.4.6](#) 31, [section 3.4.4.6](#) 33)  
[column\\_backup::commit\\_sequence](#) ([section 3.3.4.4](#) 30, [section 3.4.4.4](#) 33)  
[column\\_backup::get\\_hostname](#) 30  
[column\\_backup::get\\_row\\_id](#) ([section 3.3.4.1](#) 30, [section 3.4.4.1](#) 32)  
[column\\_backup::submit\\_sequence](#) ([section 3.3.4.3](#) 30, [section 3.4.4.3](#) 33)  
[column\\_master::abdicate](#) ([section 3.1.4.5](#) 22, [section 3.2.4.5](#) 27)  
[column\\_master::check\\_backup\\_nodes](#) ([section 3.1.4.4](#) 22, [section 3.2.4.4](#) 27)  
[column\\_master::connect\\_receiver](#) ([section 3.1.4.6](#) 22, [section 3.2.4.6](#) 27)  
[column\\_master::disconnect\\_receiver](#) ([section 3.1.4.7](#) 23, [section 3.2.4.7](#) 27)

[column\\_master::get\\_row\\_id](#) ([section 3.1.4.1](#) 21, [section 3.2.4.1](#) 27)  
[column\\_master::has\\_backup\\_node](#) ([section 3.1.4.3](#) 22, [section 3.2.4.3](#) 27)  
[column\\_master::register\\_backup\\_node](#) ([section 3.1.4.2](#) 21, [section 3.2.4.2](#) 27)  
[content\\_operation\\_sequence\\_store::get\\_highest\\_sequence\\_id](#) ([section 3.1.4.14](#) 24, [section 3.2.4.14](#) 28)  
[content\\_operation\\_sequence\\_store::get\\_hostname](#) ([section 3.1.4.13](#) 24, [section 3.2.4.13](#) 28)  
[content\\_operation\\_sequence\\_store::get\\_lowest\\_sequence\\_id](#) ([section 3.1.4.15](#) 24, [section 3.2.4.15](#) 28)  
[content\\_operation\\_sequence\\_store::get\\_row\\_id](#) ([section 3.1.4.12](#) 24, [section 3.2.4.12](#) 28)  
[content\\_operation\\_sequence\\_store::get\\_stored\\_sequences](#) ([section 3.1.4.9](#) 23, [section 3.2.4.9](#) 28)  
[content\\_operation\\_sequence\\_store::has\\_sequence\\_id](#) ([section 3.1.4.10](#) 23, [section 3.2.4.10](#) 28)  
[content\\_operation\\_sequence\\_store::is\\_master](#) ([section 3.1.4.8](#) 23, [section 3.2.4.8](#) 27)  
[content\\_operation\\_sequence\\_store::request\\_sequences](#) ([section 3.1.4.11](#) 24, [section 3.2.4.11](#) 28)  
[sequence\\_receptor::finished](#) ([section 3.3.4.8](#) 31, [section 3.4.4.8](#) 33)  
[sequence\\_receptor::get\\_hostname](#) ([section 3.3.4.9](#) 31, [section 3.4.4.9](#) 33)  
[sequence\\_receptor::submit\\_sequence](#) ([section 3.3.4.7](#) 31, [section 3.4.4.7](#) 33)

## N

[Normative references](#) 6

## O

[Overview \(synopsis\)](#) 7

[Column Backup interface](#) 8

[Column Master interface](#) 8

[Content Operation Sequence interface](#) 9

[Sequence Receptor interface](#) 9

## P

[Parameters - security index](#) 36

[Preconditions](#) 10

[Prerequisites](#) 10

[Product behavior](#) 40

[Protocol details](#) 17

## R

[Recover a backup indexing node example](#) 34

[Recovery code](#) 34

[Recovery sequence diagram](#) 35

References

[informative](#) 7

[normative](#) 6

[Relationship to other protocols](#) 9

[remdoclist data type](#) 13

[remove\\_collection data type](#) 13

## S

Security

[implementer considerations](#) 36

[parameter index](#) 36

[Sequence Receptor interface](#) 9

[sequence\\_log\\_info data type](#) 11

[sequence\\_operation data type](#) 12

[sequence\\_receptor::finished](#) method ([section 3.3.4.8](#) 31, [section 3.4.4.8](#) 33)

[sequence\\_receptor::get\\_hostname](#) method ([section 3.3.4.9](#) 31, [section 3.4.4.9](#) 33)

[sequence\\_receptor::submit\\_sequence](#) method ([section 3.3.4.7](#) 31, [section 3.4.4.7](#) 33)

Sequencing rules

client ([section 3.2.4](#) 26, [section 3.4.4](#) 32)

server ([section 3.1.4](#) 20, [section 3.3.4](#) 29)

Server

[abstract data model](#) ([section 3.1.1](#) 17, [section 3.3.1](#) 28)

[column\\_backup\\_and\\_sequence\\_receptor interface](#) 28

[column\\_backup::abort\\_sequence](#) method 30

[column\\_backup::activate\\_index\\_set](#) method 31

[column\\_backup::commit\\_sequence](#) method 30

[column\\_backup::get\\_hostname](#) method 30

[column\\_backup::get\\_row\\_id](#) method 30

[column\\_backup::submit\\_sequence](#) method 30

[column\\_master and](#)

[content\\_operation\\_sequence\\_store interface](#) 17

[column\\_master::abdicate](#) method 22

[column\\_master::check\\_backup\\_nodes](#) method 22

[column\\_master::connect\\_receiver](#) method 22

[column\\_master::disconnect\\_receiver](#) method 23

[column\\_master::get\\_row\\_id](#) method 21

[column\\_master::has\\_backup\\_node](#) method 22

[column\\_master::register\\_backup\\_node](#) method 21

[content\\_operation\\_sequence\\_store::get\\_highest\\_sequence\\_id](#) method 24

[content\\_operation\\_sequence\\_store::get\\_hostname](#) method 24

[content\\_operation\\_sequence\\_store::get\\_lowest\\_sequence\\_id](#) method 24

[content\\_operation\\_sequence\\_store::get\\_row\\_id](#) method 24

[content\\_operation\\_sequence\\_store::get\\_stored\\_sequences](#) method 23

[content\\_operation\\_sequence\\_store::has\\_sequence\\_id](#) method 23

[content\\_operation\\_sequence\\_store::is\\_master](#) method 23

[content\\_operation\\_sequence\\_store::request\\_sequences](#) method 24

initialization ([section 3.1.3](#) 18, [section 3.3.3](#) 29)

local events ([section 3.1.6](#) 25, [section 3.3.6](#) 32)

message processing ([section 3.1.4](#) 20, [section 3.3.4](#) 29)

overview ([section 3.1](#) 17, [section 3.3](#) 28)  
[sequence\\_receptor::finished method](#) 31  
[sequence\\_receptor::get\\_hostname method](#) 31  
[sequence\\_receptor::submit\\_sequence method](#) 31  
sequencing rules ([section 3.1.4](#) 20, [section 3.3.4](#) 29)  
timer events ([section 3.1.5](#) 25, [section 3.3.5](#) 31)  
timers ([section 3.1.2](#) 17, [section 3.3.2](#) 29)  
[Standards assignments](#) 10

## T

Timer events  
  client ([section 3.2.5](#) 28, [section 3.4.5](#) 33)  
  server ([section 3.1.5](#) 25, [section 3.3.5](#) 31)  
Timers  
  client ([section 3.2.2](#) 25, [section 3.4.2](#) 32)  
  server ([section 3.1.2](#) 17, [section 3.3.2](#) 29)  
[Tracking changes](#) 41  
[Transport](#) 11

## V

[Vendor-extensible fields](#) 10  
[Versioning](#) 10