# [MS-SRCHTP]:
# Search Topology 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.

## Revision Summary

| Date | Revision History | Revision Class | Comments |
|---|---|---|---|
| 07/13/2009 | 0.1 | Major | Initial Availability |
| 08/28/2009 | 0.2 | Editorial | Revised and edited the technical content |
| 11/06/2009 | 0.3 | Editorial | Revised and edited the technical content |
| 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 | 1.05 | Minor | Clarified the meaning of the technical content. |
| 09/27/2010 | 1.05 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 11/15/2010 | 1.06 | Major | Significantly changed the technical content. |
| 12/17/2010 | 1.06 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 03/18/2011 | 1.06 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 06/10/2011 | 1.06 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 01/20/2012 | 1.06 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 04/11/2012 | 1.06 | No change | No changes to the meaning, language, or formatting of the technical content. |
| 07/16/2012 | 1.06 | No change | No changes to the meaning, language, or formatting of the technical content. |

# Table of Contents

# 1 Introduction

This document specifies the Search Topology Protocol. This protocol enables the **application server** and **back-end database server** to configure a **search service application**.

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

## 1.1 Glossary

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

**Augmented Backus-Naur Form (ABNF)**
**Coordinated Universal Time (UTC)**
**cyclic redundancy check (CRC)**
**language code identifier (LCID)**
**path**
**topology**

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

**access URL**
**administration component**
**anchor**
**anchor crawl**
**anchor text**
**application server**
**back-end database server**
**change log**
**colleague**
**compact URL**
**content source**
**crawl**
**crawl component**
**crawl queue**
**crawl rule**
**crawl store**
**crawl topology**
**crawl URL history**
**crawler**
**delete crawl**
**display URL**
**document distribution identifier**
**document identifier**
**excluded item**
**full crawl**
**full-text index catalog**
**group**
**host distribution rule**
**host hop**
**host name**
**incremental crawl**
**index partition**

**index server**
**master crawl component**
**MD5**
**metadata index**
**page hop**
**parent item**
**portal content**
**privacy level**
**protocol**
**query component**
**query text**
**query topology**
**refactoring task**
**refactoring task batch**
**result set**
**return code**
**search application**
**search catalog**
**search component**
**search database**
**search scope compilation identifier**
**search service application**
**start address**
**stored procedure**
**token**
**Transact-Structured Query Language (T-SQL)**
**user profile**
**user profile record identifier**
**XML namespace**
**XML namespace prefix**

The following terms are specific to this document:

**administrative host distribution rule:** A rule that is created by an administrator to ensure that documents from a specific host are crawled by a specific crawl component.

**Application directory:** The directory on an index server or a query server where all files are stored for the purpose of creating a full-text index catalog or performing queries on a full-text index catalog.

**automatic host distribution rule:** A rule that is created by a system automatically after a crawl topology is changed. The rule ensures that documents from a specific host are crawled by a specific crawl component.

**crawl type:** A setting that specifies whether to evaluate all of the users and member groups in the directory service that is crawled, or only those users and member groups that were modified after the last crawl.

**partition scheme:** A database object that maps the partitions of a partitioned table or index to a set of file groups. The number and domain of the partitions of a partitioned table or index are defined by a partition function.

**topology activation action:** An operation that is performed to activate a crawl topology or a query topology. The state of each action is stored as an integer.

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

## 1.2 References

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

### 1.2.1 Normative References

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

[Iseminger] Microsoft Corporation, "SQL Server 2000 Architecture and XML/Internet Support", Volume 1 of Microsoft SQL Server 2000 Reference Library, Microsoft Press, 2001, ISBN 0-7356-1280-3, http://www.microsoft.com/mspress/books/5001.aspx

[MS-CIFO] Microsoft Corporation, "Content Index Format Structure Specification".

[MS-CIPROP] Microsoft Corporation, "Index Propagation Protocol Specification".

[MS-CIPROP2] Microsoft Corporation, "Index Propagation Version 2 Protocol Specification".

[MSDN-TSQL-Ref] Microsoft Corporation, "Transact-SQL Reference", http://msdn.microsoft.com/en-us/library/ms189826(SQL.90).aspx

[MS-DTYP] Microsoft Corporation, "Windows Data Types".

[MS-SMB] Microsoft Corporation, "Server Message Block (SMB) Protocol Specification".

[MS-SQLPADM2] Microsoft Corporation, "SQL Administration Version 2 Protocol Specification".

[MS-SQLPGAT2] Microsoft Corporation, "SQL Gatherer Version 2 Protocol Specification".

[MS-SQLPQ2] Microsoft Corporation, "Search Service Database Query Version 2 Protocol Specification".

[MS-TDS] Microsoft Corporation, "Tabular Data Stream Protocol Specification".

[MS-UPSPROF2] Microsoft Corporation, "User Profile Stored Procedures Version 2 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

[XML10] World Wide Web Consortium, "Extensible Markup Language (XML) 1.0 (Third Edition)", February 2004, http://www.w3.org/TR/REC-xml

[XMLINFOSET] World Wide Web Consortium, "XML Information Set (Second Edition)", February 2004, http://www.w3.org/TR/2004/REC-xml-infoset-20040204

[XMLNS] Bray, T., Hollander, D., Layman, A., et al., Eds., "Namespaces in XML 1.0 (Third Edition)", W3C Recommendation, December 2009, http://www.w3.org/TR/2009/REC-xml-names-20091208/

[XMLSCHEMA1] Thompson, H.S., Ed., Beech, D., Ed., Maloney, M., Ed., and Mendelsohn, N., Ed., "XML Schema Part 1: Structures", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-1-20010502/

[XMLSCHEMA2] Biron, P.V., Ed. and Malhotra, A., Ed., "XML Schema Part 2: Datatypes", W3C Recommendation, May 2001, http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/

### 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)

This protocol specifies the communication between the application server, and the back-end database server used for the configuration of a search service application. This server-to-server protocol uses the Tabular Data Stream Protocol, as described in [MS-TDS], as its transport between the application server and the back-end database server.

This protocol is used for creation and configuration of the components of a search service application. Typical scenarios for this protocol include:

- **Creation and Configuration of a Query Topology**

  This protocol allows clients to create and configure a **query topology** for a search service application. The query topology consists of a set of **index partitions**, **query components (2)** and **metadata indexes** which work together to satisfy requests for search query operations. The following operations are typical tasks associated with configuring a query topology: creation of the query topology, adding a query component (2), associating a query component (2) with an index partition, and activating the query topology.

- **Creation and Configuration of a Crawl Topology**

  This protocol allows clients to create and configure a **crawl topology** for a search service application. The crawl topology consists of a set of **crawl components** and **crawl stores** which work together to satisfy requests for search **crawl** tasks. The following operations are typical tasks associated with configuring a crawl topology: creation of the crawl topology, adding a crawl component, associating a crawl component with a crawl store, and activating the crawl topology.

- **Creation and Configuration of an Administration Component**

  This protocol allows clients to create and configure the properties of the **administration component** of a search service application. The administration component is a **search component** that performs administration tasks. The most common operation associated with the configuration of an administration component is the creation of the administration component.

- **Database Repartitioning**

  To support query topology configurations which distribute the metadata index across multiple databases the protocol allows for clients to update the query topology of a **search application** with a different **partition scheme**.

- **Host Distribution**

  To support crawl topology configurations which distribute the task of performing a crawl across multiple **crawlers** the protocol allows for clients to add, change, and delete **host distribution rules** from a store on the back-end database server.

## 1.4 Relationship to Other Protocols

This protocol relies on [MS-TDS] as its transport protocol to call **stored procedures** to inspect and manipulate item properties via **result sets** and **return codes**.

The following diagram shows the transport stack that the protocol uses:

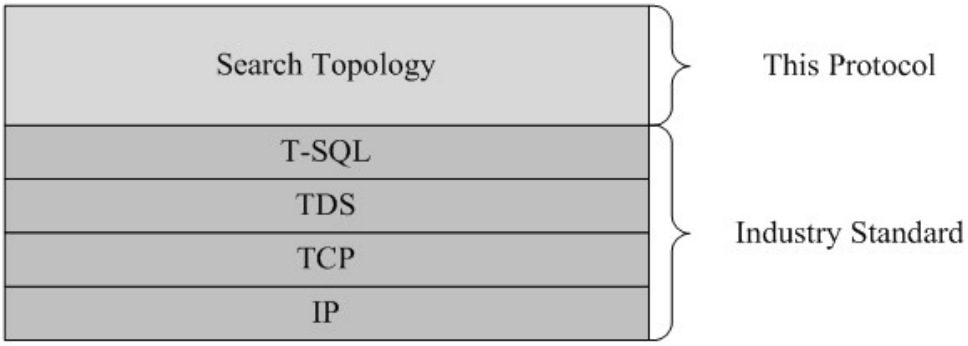


**Figure 1: This protocol in relation to other protocols**

This protocol relies on Server Message Control Block (SMB) Specification [MS-SMB] as its transport protocol to perform server-to-server copies of **full-text index catalog** files.

## 1.5 Prerequisites/Preconditions

Unless otherwise specified, this protocol requires that the stored procedures and any related data be present in the metadata index or crawl store that is being queried on the back-end database server. The metadata index and crawl store contain valid data in a consistent state in the order to be queried successfully by the stored procedures.

## 1.6 Applicability Statement

This protocol is applicable only to the activity of application servers when communicating with the back-end database server for creation and configuration of the components of one particular search application. This protocol is designed for use by no more than 256 index partitions per search service application.

## 1.7 Versioning and Capability Negotiation

### Version Negotiation

Versions of the data structures or stored procedures in the database require the same calling parameters and return code values that are expected by the protocol client in order for the stored procedures to be called correctly. The results of the call are indeterminate if the stored procedures do not provide the same calling parameters or return values as expected. The application server uses stored procedure **proc_MSS_GetDatabaseSchemaVersion** (section 3.1.5.31) to retrieve version of the protocol implemented on the back-end database server and continues using that server only if that version is supported.

### Security and Authentication Methods

This protocol supports the SSPI and SQL Authentication with the back-end database server. These authentication methods are described in [MS-TDS].

## 1.8 Vendor-Extensible Fields

None.

## 1.9 Standards Assignments

None.

# 2 Messages

## 2.1 Transport

[MS-TDS] is the transport protocol used to call the stored procedures, query SQL Views or SQL Tables and return result sets and return codes.

[MS-SMB] is the transport protocol used to copy files from another server.

## 2.2 Common Data Types

This section contains common definitions used by this protocol.

### 2.2.1 Simple Data Types and Enumerations

#### 2.2.1.1 Administration Component Type

An **Administration Component Type** defines the type of an administration component. The value MUST be an integer listed in the following table:

| Value | Description |
|---|---|
| **0** | A regular administration component. |
| **1** | A standalone administration component. |

#### 2.2.1.2 Query Topology State

A **Query Topology State** defines state of a query topology. The value MUST be an integer listed in the following table:

| Symbolic Name | Value | Description |
|---|---|---|
| Inactive | **0** | The query topology is inactive. |
| Active | **1** | The query topology is active. Only one query topology MUST be in this state. |
| Activating | **2** | The query topology is being activated. |
| Deactivating | **3** | The query topology is being deactivated. Not more than one query topology MUST be in the Activating or the Deactivating state. |

#### 2.2.1.3 Query Component State

A **Query Component State** defines state of a query component (2). The value MUST be an integer listed in the following table:

| Symbolic Name | Value | Description |
|---|---|---|
| Uninitialized | **0** | Query component is uninitialized. |
| Ready | **1** | Query component is ready and serves queries. |

| Symbolic Name | Value | Description |
|---|---|---|
| Offline | **2** | Query component is initialized but is not actively updated. |
| IndexSplitDone | **103** | Query component has finished splitting the full-text index catalog. This state is used during index repartitioning. |

### 2.2.1.4 Query Component Type

A **Query Component Type** is used define whether or not a query component (2) is used in presence of other query components that hold the same index partition. The value MUST be an integer listed in the following table:

| Value | Description |
|---|---|
| **0** | A regular query component. Can be used at any time. |
| **1** | A hot-swap query component. It MUST be used only if all other regular query components that have the same index partition are not responding. |

### 2.2.1.5 Query Component Transition Status

A **Query Component Transition Status** defines a state of a query component transition sequence. Its value MUST be an integer listed in the following table:

| Symbolic Name | Value | Description |
|---|---|---|
| Executing | **0** | A query component transition sequence is being executed. |
| Completed | **1** | A query component transition sequence has been completely executed. |
| RollingBack | **2** | A query component transition sequence has failed or been cancelled, and is stepping back to the original state. |
| Canceled | **3** | A query component transition sequence has been proactively cancelled and execution will not continue. |
| Failed | **4** | A query component transition sequence has failed and execution will not continue. |

### 2.2.1.6 Crawl Topology State

A **Crawl Topology State** defines the state of a crawl topology. The value MUST be an integer listed in the following table:

| Symbolic Name | Value | Description |
|---|---|---|
| Inactive | **0** | The crawl topology is inactive. |
| Active | **1** | The crawl topology is active. |
| Activating | **2** | The crawl topology is being activated. |
| Deactivating | **3** | The crawl topology is being deactivated. |

| Symbolic Name | Value | Description |
|---|---|---|
| ActiveToBeRemoved | **4** | The crawl topology is active but will be removed directly after deactivation. |
| DeactivatingToBeRemoved | **5** | The crawl topology is deactivating and will be removed directly after deactivation. |

### 2.2.1.7 Crawl Component State

A **Crawl Component State** defines state of a crawl component. The value MUST be an integer listed in the following table:

| Symbolic Name | Value | Description |
|---|---|---|
| Uninitialized | **0** | The crawl component is uninitialized. |
| Ready | **1** | The crawl component is active. |
| Disabled | **2** | The crawl component is disabled and has been taken out of crawls because it was not responding for more than one hour. |
| Remount | **3** | The crawl component is active and needs to be remounted. |
| Inactive | **4** | The crawl component has been deactivated because query topology activation is in progress. |
| DisableForRemove | **5** | The crawl component is disabled and has been taken out of crawls. It MUST NOT be automatically reactivated. |

### 2.2.1.8 Topology Activation Action State

A **Topology Activation Action State** defines the state of a **topology activation action**. The value MUST be an integer listed in the following table:

| Symbolic Name | Value | Description |
|---|---|---|
| NotStarted | **0** | The topology activation action hasn't been started yet. |
| InProgress | **1** | The topology activation action is in progress. |
| Finished | **2** | Execution of the topology activation action has been finished. |
| Aborted | **3** | Topology activation action has been aborted because topology activation was canceled. |

### 2.2.1.9 Refactoring Task State

A **Refactoring Task State** defines the state of a **refactoring task**. The value MUST be an integer listed in the following table:

| Symbolic Name | Value | Description |
|---|---|---|
| NotStarted | **0** | The refactoring task hasn't been started yet. |

| Symbolic Name | Value | Description |
|---|---|---|
| InProgress | **1** | The refactoring task is in progress. |
| Completed | **3** | Execution of the refactoring task has finished. |

### 2.2.1.10 Refactoring Task Type

A **Refactoring Task Type** defines the type of a refactoring task. The value MUST be a string listed in the following table:

| Value | Description |
|---|---|
| **"PropertyStoreCopy"** | Metadata Index Copy refactoring task. Used to copy data from one metadata index to another. |
| **"PropertyStoreDelete"** | Metadata Index Delete refactoring task. Used to delete data from a metadata index. |
| **"CrawlStoreMove"** | Crawl Store Move refactoring task. Used to move data from one crawl store to another. |

### 2.2.1.11 Refactoring Task Batch State

A **Refactoring Task Batch State** defines state of a **refactoring task batch**. The value MUST be an integer listed in the following table:

| Symbolic Name | Value | Description |
|---|---|---|
| NotStarted | **0** | The refactoring task batch hasn't been started yet. |
| InProgress | **1** | The refactoring task batch is in progress. |
| Finished | **2** | Execution of the refactoring task batch has been finished. |

### 2.2.1.12 Component Type

A **Component Type** defines the type of a search component. The value MUST be an integer listed in the following table:

| Value | Description |
|---|---|
| **0** | Administration component of type 0 (see Section 2.2.1.1). |
| **1** | Query component (2). |
| **2** | Crawl component. |
| **3** | Administration component of type 1 (see Section 2.2.1.1). |

### 2.2.1.13 Crawl Store Type

A **Crawl Store Type** defines a type of a crawl store. The value MUST be an integer listed in the following table:

| Symbolic Name | Value | Description |
|---|---|---|
| **NonDedicated** | **0** | Non-dedicated crawl store. Newly discovered **host names** will be added to this crawl store. |
| **Dedicated** | **1** | Dedicated crawl store. Newly discovered host names won't be added to this crawl store. |

#### 2.2.1.14 Index Type

An **Index Type** specifies whether the crawl URL can be returned in search results. The value MUST be an integer listed in the following table:

| Value | Description |
|---|---|
| **0** | The item cannot be returned in search results. |
| **1** | The item can be returned in search results. |

#### 2.2.1.15 Delete Status

A **Delete Status** specifies the crawl URL deletion status. The value MUST be an integer listed in the following table:

| Value | Description |
|---|---|
| **0** | The item is active. |
| **1** | The item is marked for deletion, but the contents of the item are not yet deleted. |
| **2** | The contents of the item are deleted and the item is yet to be removed. |

#### 2.2.1.16 Delete Reason Type

A **Delete Reason Type** specifies the delete reason for the items deleted from the crawl URL history. The value MUST be an integer listed in the following table:

| Value | Description |
|---|---|
| **1** | The item returned an error in the crawl that was marked delete. |
| **2** | The item was deleted because of a **delete crawl**. |
| **3** | The item was deleted as it was not discovered in the **full crawl**. |
| **4** | The item was deleted as it was not discovered when the parent item was crawled in the **incremental crawl**. |
| **5** | The item that supports incremental crawl based on change log was deleted as it was not discovered when the parent item was crawled in the incremental crawl. |
| **6** | The item was deleted as the **parent item** was deleted. |
| **7** | The item was deleted as it was excluded by a **crawl rule**. |

| Value | Description |
|---|---|
| **8** | The item was deleted as there was a delete change for this item in the **change log**. |

#### 2.2.1.17 Link Type

A **Link Type** is the type of the link between items. It MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **10** | Anchor link |
| **393** | **User profile** link |

### 2.2.2 Bit Fields and Flag Structures

The following subsections define the bit fields and flag structures for this specification.

#### 2.2.2.1 End Path Flag

An **End Path Flag** is a bitmask that specifies whether the URL ends with a "/". Its value MUST be from the combination of the flags in the following table:

| Value | Description |
|---|---|
| **0x0001** | The access URL ends with a slash. |
| **0x0002** | The display URL ends with a slash. |

### 2.2.3 Binary Structures

The following subsections define the binary structures for this specification.

#### 2.2.3.1 Refactored Full-Text Index Catalog

This structure is a variant of the full-text index component structure specified in [MS-CIFO] section 2.17. The file formats for each individual file are appropriate for its file extension, as specified in [MS-CIFO] section 2.17, but the parts of the file names before the extensions are generated using different rules.

The file names of this variant are generated via the following **Augmented Backus-Naur Form (ABNF)** rules:

```
index-name = regular-base-name ".ci"

index-directory-name = regular-base-name ".dir"

basic-scopes-name = regular-base-name ".bsi"

basic-scopes-directory-name = regular-base-name ".bsd"

compound-scopes-name = regular-base-name ".00000001.csi"

compound-scopes-directory-name = regular-base-name ".00000001.csd"
```

```
wid-set-name = "01" new-partition-count "0001.wid"

wid-set-bitmap-name = "01" new-partition-count "0001.wsb"

regular-base-name = partition-ordinal new-partition-count "0001"

partition-ordinal = HEXDIG HEXDIG

new-partition-count = HEXDIG HEXDIG
```

Where `HEXDIG` refers to any hexadecimal digit.

The structure MUST contain files with exactly one of each of the following names defined previously:

- `index-name`
- `index-directory-name`
- `basic-scopes-name`
- `basic-scopes-directory-name`
- `compound-scopes-name`
- `compound-scopes-directory-name`
- `wid-set-name`

The structure MAY contain one file with the name defined previously:

- `wid-set-bitmap-name`

The exact criteria for inclusion of the `wid-set-bitmap-name` file in the set are specified in [MS-CIFO] section 2.17.1, though for the purposes of this protocol, it is sufficient just to look for the existence of a file by that name, in the same directory where the other files are found.

Each name in the set of files is generated from the same values of `partition-ordinal` and `new-partition-count`.

### 2.2.4 Result Sets

The following subsections define the result sets for this specification.

#### 2.2.4.1 Crawl Component Result Set

The Crawl Component Result set returns a list of crawl components. Each row specifies a separate crawl component (see Crawl Component Set in section 3.1.1.3) or contains zero rows if the requested component(s) does not exist.

The **Transact-Structured Query Language (T-SQL)** syntax for the result set is as follows:

```
CrawlComponentNumber          int NOT NULL,
CrawlComponentID              uniqueidentifier NOT NULL,
ServerName                    nvarchar(256) NOT NULL,
ServerID                      uniqueidentifier NULL,
LocalStoragePath              nvarchar(260) NOT NULL,
Master                        int NOT NULL,
CrawlStoreID                  uniqueidentifier NOT NULL,
DesiredState                  int NOT NULL,
```

```
DesiredStateSetTime          datetime NOT NULL,
State                        int NOT NULL,
ReportTime                   datetime NOT NULL,
ScopeCompilationID           int NOT NULL;
```

**CrawlComponentNumber:** The unique integer identifier of the crawl component.

**CrawlComponentID:** The unique identifier of the crawl component.

**ServerName:** The name of the server where the crawl component is located.

**ServerID:** The unique identifier of the server where the crawl component is located.

**LocalStoragePath:** The local storage path for the crawl component.

**Master:** MUST be set to **1** if the crawl component is a **master crawl component**; otherwise, it MUST be set to **0**.

**CrawlStoreID:** The unique identifier of the crawl store this crawl component is associated with.

**DesiredState:** The desired state of the crawl component. The value MUST be a Crawl Component State data type as specified in section 2.2.1.7.

**DesiredStateSetTime:** The **Coordinated Universal Time (UTC)** time when the DesiredState was set.

**State:** The current state of the crawl component. The value MUST be a Crawl Component State data type as specified in section 2.2.1.7.

**ReportTime:** The **UTC** time when the crawl component was reported as alive.

**ScopeCompilationID:** The **search scope compilation identifier** of the **search catalog** "Portal_Content" (see [MS-SQLPGAT2]) of the query component (2).

### 2.2.4.2 Query Component Result Set

The **Query Component Result Set** contains information about query components (2). The result set MUST contain zero or more rows with each row corresponding to a single query component.

The T-SQL syntax for the result set is as follows:

```
QueryComponentNumber         int NOT NULL IDENTITY(0, 1),
QueryComponentID             uniqueidentifier NOT NULL PRIMARY KEY CLUSTERED,
ServerName                   nvarchar(256) NOT NULL,
ServerID                     uniqueidentifier,
LocalStoragePath             nvarchar(260) NOT NULL,
PartitionID                  uniqueidentifier NOT NULL,
DesiredState                 int NOT NULL,
DesiredStateSetTime          datetime NOT NULL,
HotSwap                      int,
ShareName                    nvarchar(260),
UsesCustomShare              int,
State                        int NOT NULL,
LastPropagationTime          datetime,
TransitionStep               int,
TransitionStepStartTime      datetime,
TransitionStatus             int,
```

```
TransitionError              nvarchar(2048),
SourceComponentID            uniqueidentifier,
SourceComponentPath          nvarchar(260),
PauseRequested               int,
SettingsInRegistry           int default 0,
ScopeCompilationID           int,
TransitionSequenceName       nvarchar(260),
TransitionCancelRequested    int,
OfflineReason                int;
```

**QueryComponentNumber:** The unique integer identifier of the query component (2).

**QueryComponentID:** The unique identifier of the query component (2).

**ServerName:** The name of the server where the query component (2) is located.

**ServerID:** The unique identifier of the server where the query component (2) is located.

**LocalStoragePath:** The local storage path for the query component (2).

**PartitionID:** The unique identifier of the index partition this query component (2) is associated with.

**DesiredState:** The desired state of the query component. The value MUST be a Query Component State data type as specified in Section 2.2.1.3.

**DesiredStateSetTime:** The UTC time when the **DesiredState** of the query component was changed.

**HotSwap:** The type of the query component (2). The value must be a Query Component Type data type as specified in section 2.2.1.4.

**ShareName:** The name of the shared folder used by this query component (see [MS-CIPROP2]).

**UsesCustomShare:** If set to **1** then the query component (2) MUST use a custom name for the shared folder that is used to copy the full-text index catalog to that query component (2) (see [MS-CIPROP2]). The name of the shared folder is specified with the ShareName column. If set to **0** then the default shared folder name MUST be used by the query component (see Section 3.1.1.2).

**State:** The current state of the query component (2). The value MUST be a Query Component State data type as specified in Section 2.2.1.3.

**LastPropagationTime:** The UTC time when the full-text index catalog on that query component was updated (see [MS-CIPROP]). This value MUST be set to **NULL** if the index has never been updated on that query component.

**TransitionStep:** The number of the current step in the current query component transition sequence. The value MUST be set to **NULL** or **-1** if the query component is not executing a query component transition sequence.

**TransitionStepStartTime:** The UTC time when the current transition step was updated. The value MUST be set to **NULL** if the query component is not executing a query component transition sequence.

**TransitionStatus:** The status of the current or most recently executed query component transition sequence. The value MUST be a Query Component Transition Status data type as specified in Section 2.2.1.5.

**TransitionError:** The error message for the error that occurred during execution of the query component transition sequence.

**SourceComponentID:** The unique identifier of the query component (2) that contains index files in which the given component is going to be initialized.

**SourceComponentPath:** The **Application directory** that contains the full-text index catalog that will be used to initialize or recover the full-text index catalog of the query component. By default this value should be set to **NULL** for newly query created components.

**PauseRequested:** MUST be set to **1** if the component is in a state that requires a pause of the search service application; otherwise, it MUST be set to **0**.

**SettingsInRegistry:** MUST be set to **0**. Client MUST ignore this value.

**ScopeCompilationID:** The search scope compilation identifier of the search catalog "Portal_Content" (see [MS-SQLPGAT2]) of the query component (2). This value MUST be set to **NULL** for newly created query components (2).

**TransitionSequenceName:** The name of the current or most recently executed query component transition sequence of the query component (2).

**TransitionCancelRequested:** The cancelation status of the query component transition sequence the query component (2) is currently executing. If the cancelation of the current query component transition sequence has been requested this value MUST be set to **1**; otherwise, it MUST be set to either **0** or **NULL**.

**OfflineReason:** MUST be set to **0**. Client MUST ignore this value.

### 2.2.4.3 Refactoring Task Batches Result Set

The **Refactoring Task Batches Result Set** contains information about refactoring task batches. Each row in this result set corresponds to a single refactoring task batch.

The T-SQL syntax for the result set is as follows:

```
BatchID                int NOT NULL,
TaskID                 int NOT NULL,
StartDocID             int NOT NULL,
EndDocID               int NOT NULL,
ServerName             nvarchar(256) NOT NULL,
AssignedTime           datetime NULL,
State                  smallint NOT NULL,
StartedTime            datetime NULL,
HeartbeatTime          datetime NULL,
FinishedTime           datetime NULL,
LastErrorDescription   nvarchar(1024) NULL,
LastErrorTime          datetime NULL,
ErrorCount             int NOT NULL,
NumOfDocs              int NOT NULL;
```

**BatchID:** The unique identifier of the refactoring task batch.

**TaskID:** The unique identifier of the refactoring task this refactoring task batch is a part of.

**StartDocID:** The beginning of the interval of document identifiers that defines a set of documents that need to be processed by this refactoring task batch. If the type of the refactoring task is set to

"CrawlStoreMove" then this field can be set to **-1**. If it is set to **-1**, this batch corresponds to the steps that need to be performed to finish the refactoring task (see Section 3.2.5.4).

**EndDocID:** The end of the interval of document identifiers that defines the set of documents that need to be processed by this refactoring task batch. This value is set to -1 if and only if StartDocID is set to **-1**.

**ServerName:** The name of the server the refactoring task batch is assigned to.

**AssignedTime:** The time the refactoring task batch was assigned.

**State:** The state of the refactoring task batch. The value MUST be a Refactoring Task Batch State data type as specified in Section 2.2.1.11.

**StartedTime:** The time when execution of this refactoring task batch started. This value MUST be set to **NULL** if the execution of this refactoring task batch has not started.

**HeartbeatTime:** The UTC time when the server that executes this refactoring task batch reported status of the batch. This value MUST be set to **NULL** if the execution of this refactoring task batch has not started.

**FinishedTime:** The date and time when execution of this refactoring task batch finished. This value MUST be set to **NULL** if the execution of this refactoring task batch has not finished.

**LastErrorDescription:** Text description of the last error that occurred during the execution of this refactoring task batch.

**LastErrorTime:** The time when the most recent error with this batch occurred.

**ErrorCount:** The number of unsuccessful attempts to execute this refactoring task batch.

**NumOfDocs:** MUST be set to "-1" for refactoring task batches created for a refactoring task of type "PropertyStoreCopy" or "PropertyStoreDelete". If the type of the refactoring task this refactoring task batch is associated with is set to "CrawlStoreMove", then this field MUST contain the number of documents being copied by this refactoring task batch.

## 2.2.5 Tables and Views

### 2.2.5.1 MSSAnchorChangeLog

The **MSSAnchorChangeLog** table stores the documents whose **anchors** MUST be processed during the **anchor crawl**. It is used in the implementation of the Anchor Text Information as described in [MS-SQLPGAT2] section 3.1.1.8.

The T-SQL syntax for the table is as follows:

```
TABLE MSSAnchorChangeLog(
    CrawlId             int NOT NULL,
    TargetDocId         int NOT NULL,
    ChangeType          int NOT NULL
);
```

**CrawlId**: A unique identifier of the crawl.

**TargetDocId**: The **document identifier(1)** whose anchors from other documents were modified in the crawl.

**ChangeType**: A number that specifies whether there are any anchors from other documents to the @TargetDocId. It MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| 1 | No anchor points to the @TargetDocId. |
| 2 | One or more anchors point to the @TargetDocId. |

### 2.2.5.2 MSSAnchorText

The **MSSAnchorText** table stores the link information for all the URLs discovered during crawls. It is used in the implementation of Anchor Text Information as specified in [MS-SQLPGAT2] section 3.1.1.8.

The T-SQL syntax for the table is as follows:

```
TABLE MSSAnchorText(
    SourceDocID               int NULL,
    TargetDocID               int NULL,
    LinkHash                  int NULL,
    Link                      nvarchar(4000) NULL,
    LCID                      int NULL,
    LinkId                    bigint IDENTITY(1,1) NOT NULL,
    AnchorText                nvarchar(1024) NULL,
    AnchorHash                int NOT NULL,
    CrawlID                   int NOT NULL,
    LinkOrdinal               int NOT NULL,
    Pid                       int NOT NULL,
    SourceDocSiteID           uniqueidentifier NULL,
    InterSite                 int NOT NULL,
    HostID                    int NOT NULL
);
```

**SourceDocID**: The document identifier(1) of the item that contains the link.

**TargetDocID**: The document identifier(1) of the item to which the link points.

**LinkHash**: The **CRC** hash of the @Link.

**Link**: The URL in the @SourceDocID that links to the @TargetDocID.

**LCID**: The **language code identifier (LCID)** of the link.

**LinkId**: The unique identifier of the link.

**AnchorText**: The **anchor text** in the @Link.

**AnchorHash**: The CRC hash of the @AnchorText.

**CrawlID**: A unique identifier of the crawl in which the link was discovered.

**LinkOrdinal**: A number indicating the order in which the links were discovered for @SourceDocID. The first link MUST be set to **0**, the second link MUST be set to **1**, and so on.

**Pid**: The item's Link Type. See Link Type defined in section 2.2.1.17

**SourceDocSiteID**: A unique identifier of the site where the link was discovered (see Section 2.2.5.7).

**InterSite**: This MUST be set to **1** if the link points to a different site from @SiteID; otherwise, this MUST be set to **0**.

**HostID**: The identifier of the host name.

### 2.2.5.3 MSSCrawlChangedCommittedDocs

The **MSSCrawlChangedCommittedDocs** table stores all the documents committed in the crawl for which Crawl Log Error Level is not set to 2 (see [MS-SQLPADM2]).

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlChangedSourceDocs(
    CrawlId             int NOT NULL,
    DocId               int NOT NULL
);
```

**CrawlId**: A unique identifier of the crawl

**DocId**: The document identifier(1) of the document committed in the crawl for which Crawl Log Error Level is not set to 2.

### 2.2.5.4 MSSCrawlChangedDeletedDocs

The **MSSCrawlChangedDeletedDocs** table stores all the documents deleted in the crawl.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlChangedSourceDocs(
    CrawlId             int NOT NULL,
    DocId               int NOT NULL
);
```

**CrawlId**: A unique identifier of the crawl.

**DocId**: The document identifier(1) of the deleted document.

### 2.2.5.5 MSSCrawlChangedSourceDocs

The **MSSCrawlChangedSourceDocs** table stores all the documents updated during the crawl.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlChangedSourceDocs(
    CrawlId             int NOT NULL,
    DocId               int NOT NULL
);
```

**CrawlId**: A unique identifier of the crawl

**DocId**: The document identifier(1) for the item that was crawled.

### 2.2.5.6 MSSCrawlChangedTargetDocs

The **MSSCrawlChangedTargetDocs** table stores all the document identifiers that the crawled documents point to.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlChangedSourceDocs(
    CrawlId            int NOT NULL,
    DocId              int NOT NULL,
    IsDuplicate        bit NOT NULL
);
```

**CrawlId**: A unique identifier of the crawl

**DocId**: The document identifier(1) containing one or more links from the other crawled documents.

**IsDuplicate**: A bit that MUST be **1** if the document is a duplicate of another document; otherwise, it MUST be **0**.

### 2.2.5.7 MSSCrawlURL

The **MSSCrawlURL** table implements the **crawl URL history** data structure.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlURL (
    DocID                       int NOT NULL,
    StartAddressID              int NOT NULL,
    ContentSourceID             int NOT NULL,
    ProjectID                   int NOT NULL,
    CrawlID                     int NOT NULL,
    CommitCrawlID               int NOT NULL,
    AccessURL                   nvarchar(4000) NOT NULL,
    AccessHash                  int NOT NULL,
    CompactURL                  nvarchar(40) NULL,
    CompactHash                 int NULL,
    DisplayURL                  nvarchar(4000) NOT NULL,
    DisplayHash                 int NOT NULL,
    TransactionFlags            int NOT NULL,
    HostDepth                   int NOT NULL,
    EnumerationDepth            int NOT NULL,
    ParentDocID                 int NOT NULL,
    UseChangeLog                int NOT NULL,
    ChangeLogCookie             varbinary(8000) NULL,
    ChangeLogCookieType         int NOT NULL,
    IndexType                   int NOT NULL,
    MD5                         int NOT NULL,
    PropMD5                     int NOT NULL,
    Retry                       int NOT NULL,
    LastModifiedTime            bigint NOT NULL,
    FolderDelCount              int NOT NULL,
    LCID                        int NOT NULL,
    ParentUpdateCrawlID         int NOT NULL,
    DeletePending               int NOT NULL,
    EndPathFlag                 int NOT NULL,
    HostID                      int NOT NULL,
```

```
    ErrorID                     int NOT NULL,
    ErrorLevel                  int NOT NULL,
    LastTouchStart              datetime NULL,
    ErrorCount                  int NOT NULL,
    ErrorDesc                   nvarchar(512) NULL,
    DocPropsMD5                 bigint NOT NULL,
    CrawlScope                  int NOT NULL,
    RetryCount                  int NOT NULL,
    DocPropsBlob                varbinary(8000) NULL,
    ParentHostID                int NOT NULL,
    LinksBitmap                 int NOT NULL,
    Title                       nvarchar(1500) NULL,
    TitleLCID                   int NULL,
    SecurityUpdateCrawlID       int NOT NULL,
    ReusedId                    bit NOT NULL,
    ProtocolLength              int NOT NULL,
    CachedSecurityUpdateCrawlID int NOT NULL,
    SecurityId                  nvarchar (40) NULL,
    PHFlags                     int NOT NULL,
    SiteID                      uniqueidentifier NULL,
    SecurityUpdateErrorID       int,
    DelayRetryCount             int NOT NULL,
    LogLevel                    int NOT NULL,
    ErrorDeleteCount            int NOT NULL,
    FirstErrorTime              datetime NULL,
    FirstErrorDeleteTime        datetime NULL,
    ErrorSource                 int NOT NULL,
    ChangeLogCookieEnd          varbinary(8000) NULL
);
```

**DocID**: The document identifier(1) of the crawl URL history.

**StartAddressID**: A unique identifier of the **start address**.

**ContentSourceID**: A unique identifier of the **content source**.

**ProjectID**: See Project Identifier defined in [MS-SQLPGAT2] section 2.2.1.1.

**CrawlID**: A unique identifier of the crawl in which this item was last added to **crawl queue**.

**CommitCrawlID**: A unique identifier of the crawl in which this item was crawled.

**AccessURL**: The item's **access URL**.

**AccessHash**: The CRC hash of the @AccessURL string.

**CompactURL**: The item's **compact URL**.

**CompactHash**: The CRC hash of the @CompactURL string.

**DisplayURL**: The item's **display URL**.

**DisplayHash**: The CRC hash of the @DisplayURL string.

**TransactionFlags**: The transaction flags. See Transaction Flags defined in [MS-SQLPGAT2] section 2.2.2.3

**HostDepth**: The number of **host hops** from the start address to this item.

**EnumerationDepth**: The number of **page hops** from the start address to this item.

**ParentDocID**: A unique  identifier of the parent item.

**UseChangeLog**: An integer that MUST be **1** if the item belongs to a site that supports incremental crawl based on a change log; otherwise, it MUST be **0**.

**ChangeLogCookie**: A token that represents the last change that was retrieved from the change log (see [MS-SQLPGAT2]).

**ChangeLogCookieType**: The type of @ChangeLogCookie (see [MS-SQLPGAT2]).

**IndexType**: An integer that MUST be an Index Type data type (section 2.2.1.14) for the item.

**MD5**: The **MD5** hash of the item content.

**PropMD5**: The MD5 hash of the item properties. In the incremental crawl if the value of the parameter is different than the existing value then the item and any child items will be re-crawled.

**Retry**: The number of times the item was tried in the last crawl.

**LastModifiedTime**: The UTC time when the item was modified.

**FolderDelCount**: A **token** that indicates when the last child item was deleted from the current container item.

**LCID**: The LCID.

**ParentUpdateCrawlID**: A unique identifier of the crawl in which the parent item was crawled.

**DeletePending**: An integer that MUST be a Delete Status data type (section 2.2.1.15) for the item.

**EndPathFlag**: An integer that MUST be an End Path Flag data type (section 2.2.2.1) for the item.

**HostID**: The identifier of the host name.

**ErrorID**: A unique identifier for the error if the item was not crawled successfully; otherwise, it MUST be **0**.

**ErrorLevel**: A number which specifies the Crawl Log Error Level defined in [MS-SQLPADM2] section 2.2.1.7.

**LastTouchStart**: The UTC time when the item was crawled.

**ErrorCount**: The number of consecutive times the item @ErrorLevel was in Error.

**ErrorDesc**: An additional error description retrieved by the **index server** while processing the item.

**DocPropsMD5**: The MD5 hash of the item properties.

**CrawlScope**: An integer that be a Transaction Scope [MS-SQLPGAT2] section 2.2.1.15 for the item.

**RetryCount**: The number of times the item was retried in the last crawl.

**DocPropsBlob**: The item properties (see [MS-SQLPGAT2]).

**ParentHostID**: A unique identifier for the host name of the parent item.

**LinksBitmap**: A bitmap of all crawl store identifiers which has crawl URLs linked to this item (see [MS-SQLPGAT2]).

**Title**: The title of the item.

**TitleLCID**: The LCID of the title.

**SecurityUpdateCrawlID**: A unique identifier of the crawl in which only the security of the item was updated for this item.

**ReusedId**: A bit that must be **1** if the document identifier was reused from a deleted crawl URL; otherwise, it MUST be **0**.

**ProtocolLength**: The number of characters before the first occurrence of ":" in the @AccessURL. If ":" is not present in the @AccessURL then this MUST be set to **0**.

**CachedSecurityUpdateCrawlID**: A unique identifier of the crawl in which only the security of the item was updated using the @SecurityId.

**SecurityId**: Security identifier of the item (see [MS-SQLPGAT2]).

**PHFlags**: Flags used by the protocol handler (see [MS-SQLPGAT2]).

**SiteID**: A unique identifier of the site to which the item belongs.

**SecurityUpdateErrorID**: A unique identifier for the error in which only the security of the item was updated; otherwise, it MUST be **0.**

**DelayRetryCount**: The number of times the item failed with an error and MUST be retried in the current crawl.

**LogLevel**: A number which specifies the Crawl Log Level as described in [MS-SQLPADM2] section 2.2.1.8

**ErrorDeleteCount**: An integer representing the number of times the error *"marked as deleted"* was returned during the crawl. (see [MS-SQLPGAT2]).

**FirstErrorTime**: The UTC time of the first crawl error.

**FirstErrorDeleteTime**: The UTC time when the first error *"marked as deleted"* was returned during the crawl.

**ErrorSource**: The unique identifier of the internal search component which has reported the error for the item.

**ChangeLogCookieEnd**: If the item belongs to a site that supports incremental crawl based on the change log, this parameter MAY specify the last change which will be processed by current crawl.

### 2.2.5.8 MSSCrawlURLLog

The **MSSCrawlURLLog** table keeps track of the history of errors encountered in the crawls.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlURLLog(
    TrackID bigint            IDENTITY(1,1) NOT NULL,
    StartAddressID            int NULL,
    ContentSourceID           int NULL,
```

```
    ProjectID               int NULL,
    ErrorID                 int NULL,
    DocID                   int NULL,
    CrawlID                 int NULL,
    AccessURL               nvarchar(4000),
    AccessHash              int NULL,
    DisplayURL              nvarchar(4000),
    DisplayHash             int NULL,
    TransactionType         int NULL,
    Scope                   int NULL,
    TransactionFlags        int NULL,
    HostDepth               int NULL,
    EnumerationDepth        int NULL,
    ParentDocID             int NULL,
    UseChangeLog            int NULL,
    ChangeLogCookie         varbinary(8000) NULL,
    ChangeLogCookieType     int NULL,
    HostID                  int NULL,
    LastTouchStart          datetime NULL,
    ErrorDesc               nvarchar(512),
    LogLevel                int NOT NULL,
    ErrorCount              int NOT NULL,
    ErrorDeleteCount        int NOT NULL,
    FirstErrorTime          datetime NULL,
    FirstErrorDeleteTime    datetime NULL
);
```

**TrackID** : The unique identifier of the **MSSCrawlURLLog** table.

**StartAddressID**: A unique identifier of the start address.

**ContentSourceID**: A unique identifier of the content source.

**ProjectID**: See Project Identifier defined in [MS-SQLPGAT2] section 2.2.1.1.

**ErrorID**: A unique identifier of the error.

**DocID**: The document identifier(1) of the crawl URL history.

**CrawlID**: A unique identifier of the crawl in which this item was last added to crawl queue.

**AccessURL**: The item's access URL.

**AccessHash**: The CRC hash of the **@AccessURL** string.

**DisplayURL**: The item's display URL.

**DisplayHash**: The CRC hash of the **@DisplayURL** string.

**TransactionType**: An integer that MUST be the Transaction type ( [MS-SQLPGAT2] section 2.2.1.14) of the item.

**Scope**: An integer that MUST be the Transaction Scope ([MS-SQLPGAT2] section 2.2.1.15) of the item.

**TransactionFlags**: An integer MUST be the Transaction Flags ([MS-SQLPGAT2] section 2.2.2.3) of the item.

**HostDepth**: An integer representing the number of host hops from the start address to this item.

**EnumerationDepth**: An integer representing the number of page hops from the start address to this item.

**ParentDocID**: A unique identifier of the parent item.

**UseChangeLog**: An integer that MUST be **1** if the item belongs to a site that supports incremental crawl based on a change log; otherwise, it MUST be **0**.

**ChangeLogCookie**: A cookie that represents the last change that was retrieved from the change log (see [MS-SQLPGAT2]).

**ChangeLogCookieType**: An integer that represents the Cookie Type (see [MS-SQLPGAT2]) of @ChangeLogCookie.

**HostID**: The identifier of the host name.

**LastTouchStart**: The UTC time when the item was crawled.

**ErrorDesc**: An additional error description retrieved by the index server while processing the item.

**LogLevel**: An integer representing the Crawl Log Level as specified in [MS-SQLPADM2] section 2.2.1.8

**ErrorCount**: An integer representing the number of consecutive times the item @ErrorLevel had an Error (see Section 2.2.5.7).

**ErrorDeleteCount**: An integer representing the number of times the error *"marked as deleted"* was returned during the crawl.

**FirstErrorTime**: The UTC time of the first crawl error.

**FirstErrorDeleteTime**: The UTC time when the first error *"marked as deleted"* was returned during the crawl.

### 2.2.5.9 MSSCrawlDeletedURL

The **MSSCrawlDeletedURL** table keeps track of the deleted items from the crawl URL history.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlDeletedURL(
    TrackID                 bigint IDENTITY(1,1) NOT NULL,
    StartAddressID          int NOT NULL,
    ContentSourceID         int NULL,
    ProjectID               int NULL,
    DocID                   int NULL,
    CrawlID                 int NOT NULL,
    HisCrawlID              int NULL,
    HisCommitCrawlID        int NULL,
    AccessURL               nvarchar(4000),
    AccessHash              int NOT NULL,
    DisplayURL              nvarchar(4000),
    DisplayHash             int NOT NULL,
    TransactionType         int NULL,
    Scope                   int NULL,
    TransactionFlags        int NULL,
```

```
    HostDepth                int NULL,
    EnumerationDepth         int NULL,
    ParentDocID              int NULL,
    UseChangeLog             int NULL,
    ChangeLogCookie          varbinary(8000) NULL,
    ChangeLogCookieType      int NULL,
    HostID                   int NULL,
    LogTime                  datetime NULL,
    ErrorID                  int NULL,
    ErrorLevel               int NULL,
    DeleteReason             int NULL,
    ProtocolLength           int NOT NULL,
    LogLevel                 int NOT NULL,
    ErrorCount               int NOT NULL,
    ErrorDeleteCount         int NOT NULL,
    FirstErrorTime           datetime NULL,
    FirstErrorDeleteTime     datetime NULL
);
```

**TrackID**: A unique identifier of the deleted item.

**StartAddressID**: A unique identifier of the start address.

**ContentSourceID**: A unique identifier of the content source.

**ProjectID**: See Project Identifier defined in [MS-SQLPGAT2] section 2.2.1.1.

**DocID**: The document identifier(1) of the crawl URL history.

**CrawlID**: A unique identifier of the crawl in which this item was last added to the crawl queue.

**HisCrawlID**: A unique identifier of the crawl in which this item was added to the crawl queue prior to the @CrawlID.

**HisCommitCrawlID**: A unique identifier of the crawl in which this item was crawled prior to the @CrawlID.

**AccessURL**: The item's access URL.

**AccessHash**: The CRC hash of the **@AccessURL** string.

**DisplayURL**: The item's display URL.

**DisplayHash**: The CRC hash of the **@DisplayURL** string.

**TransactionType**: An integer that MUST be a Transaction type ([MS-SQLPGAT2] section 2.2.1.14) for the item.

**Scope**: An integer that MUST be the Transaction Scope ( [MS-SQLPGAT2] section 2.2.1.15 ) for the item.

**TransactionFlags**: An integer that MUST be Transaction Flags ([MS-SQLPGAT2] section 2.2.2.3).

**HostDepth**: An integer representing the number of host hops from the start address to this item.

**EnumerationDepth**: An integer representing the number of page hops from the start address to this item.

**ParentDocID**: A unique identifier of the parent item.

**UseChangeLog**: An integer that MUST be **1** if the item belongs to a site that supports incremental crawl based on a change log; otherwise, it MUST be **0**.

**ChangeLogCookie**: A cookie that represents the last change that was retrieved from the change log (see [MS-SQLPGAT2]).

**ChangeLogCookieType**: The type of @ChangeLogCookie (see [MS-SQLPGAT2]).

**HostID**: The identifier of the host name.

**LogTime**: The UTC time that indicates when the item was deleted from the crawl URL history

**ErrorID**: A unique identifier for the error if an error has occurred; otherwise, **0** if the item was crawled successfully.

**ErrorLevel**: A number which specifies the Crawl Log Error Level defined in [MS-SQLPADM2] section 2.2.1.7

**DeleteReason**: An integer that MUST be a Delete Reason Type (section 2.2.1.16) for the item.

**ProtocolLength**: An integer that MUST be equal the number of characters before the first occurrence of the ":" in the @AccessURL; otherwise, MUST be **0** if the ":" is not present in the @AccessURL.

**LogLevel**: An integer which specifies the Crawl Log Level as specified in [MS-SQLPADM2] section 2.2.1.8

**ErrorCount**: An integer representing the number of consecutive times the item's @ErrorLevel had an Error (see Section 2.2.5.7).

**ErrorDeleteCount**: An integer representing the number of times the error *"marked as deleted"* was returned during the crawl. (see [MS-SQLPGAT2]).

**FirstErrorTime**: The UTC time of the first crawl error.

**FirstErrorDeleteTime**: The UTC time when the first error *"marked as deleted"* was returned during the crawl.

### 2.2.5.10 MSSCrawlHostList

The table **MSSCrawlHostList** implements the **Crawl Host Set** as described in [MS-SQLPADM2] section 3.1.1.3.

```
TABLE MSSCrawlHostList(
    HostID                  int IDENTITY(1,1) NOT NULL,
    HostName                nvarchar(300),
    SuccessCount            int NOT NULL,
    ErrorCount              int NOT NULL,
    WarningCount            int NOT NULL,
    DeleteCount             int NOT NULL,
    LevelHighErrorCount     int NOT NULL
);
```

**HostID**: A unique identifier of the host name.

**HostName**: The host name.

**SuccessCount**: The number of documents that were crawled successfully for the host name.

**ErrorCount**: The number of errors for the host name.

**WarningCount**: The number of warnings for the host name.

**DeleteCount**: The number of deleted items for the host name.

**LevelHighErrorCount**: The number of items with @ErrorLevel = 2 and @LogLevel = 2 for the host name.

#### 2.2.5.11 MSSCrawlHostsLog

The **MSSCrawlHostsLog** table stores the hosts of all the URLs processed in the crawl.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlHostsLog(
    CrawlID            int NOT NULL,
    HostID             int NOT NULL
);
```

**CrawlID**: A unique identifier of the crawl.

**HostID**: A unique identifier of the host name.

#### 2.2.5.12 MSSCrawlLinksLog

The **MSSCrawlLinksLog** table keeps the history of links discovered.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlLinksLog(
    DocID                  int NOT NULL,
    CrawlID                int NOT NULL,
    AccessURL              nvarchar(1500),
    Reason                 int NOT NULL,
    StartAddressID         int NOT NULL,
    SourceDocID            int NOT NULL,
    HostID                 int NOT NULL,
    SourceHostID           int NOT NULL,
    HisStartAddressID      int NOT NULL,
    HisParentDocID         int NOT NULL
);
```

**DocID**: The document identifier(1) of the crawl URL history.

**CrawlID**: A unique identifier of the crawl.

**AccessURL**: The link's access URL.

**Reason**: An integer representing the type of the link. It MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **1** | The link is for start address. |
| **2** | The link is for a different host. |
| **3** | The link is for a different host and different protocol but the parent is not a start address. |
| **4** | Any type of link that is not mentioned earlier. |

**StartAddressID**: A unique identifier of the start address.

**SourceDocID**: The document identifier(1) that discovered this link.

**HostID**: The identifier of the host.

**SourceHostID**: The host identifier of the parent item.

**HisStartAddressID**: An integer representing the previous start address identifier of the link; otherwise, it MUST be **0** if the link was discovered the first time.

**HisParentDocID**: An integer representing the previous document identifier of the parent item; otherwise, it MUST be **0** if the link was discovered the first time.

### 2.2.5.13 MSSCrawlQueue

The **MSSCrawlQueue** table implements the crawl queue data structure.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlQueue(
    SeqID              bigint IDENTITY(1,1) NOT NULL,
    CrawlID            int NOT NULL,
    StartAddressID     int NOT NULL,
    DocID              int NOT NULL,
    TransactionType    int NOT NULL,
    Scope              int NOT NULL,
    TransactionFlags   int NOT NULL,
    HostDepth          int NOT NULL,
    EnumerationDepth   int NOT NULL,
    SourceDocID        int NOT NULL,
    ChangeLogBatchID   int NOT NULL,
    BatchID            bigint NOT NULL,
    ContentSourceID    int NULL,
    ProjectID          int NOT NULL,
    DeleteReason       int NULL,
    ComponentID        int NULL,
    CachedBlob         varbinary(8000) NULL
);
```

**SeqID**: The unique identifier of the **MSSCrawlQueue**.

**CrawlID**: A unique identifier of the crawl.

**StartAddressID**:  A unique identifier of the start address.

**DocID**: The document identifier(1) of the crawl URL history.

**TransactionType**: An integer that MUST be a Transaction type ([MS-SQLPGAT2] section 2.2.1.14) for the item.

**Scope**: An integer that MUST be a Transaction Scope ([MS-SQLPGAT2] section 2.2.1.15) for the item.

**TransactionFlags**: An integer that MUST be the Transaction Flags ([MS-SQLPGAT2] Section 2.2.2.3) for the item.

**HostDepth**: An integer that is the number of host hops from the start address to this item.

**EnumerationDepth**: An integer that is the number of page hops from the start address to this item.

**SourceDocID**: The document identifier(1) of the parent item.

**ChangeLogBatchID**: The identifier of the subset of the change log to which the current item belongs.

**BatchID**: A unique identifier of batch where this document belongs; otherwise, MUST be **0** if the item does not belong to any batches.

**ContentSourceID**: A unique identifier of the content source.

**ProjectID**: An integer that MUST be a Project Identifier as specified in [MS-SQLPGAT2] section 2.2.1.1.

**DeleteReason**: An integer MUST be Delete Reason Type ( section 2.2.1.16 ) for the item.

**ComponentID**: A unique identifier of the crawl component.

**CachedBlob**: MAY contain additional information about the item when it was discovered.

**UseSecurityInfo**: An integer that MUST be set to **1** if the document @DocId has a valid @SecurityID in the **MSSCrawlURL** table; otherwise, this MUST be **0**.

### 2.2.5.14 MSSCrawlUrlReport

The **MSSCrawlURLReport** table stores the results of the crawl for display URLs.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlUrlReport(
    Protocol           nvarchar(15) NULL,
    DisplayURL         nvarchar(450) NOT NULL,
    DisplayURLTail     nvarchar(3550) NULL,
    DocID              int NOT NULL,
    CrawlID            int NOT NULL,
    IsDeleted          bit NOT NULL,
    ContentSourceID    int NOT NULL,
    ErrorID            int NOT NULL,
    ErrorLevel         int NOT NULL,
    TimeStamp          datetime NULL,
    HostID             int NOT NULL,
    DeleteReason       int NULL,
    ErrorDesc          nvarchar(512) NULL,
    LogLevel           int NOT NULL
```

```
);
```

**Protocol:** The prefix string of @DisplayURL from the **MSSCrawlURL** table for the matching @DocId until the length @ProtocolLength. This parameter will be truncated to 15 characters if the length is greater than 15.

**DisplayUrl** : The suffix string of @DisplayUrl from the **MSSCrawlURL** table for the matching @DocId that starts at position @ProtocolLength+1. This parameter will be truncated to 450 characters if the length is greater than 450 and the truncated part is stored in the @DisplayUrlTail.

**DisplayUrlTail**: The truncated part from the @DisplayURL.

**DocID**: The document identifier(1) of the crawled link.

**CrawlID**: A unique identifier of the crawl.

**IsDeleted**: This MUST be set to **1** if the document was deleted from the search result; otherwise, this MUST be set to **0**.

**ContentSourceID**: A unique identifier of the content source.

**ErrorID**: A unique identifier for the error, or **0** if the item was crawled successfully.

**ErrorLevel**: An integer which specifies the Crawl Log Error Level as specified in [MS-SQLPADM2] section 2.2.1.7

**TimeStamp**: The UTC time when the item was crawled.

**HostID**: A unique identifier of the host name.

**DeleteReason**: An integer that MUST be Delete Reason Type data type (section 2.2.1.16)

**ErrorDesc**: An additional error description retrieved by the index server while processing the item.

**LogLevel**: An integer which specifies the Crawl Log Level as specified in [MS-SQLPADM2] section 2.2.1.8

### 2.2.5.15 MSSAnchorPendingChangeLog

The **MSSAnchorPendingChangeLog** table stores the document identifiers whose anchors from other documents are modified in the crawl. This table is used to populate the M**SSAnchorChangeLog**.

The T-SQL syntax for the table is as follows:

```
TABLE MSSAnchorPendingChangeLog(
    CrawlId            int NOT NULL,
    TargetDocId        int NOT NULL
);
```

**CrawlId**: A unique identifier of the crawl.

**TargetDocId**: The document identifier(1) whose anchors from other documents are modified in the crawl.

### 2.2.5.16 MSSAnnotationsPending

The **MSSAnnotationsPending** table stores the details of which links are clicked or skipped in the search results.

The T-SQL syntax for the table is as follows:

```
TABLE MSSAnnotationsPending(
    Pid                     int NOT NULL,
    TargetDisplayURL        nvarchar(4000) NOT NULL,
    TargetDisplayHash       int NOT NULL,
    LCID                    int NULL,
    AnnotationText          nvarchar(1024) NULL,
    AnnotationNumeric       int NOT NULL,
    RecordId                int IDENTITY(-10,-1) NOT NULL,
    UpdateTime              datetime NOT NULL
);
```

**Pid**: The type of the annotation. It MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| 100 | The search result was for a specified **query text** @AnnotationText. The @AnnotationNumeric MUST contain the number of times the link was clicked for that query. |
| 306 | The @AnnotationNumeric MUST contain the number of times the link was clicked for any query text. |
| 307 | The @AnnotationNumeric MUST contain the number of times the link was skipped for any query text. |

**TargetDisplayURL**: The display URL of the link in the search result.

**TargetDisplayHash**: The CRC hash of the @TargetDisplayURL.

**LCID**: The LCID of the link in the search result.

**AnnotationText**: If @Pid is **100** then this MUST be set to the query text; otherwise, this value MUST be ignored.

**AnnotationNumeric**: A number which holds a different value for @Pid as specified in the @Pid definition.

**RecordId**: A unique identifier of the **MSSAnnotationsPending** table.

**UpdateTime**: The UTC time when the details about the search results were retrieved.

### 2.2.5.17 MSSTranTempTable1

The **MSSTranTempTable1** table temporarily keeps track of the **colleague** relationship between two user profile users during the crawl.

The T-SQL syntax for the table is as follows:

```
TABLE MSSTranTempTable1(
    CrawlID                 int NOT NULL,
```

```
    SourceDocID              int NOT NULL,
    SourceUserID             uniqueidentifier NOT NULL,
    TargetUserID             uniqueidentifier NOT NULL,
    Pid                      int NOT NULL,
    LinkID                   int IDENTITY(1,1) NOT NULL
);
```

**CrawlID**: A unique identifier of the crawl.

**SourceDocID**: The document identifier(1) of the item where the colleague relationship originated.

**SourceUserID**: The unique identifier of the user who owns the user profile corresponding to the document identifier @SourceDocId (see [MS-UPSPROF2]).

**TargetUserID**: The user profile user's unique identifier for the colleague of the user profile user @SourceUserID when @Pid is greater than 0.

**Pid**: The **privacy level** of the colleague. It MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| -1 | Indicates the end of colleague relationship for the user profile user @SourceUserID. This link MUST be ignored. |
| 394 | A public colleague. Other users can see this colleague. |
| 395 | A private colleague. Other users do not see this colleague. |

**LinkID**: A unique identifier for the colleague relationship between two user profile users.

### 2.2.5.18 MSSTranTempTable0

The **MSSTranTempTable0** table implements the Links data structure specified in [MS-SQLPGAT2] section 3.1.1.5.

The T-SQL syntax for the table is as follows:

```
TABLE MSSTranTempTable0(
    CrawlID                  int NOT NULL,
    SourceDocID              int NOT NULL,
    DocID                    int NOT NULL,
    StartAddressID           int NOT NULL,
    ContentSourceID          int NOT NULL,
    ProjectID                int NOT NULL,
    AccessURL                nvarchar(4000),
    AccessHash               int NOT NULL,
    CompactURL               nvarchar(40),
    CompactHash              int NULL,
    ParentCompactURL         nvarchar(40),
    ParentCompactHash        int NULL,
    DisplayURL               nvarchar(4000),
    DisplayHash              int NOT NULL,
    Host                     nvarchar(300),
    hrResult                 int NOT NULL,
    AnchorText               nvarchar(512),
    FirstLink                int NOT NULL,
```

```
    TransactionType           int NOT NULL,
    Scope                     int NOT NULL,
    ItemType                  int NOT NULL,
    TransactionFlags          int NOT NULL,
    HostDepth                 int NOT NULL,
    EnumerationDepth          int NOT NULL,
    UseChangeLog              int NOT NULL,
    IndexType                 int NOT NULL,
    ChangeLogBatchID          int NOT NULL,
    SeqID                     bigint NOT NULL,
    LCID                      int NOT NULL,
    EndPathFlag               int NOT NULL,
    PropMD5                   int NOT NULL,
    LastModifiedTime          bigint NOT NULL,
    ProtocolSwitch            int NOT NULL,
    CrawlType                 int NOT NULL,
    HostID                    int NOT NULL,
    SourceHostID              int NOT NULL,
    SourceIsStartAddress      int NOT NULL,
    SourceHostHop             int NOT NULL,
    ErrorID                   int NOT NULL,
    ErrorLevel                int NOT NULL,
    MarkDelete                int NOT NULL,
    AnchorHash                int NOT NULL,
    CachedBlob                varbinary(8000) NULL,
    ParentProcessChangeLog    int NOT NULL,
    Pid                       int NOT NULL,
    SiteID                    uniqueidentifier NULL,
    InterSite                 int NOT NULL,
    ProtocolLength            int NOT NULL,
    LinkID                    bigint IDENTITY(1,1) NOT NULL
);
```

**CrawlID**: A unique identifier of the crawl.

**SourceDocID**: An integer that is the document identifier(1) that discovered this link.

**DocID**: An integer that is the document identifier(1) of the crawl URL history.

**StartAddressID**: A unique identifier of the start address.

**ContentSourceID**: A unique identifier of the content source.

**ProjectID**: An integer that MUST be the Project Identifier ([MS-SQLPGAT2] section 2.2.1.1) of the item.

**AccessURL**: The item's access URL.

**AccessHash**: An integer based CRC hash of the @AccessURL string.

**CompactURL**: The item's compact URL.

**CompactHash**: An integer based identifier of the @CompactURL string.

**ParentCompactURL**: The compact URL of the parent item.

**ParentCompactHash**: An integer based identifier of the @ParentCompactURL.

**DisplayURL**: The item's display URL.

**DisplayHash**: An integer based identifier of the @DisplayURL string.

**Host**: The host name of the link.

**hrResult**: This MUST be 0x80040d07 if the link is an **excluded item**; otherwise, this MUST be **0**.

**AnchorText**: The string value of the anchor text.

**FirstLink**: An integer indicating the order in which the links are discovered for @SourceDocID. For the first link this MUST be set to **0**, for the second link this MUST be set to **1,** and so on.

**TransactionType**: An integer that MUST be the Transaction Type ([MS-SQLPGAT2] section 2.2.1.14) for the item.

**Scope**: An integer that MUST be Transaction Scope ([MS-SQLPGAT2] section 2.2.1.15) for the item.

**ItemType**: An integer representing the type of link that MUST be a value in the following table:

| Value | Description |
|---|---|
| **1** | The link is a start address. |
| **2** | The link was discovered in a **portal content** crawl. |
| **6** | The link was discovered in an anchor crawl. |
| **7** | The last link discovered for @SourceDocID. This link MUST be ignored. |

**TransactionFlags**: An integer that MUST be the Transaction Flags ([MS-SQLPGAT2] section 2.2.2.3) for the item.

**HostDepth**: An integer representing the number of host hops from the start address to this item.

**EnumerationDepth**: An integer representing the number of page hops from the start address to this item.

**UseChangeLog**: An integer that MUST be **1** if the item belongs to a site that supports incremental crawl based on a change log; otherwise, it MUST be **0**.

**IndexType**: An integer that MUST be an Index Type data type (section 2.2.1.14).

**ChangeLogBatchID**: The identifier of the subset of the change log to which the current item belongs.

**SeqID**: The @SeqID of the document in MSSCrawlQueue (section 2.2.5.13) from which this link was discovered.

**LCID**: The LCID.

**EndPathFlag**: An integer that MUST be an End Path Flag data type (section 2.2.2.1).

**PropMD5**: The MD5 hash of the item properties. In the incremental crawl, if the value of this parameter is different than the existing value, the item and any child items will be crawled.

**LastModifiedTime**: The UTC time that indicates when the document was modified. The value MUST be in FILETIME format as defined in [MS-DTYP] section 2.3.1.

**ProtocolSwitch**: An integer that MUST be set to **1** if the **protocol** of the link and the protocol of the parent item was different; otherwise, this MUST be set to **0**.

**CrawlType**: An integer that MUST be the **crawl type** ([MS-SQLPGAT2] section 2.2.1.2) in which the link was discovered.

**HostID**: The identifier of the host name.

**SourceHostID**: The host identifier of the parent item.

**SourceIsStartAddress**: An integer that MUST be set to **1** if the parent item is a start address; otherwise, this MUST be set to **0**.

**SourceHostHop**: An integer that MUST be set to **1** if the @SourceHostID and @HostID are different; otherwise, this MUST be set to **0**.

**ErrorID**: MUST be a unique identifier of the crawl error for the parent item; otherwise, it MUST be **0** when the parent item is crawled successfully.

**ErrorLevel**: An integer which specifies the Crawl Log Error Level ([MS-SQLPADM2] section 2.2.1.7) for the parent item.

**MarkDelete**: An integer that MUST be set to **1** if the @ErrorId is marked as deleted; otherwise, this MUST be set to **0**.

**AnchorHash**: The CRC hash of the @AnchorText.

**CachedBlob**: MAY contain additional information about the item when it was discovered.

**ParentProcessChangeLog**: This MUST be set to **1** if the link is discovered in a change log crawl; otherwise, this MUST be set to **0**.

**Pid**: An integer that MUST be a Link Type data type (section 2.2.1.17).

**UseSecurityInfo**: An integer that MUST be set to **1** if the document @DocId has valid @SecurityID in MSSCrawlUrl table; otherwise, this MUST be set to **0**.

**SiteID**: A unique identifier of the site where the link was discovered.

**InterSite**: An integer that MUST be set to **1** if the link points to a different site from @SiteID; otherwise, this MUST be set to **0**.

**ProtocolLength**: An integer representing the number of characters before the first occurrence of the ":" in the @DisplayURL; otherwise, this MUST be set to **0**.

**LinkID**: A unique identifier of the link.

### 2.2.5.19 MSSUserHosts

The **MSSUserHosts** table stores the host names for the start addresses.

The T-SQL syntax for the table is as follows:

```
TABLE MSSUserHosts(
    HostID              int NOT NULL
);
```

**HostID**: A unique identifier of the host name.

### 2.2.5.20 MSSSocialDistance

The **MSSSocialDistance** table implements the Social Distance Property as specified in [MS-SQLPGAT2] section 3.1.1.11.

The T-SQL syntax for the table is as follows:

```
TABLE MSSSocialDistance (
    SourceDocID      int NULL,
    SourceUserID     uniqueidentifier NULL,
    TargetDocID      int NULL,
    TargetUserID     uniqueidentifier NULL,
    Pid              int NOT NULL,
    LinkID           int NOT NULL
);
```

**SourceDocID**: The document identifier specifying the source user.

**SourceUserID**: The **user profile record identifier** of the source user.

**TargetDocID**: The document identifier specifying the target user.

**TargetUserID**: The user profile record identifier of the target user.

**Pid**: An integer representing the type of relationship that MUST be set to one of the values in the following table:

| Value | Description |
|---|---|
| **394** | Public relationship. |
| **395** | Private relationship. |

**LinkID**: The unique identifier of the corresponding link.

### 2.2.5.21 MSSCrawlReportCrawlErrors

The **MSSCrawlReportCrawlErrors** table contains a list of errors that occurred during the crawl. This table implements the Reported Crawl Error Set defined in [MS-SQLPGAT2] section 3.1.1.16.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlReportCrawlErrors (
    DocID             int NOT NULL,
    CrawlID           int NOT NULL,
    ErrorID           int NOT NULL,
    MarkDelete        int NOT NULL,
    ErrorCount        int NOT NULL,
    FirstErrorTime    datetime NULL,
    ChildrenCount     int NOT NULL,
    Recrawl           bit NOT NULL
);
```

**DocID**: The identifier of the document for which an error was encountered.

**CrawlID**: The identifier of the crawl which reported the error.

**ErrorID**: The identifier of the error.

**MarkDelete**: An integer that MUST be **1** if the document was marked as deleted, but has not yet been deleted from the index; otherwise, it MUST be **0**.

**ErrorCount**: An integer representing the number of errors encountered for the specified document.

**FirstErrorTime**: The UTC time when the first error for the specified document occurred.

**ChildrenCount**: An integer representing the number of child documents of the document for which the error was encountered.

**Recrawl**: An integer that MUST be **1** if the document needs to be crawled again during the next incremental crawl; otherwise, it MUST be **0**.

### 2.2.5.22 MSSCrawlUrlChanges

The **MSSCrawlUrlChanges** table contains a list of documents that were updated during crawl.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlUrlReport  (
    DocID        int NOT NULL,
    TimeStamp    datetime NULL,
    Status       int NOT NULL
);
```

**DocID**: The unique identifier of the document.

**TimeStamp**: The UTC time when the changed document was crawled.

**Status**: An integer that MUST be a Crawl Change Status data type as specified in [MS-SQLPADM2] section 2.2.1.6.

### 2.2.5.23 MSSCrawlUrlUsedContentSourceReport

The **MSSCrawlUrlUsedContentSourceReport** table contains list of host names that were crawled for each content source.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCrawlUrlUsedContentSourceReport (
    ContentSourceID   int NOT NULL,
    HostID            int NOT NULL
);
```

**ContentSourceID**: The identifier of the content source.

**HostID**: The identifier of the host name.

#### 2.2.5.24 MSSCommittedRefactoringBatches

The **MSSCommittedRefactoringBatches** table contains a list of refactoring task batches that have been completed.

The T-SQL syntax for the table is as follows:

```
TABLE MSSCommittedRefactoringBatches (
    BatchID       int NOT NULL
);
```

**BatchID**: The unique identifier of the refactoring task batch.

#### 2.2.5.25 MSSRefactoringStatistics

The **MSSRefactoringStatistics** table contains a list of item counts previously recorded for the specified table.

The T-SQL syntax for the table is as follows:

```
TABLE MSSRefactoringStatistics (
    TableName    nvarchar(256),
    NumOfRows    int
);
```

**TableName**: The name of the table.

**NumOfRows**: The number of rows in that table.

### 2.2.6 XML Structures

This section defines XML structures that are used by the stored procedure described in this document.

The syntax of the definitions in this section use XML Schema as specified in [XMLSCHEMA1] and [XMLSCHEMA2].

#### 2.2.6.1 Namespaces

This protocol specifies and references **XML namespaces** using the mechanisms specified in [XMLNS]. Although this document associates a **XML namespace prefix** for each XML namespace that is used, the choice of any particular XML namespace prefix is implementation-specific and not significant for interoperability.

| Prefix | Namespace URI | Reference |
|---|---|---|
| xs | http://www.w3.org/2001/XMLSchema | [XMLSCHEMA1]<br>[XMLSCHEMA2] |

#### 2.2.6.2 Simple Types

This specification does not define any common XML Schema simple type definitions.

### 2.2.6.3 Complex Types

This specification does not define any common XML Schema complex type definitions.

### 2.2.6.4 Elements

The following table summarizes the set of common XML Schema element definitions defined by this specification. The XML Schema element definitions that are specific to a particular operation are described with the operation.

| Element | Description |
|---|---|
| TaskParts | Represents a set of refactoring task batches. |
| PartitionsMap | Represents mapping between **document distribution identifiers** and index partitions. |

#### 2.2.6.4.1 TaskParts Schema

This is an XML structure that represents a set of refactoring task parts for a refactoring task. It is used in the **proc_MSS_CreateRefactoringTask** stored procedure.

```
<xs:element name="Root">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="Part" minOccurs="1"
            maxOccurs="unbounded" type ="xs:integer" />
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

**Root.Part:** A value of a refactoring task part (see Section 3.1.1.4).

#### 2.2.6.4.2 PartitionsMap Schema

This is an XML structure that represents mapping between the document distribution identifiers and index partitions. It is used in the **proc_MSS_UpdatePartitionsMap** stored procedure (section 3.1.5.93).

```
<xs:element name="PartitionsMap">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="PartitionsMapEntry" type ="xs:custom">
            <xs:complexType>
               <xs:attribute name="Hash" type="xs:integer"
                   use="required"/>
               <xs:attribute name="PartitionID" type="xs:string"
                   use="required"/>
            </xs:complexType>
         </xs:element>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

**PartitionsMap.PartitionsMapEntry.Hash:** A value of a document distribution identifier that MUST be in range from **0** to **255**.

**PartitionsMap.PartitionsMapEntry.PartitionID:** String representation of the unique identifier of the index partition corresponding to the document distribution identifier.

#### 2.2.6.5 Attributes

This specification does not define any common XML Schema attribute definitions.

#### 2.2.6.6 Groups

This specification does not define any common XML Schema group definitions.

#### 2.2.6.7 Attribute Groups

This specification does not define any common XML Schema attribute group definitions.

# 3 Protocol Details

## 3.1 Server Details

### 3.1.1 Abstract Data Model

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

#### 3.1.1.1 Administration Component

The following diagram shows an abstract data model for an administration component. In the diagram, each table specifies a type of entity in the model.

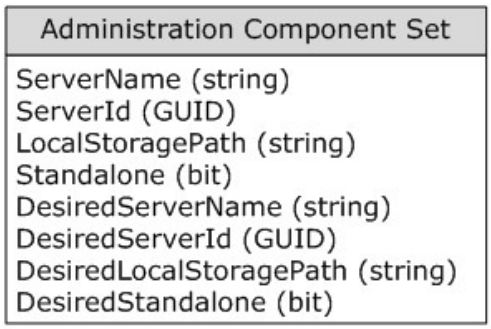


**Figure 2: Administration Component Abstract Data Model**

**Administration Component Set**: Used to store current state of the administration component. The Administration Component Set MUST contain exactly one entry. The entry has the following elements:

- **ServerName**: The name of the server where the administration component is currently located. This parameter MUST be set to **NULL** if the administration component is not initialized.
- **ServerId**: The unique identifier of the server where the administration component is currently located. This parameter MUST be set to **NULL** if the administration component is not initialized.
- **LocalStoragePath**: The current local storage path for the administration component. This parameter MUST be set to **NULL** if the administration component is not initialized.
- **Standalone**: The current type of the administration component. This value MUST be set to **NULL** if the administration component is not initialized; otherwise, the value MUST be an Administration Component Type data type as specified in Section 2.2.1.1.
- **DesiredServerName**: The desired server name for the administration component. If it is different from the ServerName, it defines the name of the server where the administration component needs to be moved to.
- **DesiredServerId**: The unique identifier of the desired server for the administration component. If it is different from the ServerId, it defines the server where the administration component needs to be moved to.

- **DesiredLocalStoragePath**: The desired local storage path for the administration component. This defines a new value for the local storage path when the administration component is being initialized or moved to a different server.
- **DesiredStandalone**: The desired type of the administration component. If this value is not set to **NULL** then it MUST be an Administration Component Type data type as specified in Section 2.2.1.1. This defines a new value for the type of the administration component when the administration component is being initialized or moved to a different server.

### 3.1.1.2 Query Topology

The following diagram shows the abstract data model for a query topology. In the diagram, each table specifies a type of entity in the model, and each arrow specifies that one type of entity that always contains a reference to another.

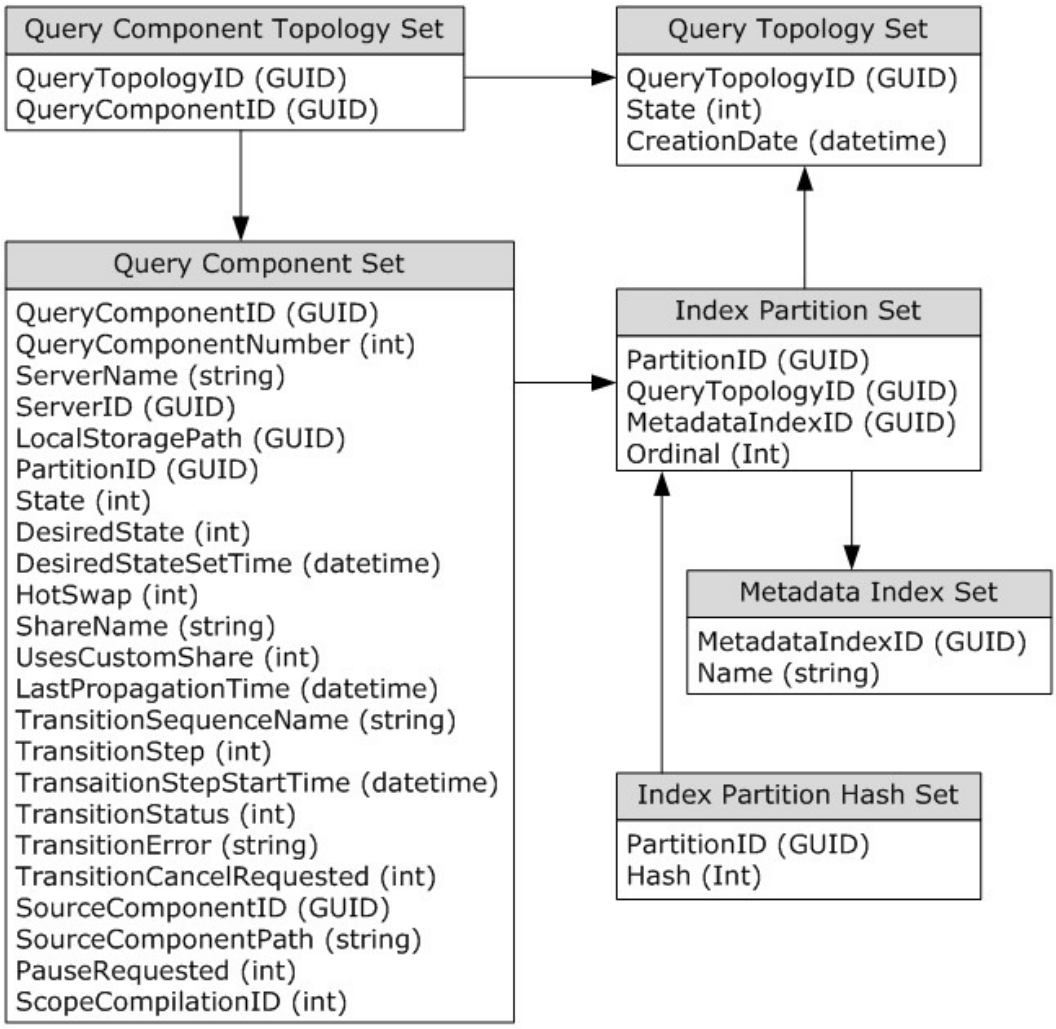


**Figure 3: Query Topology Abstract Data Model**

**Query Topology Set**: A collection of entries corresponding to query topologies. Each entry MUST be uniquely identified by its QueryTopologyID, and it MUST include the following elements:

- **QueryTopologyID**: The unique identifier for the of the query topology.
- **State**: The integer state of the query topology. The value MUST be a Query Topology State data type as specified in section 2.2.1.2. Following table defines allowed state changes for query topologies:

| Old state | New state | Description |
|---|---|---|
| Inactive | Activating | Activation of the query topology started. |
| Activating | Active | Query topology has been activated. State of the previously active query topology, if any, MUST be changed to Deactivating. |
| Activating | Deactivating | Activation of the query topology cancelled. |
| Active | Deactivating | Another query topology has been activated, deactivation of the old query topology started. This state change MUST occur if and only if the state of another query topology is changed from Activating to Active (see preceding information). |
| Deactivating | Inactive | Query topology has been deactivated. |

- **CreationDate**: The UTC time when the query topology was created.

**Metadata Index Set**: A collection of entries corresponding to metadata indexes. Each entry MUST be uniquely identified by its MetadataIndexID, and it MUST include the following elements:

- **MetadataIndexID**: The unique identifier of the metadata index.
- **Name**: The name of the metadata index.

**Index Partition Set**: A collection of entries that is used to store the associations between index partitions and query topologies. Each entry MUST include the following elements:

- **PartitionID**: The unique identifier of the of the index partition. This field is not unique in the Index Partition Set, that is, the same index partition can be associated with multiple query topologies.
- **QueryTopologyID**: The unique identifier of the query topology this index partition belongs to.
- **MetadataIndexID**: The unique identifier of the metadata index that is associated with the index partition.
- **Ordinal**: The integer ordinal of the index partition.

**Index Partition Hash Set**: A collection of entries that is used to store the associations between index partitions and document distribution identifiers. It defines to which index partition each document belongs to, that is, documents with a specific document distribution identifier belong to the index partition that document distribution identifier is associated with. Each entry MUST include the following elements:

- **PartitionID**: The unique identifier of the index partition.
- **Hash**: The document distribution identifier associated with the index partition.

**Query Component Set**: A collection of entries corresponding to the query components (2). Each entry MUST be uniquely identified by its QueryComponentID, and it MUST include the following elements:

- **QueryComponentID**: The unique identifier of the query component (2).
- **QueryComponentNumber**: The integer unique identifier of the query component (2).
- **ServerName**: The name of the server where the query component (2) is located.
- **ServerID**: The unique identifier of the server where the query component (2) is located.
- **LocalStoragePath**: The local storage path for the query component (2).
- **PartitionID**: The unique identifier of the index partition this query component (2) is associated with.
- **State**: The state of the query component (2). The value MUST be a Query Component State data type as specified in section 2.2.1.3.
- **DesiredState**: The desired state of the query component (2). The value MUST be a Query Component State data type as specified in section 2.2.1.3.
- **DesiredStateSetTime**: The last UTC time when the value of DesiredState has been changed.
- **HotSwap**: The type of the query component (2). This property defines the behavior of the query component in the presence of other query components (2) associated with the same index partition. The value must be a Query Component Type data type as specified in section 2.2.1.4.
- **ShareName**: The name of the shared folder used by this query component (2) (see [MS-CIPROP2] section 1.3).
- **UsesCustomShare**: If set to **1** then the query component (2) uses a custom shared folder name stored in the **ShareName** field for the shared folder that is used to copy the full-text index catalog to that query component (2) (see [MS-CIPROP2] section 1.3). If set to **0,** the default shared folder name MUST be used by the query component (2). The default shared folder name is "<searchAppId>-query-<componentId>", where "<searchAppId>" – is the identifier of the search service application, "<componentId>" – is the integer identifier of the query component (2).
- **LastPropagationTime**: The UTC time when the full-text index catalog on that query component (2) was updated. This value MUST be set to **NULL** if the full-text index catalog has never been updated on that query component (2).
- **TransitionSequenceName**: The name of the current or most recently executed query component transition sequence of the query component (2). The value MUST be set to **NULL** or an empty string if the query component is not executing a query component transition sequence.
- **TransitionStep**: The number of the current step in the current query component transition sequence. The value MUST be set to **NULL** or **-1** if the query component is not executing a query component transition sequence.
- **TransitionStepStartTime**: The UTC time when Transition Step was updated. The value MUST be set to **NULL** if the query component (2) is not executing a query component transition sequence.
- **TransitionStatus**: The status of the current or most recently executed query component transition sequence. The value MUST be a Query Component Transition Status data type as specified in section 2.2.1.5.

- **TransitionError**: The error message for the error that occurred during the execution of the current or most recently executed query component transition sequence. If no errors have occurred, this field MUST be set to **NULL**.
- **TransitionCancelRequested**: The cancelation status of the current query component transition sequence. If the cancelation of the current query component transition sequence has been requested this value MUST be set to **1**, otherwise it MUST be set to either **0** or **NULL**.
- **SourceComponentID**: The unique identifier of the source query component (2). The source query component (2) is used to initialize the full-text index catalog on the given query component (2).
- **SourceComponentPath**: The source Application directory that contains the full-text index catalog that will be used to initialize or recover the full-text index catalog of the query component (2). By default this value should be set to **NULL** for newly created query components (2).
- **PauseRequested**: MUST be set to **1** if the component is in a state that requires a pause of the search service application; otherwise, it MUST be set to **0**.
- **ScopeCompilationID**: The search scope compilation identifier of the search catalog "Portal_Content" (see [MS-SQLPGAT2] section 2.2.1.1) of the query component (2). By default this value should be set to **NULL** for newly query created components.

**Query Component Topology Set**: A collection of entries that is used to store an association between query components (2) and query topologies. Each entry MUST include the following elements:

- **QueryTopologyID**: The unique identifier of the query topology.
- **QueryComponentID**: The unique identifier of the query component (2).

### 3.1.1.3 Crawl Topology

The following diagram shows the abstract data model for crawl topology. In the diagram, each table specifies a type of entity in the model, and each arrow specifies that one type of entity always contains a reference to another:

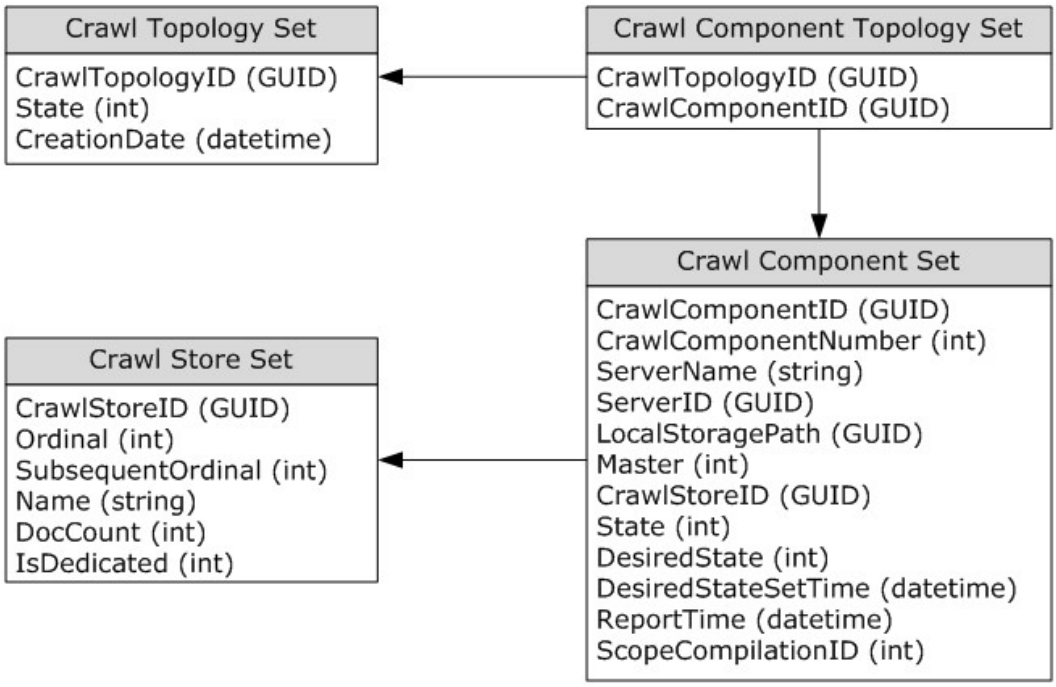


**Figure 4: Crawl Topology Abstract Data Model**

**Crawl Topology Set**: A collection of entries corresponding to crawl topologies. Each entry MUST be uniquely identified by its CrawlTopologyID, and it MUST include the following elements:

- **CrawlTopologyID**: The unique identifier of the crawl topology.
- **State**: The state of the crawl topology. The value MUST be a Crawl Topology State data type as specified in section 2.2.1.6. Following table defines allowed state changes for crawl topologies:

| Old state | New state | Description |
|---|---|---|
| Inactive | Activating | Activation of the crawl topology started. |
| Activating | Active | Crawl topology has been activated. State of the previously active crawl topology, if any, MUST be changed to Deactivating. |
| Activating | Deactivating | Activation of the crawl topology cancelled. |
| Active | Deactivating | Another crawl topology has been activated, deactivation of the old crawl topology started. This state change is MUST occur if and only if the state of another crawl topology is changed from Activating to Active (see preceding information). |
| Deactivating | Inactive | Crawl topology has been deactivated. |

- **CreationDate**: The UTC time when the crawl topology was created.

**Crawl Store Set**: A collection of entries corresponding to crawl stores. Each entry MUST be uniquely identified by the CrawlStoreID and MUST include the following elements:

- **CrawlStoreID**: The unique identifier of the crawl store.

- **Ordinal**: The integer identifier of the crawl store. This value MUST be set to **NULL** for all crawl stores that are not associated with the crawl components in the active crawl topology (that is, the crawl topology that is in the Active state).
- **SubsequentOrdinal**: The subsequent integer identifier for the crawl store. This field is used for a new value of the integer identifier of the crawl store when a crawl topology is being activated.
- **Name**: The name of the crawl store.
- **DocCount**: The total number of documents stored in the crawl store.
- **IsDedicated**: The type of the crawl store. The value MUST be a Crawl Store Type data type as specified in section 2.2.1.13.

**Crawl Component Set**: A collection of entries corresponding to crawl components. Each entry MUST be uniquely identified by its CrawlComponentID, and it MUST include the following elements:

- **CrawlComponentID**: The unique identifier of the crawl component.
- **CrawlComponentNumber**: The integer unique identifier of the crawl component.
- **ServerName**: The name of the server where the crawl component is located.
- **ServerID**: The unique identifier of the server where the crawl component is located.
- **LocalStoragePath**: The local storage path for the crawl component.
- **Master**: MUST be set to **1** if the crawl component is a master crawl component; otherwise, the value MUST be **0**.
- **CrawlStoreID**: The unique identifier of the crawl store with which this crawl component is associated.
- **State**: The state of the crawl component. The value MUST be a Crawl Component State data type as specified in section 2.2.1.7.
- **DesiredState**: The desired state of the crawl component. The value MUST be a Crawl Component State data type as specified in section 2.2.1.7.
- **DesiredStateSetTime**: The UTC time when the DesiredState was set.
- **ReportTime**: The UTC time when the component was reported as alive. If this field is not updated for more than an hour the administration component MUST set the state of this crawl component to *Disabled*.
- **ScopeCompilationID**: The search scope compilation identifier of the search catalog "Portal_Content" (see [MS-SQLPGAT2] section 2.2.1.1) of the query component (2).

**Crawl Component Topology Set**: A collection of entries that is used to store the association between crawl components and crawl topologies. Each entry MUST include the following elements:

- **CrawlTopologyID**: The identifier of the crawl topology.
- **CrawlComponentID**: The identifier of the crawl component.

### 3.1.1.4 Database Repartitioning

The following diagram shows the abstract data model for database repartitioning. In the diagram, each table specifies a type of entity in the model, and each arrow specifies that one type of entity always contains a reference to another.

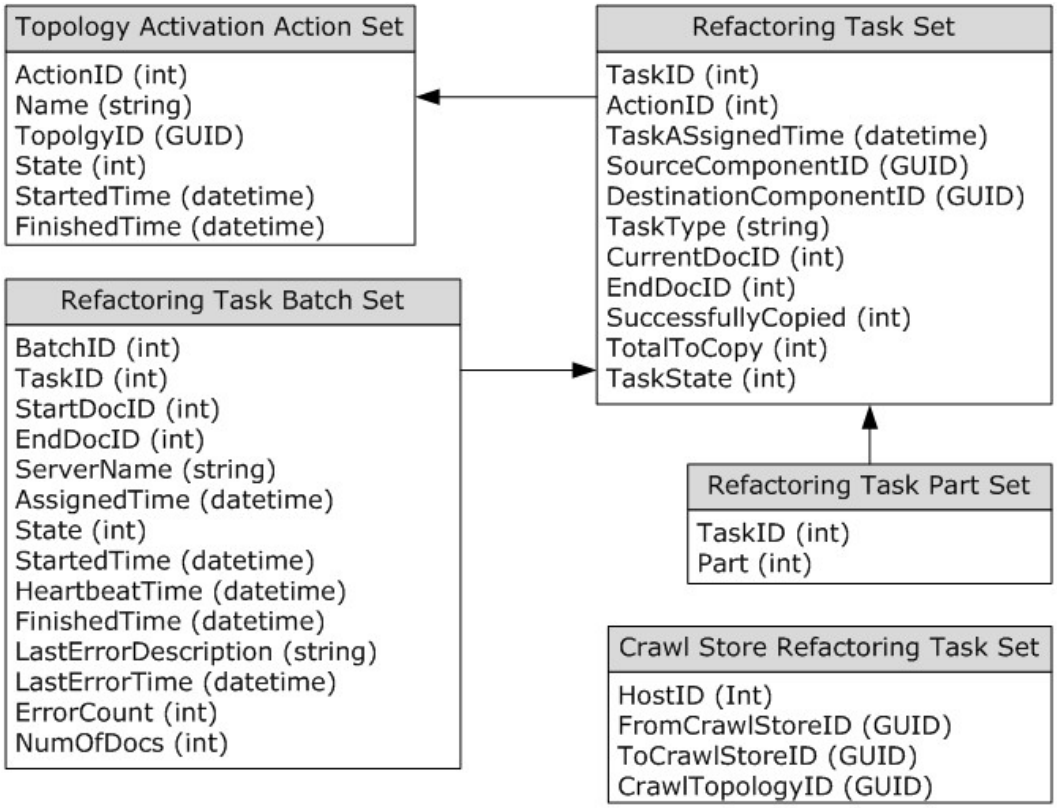


**Figure 5: Database Repartitioning Abstract Data Model**

**Topology Activation Action Set**: A collection of entries corresponding to topology activation actions. Each entry MUST be uniquely identified by its ActionID, and it MUST include the following elements:

- **ActionID**: The unique identifier of the topology activation action.
- **Name**: The name of the topology activation action.
- **TopologyID**: The unique identifier of the query or crawl topology this topology activation action is created for.
- **State**: The state of the topology activation action. The value MUST be a Topology Activation Action State data type as specified in Section 2.2.1.8. Following table defines allowed state changes for topology activation action:

| Old state | New state |
|---|---|
| NotStarted | InProgress |

| Old state | New state |
|---|---|
| InProgress | Finished |
| InProgress | Aborted |
| NotStarted | Aborted |

- **StartedTime**: The UTC time when execution of the topology activation action started. This value MUST be set to **NULL** if the execution has not started yet.
- **FinishedTime**: The UTC time when execution of the topology activation action finished. This value MUST be set to **NULL** if the execution has not finished yet.

**Refactoring Task Set**: A collection of entries corresponding to refactoring tasks. Each entry MUST be uniquely identified by its TaskID, and it MUST include the following elements:

- **TaskID**: The unique identifier of the refactoring task.
- **ActionID**: The unique identifier of the topology activation action this task is a part of.
- **TaskType**: The type of the refactoring task. The value MUST be a Refactoring Task Type data type as specified in Section 2.2.1.10.
- **SourceComponentID**: The meaning of this element is determined by the **TaskType** field for this refactoring task:

| TaskType Value | SourceComponentID |
|---|---|
| "PropertyStoreCopy" | The unique identifier of the metadata index from which data is being copied. |
| "PropertyStoreDelete" | The unique identifier of the metadata index from which data is being deleted. |
| "CrawlStoreMove" | The unique identifier of the crawl store from which data is being moved. |

- **DestinationComponentID**: The meaning of this element is determined by the **TaskType** field for this refactoring task:

| TaskType Value | DestinationComponentID |
|---|---|
| "PropertyStoreCopy" | The unique identifier of the metadata index from which data is being copied. |
| "PropertyStoreDelete" | The DestinationComponentID MUST be set to **NULL**. |
| "CrawlStoreMove" | The unique identifier of the crawl store from which data is being moved. |

- **CurrentDocID**: The document identifier(1) of the document that was last copied for this refactoring task. MUST be set to **-1** if no documents have been copied yet.
- **EndDocID**: The document identifier(1) of the last document that will be copied by this task. MUST be set to **-1** if number of documents in the source database is not known yet.
- **SuccessfullyCopied**: The number of documents that have been successfully processed for this refactoring task.
- **TotalToCopy**: The total number of documents that need to be processed for this refactoring task.

- **TaskState**: The state of the refactoring task. The value MUST be a Refactoring Task State data type as specified in Section 2.2.1.9.

**Refactoring Task Part Set**: A collection of entries that defines the list of refactoring task parts for each refactoring task. Each entry MUST include the following elements:

- **TaskID**: The unique identifier of the refactoring task.
- **Part**: The integer value that defines the refactoring task part. The meaning of this value is determined by the **TaskType** of the refactoring task, and is defined in the following table:

| TaskType Value | Part |
|---|---|
| "PropertyStoreCopy" | The Document distribution identifier for the documents that need to be copied. |
| "PropertyStoreDelete" | The Document distribution identifier for the documents that need to be deleted. |
| "CrawlStoreMove" | The unique identifier of the host name that needs to be moved from one crawl store to another. |

**Refactoring Task Batch Set**: A collection of entries corresponding to refactoring task batches. Each entry MUST be uniquely identified by its BatchID, and it MUST include the following elements:

- **BatchID**: The unique identifier of the refactoring task batch.
- **TaskID**: The unique identifier of the refactoring task this refactoring task batch is a part of.
- **StartDocID**: The beginning of the interval of document identifiers that defines a set of documents that need to be processed by this refactoring task batch. If the type of the refactoring task is set to "CrawlStoreMove" then this field can be set to **-1**. If it is set to **-1** then this batch corresponds to the steps that need to be performed to finish the refactoring task (see Section 3.2.5.4).
- **EndDocID**: The end of the interval of document identifiers that defines the set of documents that need to be processed by this refactoring task batch. This value is set to -1 if and only if StartDocID is set to **-1**.
- **ServerName**: The name of the server the refactoring task batch is assigned to.
- **AssignedTime**: The UTC time the refactoring task batch was assigned.
- **State**: The state of the refactoring task batch. The value MUST be a Refactoring Task Batch State data type as specified in Section 2.2.1.11.
- **StartedTime**: The UTC time when the execution of this refactoring task batch started. This value MUST be set to **NULL** if the execution of this refactoring task batch has not started.
- **HeartbeatTime**: The UTC time when the server that executes this refactoring task batch reported status of the batch. This value MUST be set to **NULL** if the execution of this refactoring task batch has not started.
- **FinishedTime**: The UTC time when the execution of this refactoring task batch finished. This value MUST be set to **NULL** if execution of this refactoring task batch has not finished.
- **LastErrorDescription**: Text description of the last error that occurred during the execution of this refactoring task batch. This field MUST be set to **NULL** if no errors have occurred during the execution of this refactoring task batch.

- **ErrorCount**: The number of unsuccessful attempts to execute this refactoring task batch.
- **NumOfDocs**: MUST be set to **-1** for refactoring task batches created for a refactoring task of type "PropertyStoreCopy" or "PropertyStoreDelete". If the type of the refactoring task this refactoring task batch is associated with is set to "CrawlStoreMove", then this field contains the number of documents being copied by this refactoring task batch.

**Crawl Store Refactoring Task Set**: A collection of entries corresponding to the host names that need to be moved when a new crawl topology is activated. Each entry MUST include the following elements:

- **HostID**: The identifier of the host name that will to be moved from one crawl store to another.
- **FromCrawlStoreID**: The identifier of the source crawl store data will be moved from.
- **ToCrawlStoreID**: The identifier of the source crawl store data will be moved to.
- **CrawlTopologyID**: The identifier of the crawl topology that is being activated.

### 3.1.1.5 Host Distribution Rules

The following diagram describes the abstract data model for a host distribution rules. In the diagram, each table specifies a type of entity in the model, and each arrow specifies that one type of entity always contains a reference to another.

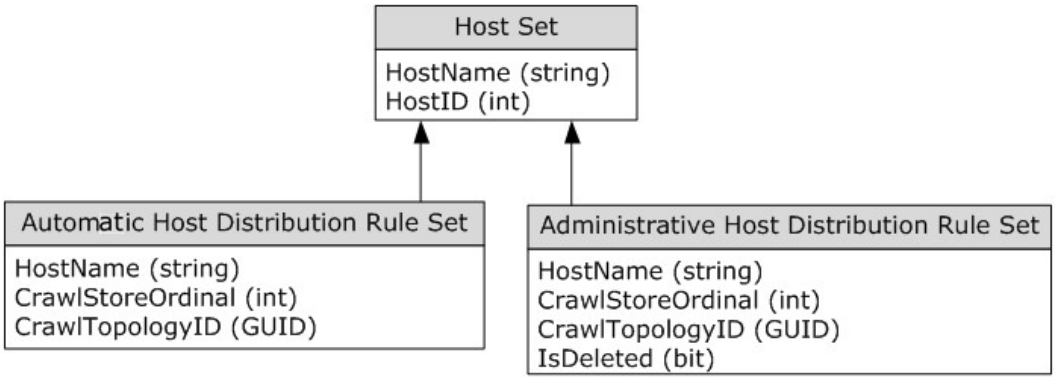


**Figure 6: Host Distribution Rule Abstract Data Model**

**Host Set**: A collection of entries each corresponding to one host name. Each host name MUST be associated with a host distribution rule. Each entry MUST be uniquely identified by the HostID and MUST include the following elements:

- **HostName**: The host name.
- **HostID**: The ordinal identifier for the host name.

**Automatic Host Distribution Rule Set**: A collection of entries each corresponding to one **automatic host distribution rule**. An automatic host distribution rule is a rule that is created by the search application when a host is crawled. Each entry MUST include the following elements:

- **HostName**: The host name for which the automatic host distribution rule applies.

- **CrawlStoreOrdinal**: The ordinal of the crawl store intended to contain crawled documents which reside on the host. The crawl store ordinal references a Crawl Store Set as defined in Section 3.1.1.3.
- **CrawlTopologyID**: The unique identifier of the crawl topology to which the host distribution rule has been applied. If this value is NULL, then it MUST be true that the rule has been added but not yet applied to a topology.

**Administrative Host Distribution Rule Set**: A collection of entries each corresponding to one **administrative host distribution rule**. If an administrative host distribution rule is applied for a crawl topology, there MUST be no automatic host distribution rules for that host and crawl topology. Each entry MUST include the following elements:

- **HostName**: The host name for which the administrative host distribution rule applies.
- **CrawlStoreOrdinal**: The ordinal of the crawl store intended to contain crawled documents which reside on the host. The crawl store ordinal references a Crawl Store Set as defined in Section 3.1.1.3.
- **CrawlTopologyID**: The unique identifier of the crawl topology to which the host distribution rule has been applied. If this value is NULL, then it MUST be true that the rule has been added, but not yet applied to a crawl topology.
- **IsDeleted**: A bit which indicates whether an administrative host distribution rule has been deleted but not yet applied to the current crawl topology.

### 3.1.2 Timers

None.

### 3.1.3 Initialization

None.

### 3.1.4 Higher-Layer Triggered Events

None.

### 3.1.5 Message Processing Events and Sequencing Rules

Unless otherwise specified, all stored procedures defined in this section are located in the **search database**.

Unless otherwise specified, all stored procedure input parameters MUST NOT be NULL. As stored procedures use the input parameters for data retrieval from tables, failure to provide valid values will (unless otherwise specified) cause an error as described in [MS-TDS] section 2.2.7.9 that MUST be handled appropriately by the protocol client or the system behavior is indeterminate.

Unless otherwise specified, all fields returned in the result sets MUST NOT be NULL. For definitional clarity, a name has been assigned to any columns in the result sets that do not have a defined name in their current implementation. This does not affect the operation of the result set, as the ordinal position of any column with no defined name is expected by the front-end Web server. Such names are designated in the text using curly braces in the form {*name*}.

The data processing specified in this section references most of the elements of the abstract data model, as described in section 3.1.1.

The following table summarizes the stored procedures that are defined in this specification.

| Procedure Name | Description |
|---|---|
| **proc_MSS_AddConfigurationProperty** | Adds a new configuration property for search service application**.** |
| **proc_MSS_AddCrawlStoreRefactoringTask** | Registers a host name that needs to be moved during crawl store refactoring. |
| **proc_MSS_AddNewHostDistributionRule** | Adds new host distribution rule. |
| **proc_MSS_AddNewRebalancingRule** | Adds new automatic host distribution rule. |
| **proc_MSS_CheckIfCrawlStoreRefactoringTasksExist** | Checks if there is at least one host name that needs to be moved between crawl stores to activate a crawl topology. |
| **proc_MSS_CloneCrawlTopology** | Creates a copy of an existing crawl topology. |
| **proc_MSS_ClonePartitionScheme** | Creates a copy of an existing query topology. |
| **proc_MSS_CopyRulesForNewTopology** | Copies all host distribution rules that are part of the currently active crawl topology to another crawl topology. |
| **proc_MSS_CreateCrawlComponent** | Creates new crawl component. |
| **proc_MSS_CreateCrawlTopology** | Creates new crawl topology. |
| **proc_MSS_CreatePartitionScheme** | Creates new query topology. |
| **proc_MSS_CreateQueryComponent** | Creates new query component (2). |
| **proc_MSS_CreateRefactoringTask** | Creates new refactoring task. |
| **proc_MSS_CreateRefactoringTaskBatch** | Creates new refactoring task batch. |
| **proc_MSS_CreateTopologyActivationAction** | Creates a topology activation action. |
| **proc_MSS_DeleteCrawlComponent** | Deletes a crawl component. |
| **proc_MSS_DeleteCrawlStore** | Deletes a crawl store. |
| **proc_MSS_DeleteCrawlTopology** | Deletes a crawl topology. |
| **proc_MSS_DeletePartitionScheme** | Deletes a query topology. |
| **proc_MSS_DeletePropertyStore** | Deletes a metadata index. |
| **proc_MSS_DeleteQueryComponent** | Deletes a query component (2). |
| **proc_MSS_GetActiveRefactoringTaskBatches** | Retrieves list of refactoring task batches that are assigned to a given server. |

| Procedure Name | Description |
|---|---|
| **proc_MSS_GetConfigurationPropertyList** | Retrieves a list of configuration properties for a search service application. |
| **proc_MSS_GetCrawlComponent** | Retrieves properties of a crawl component. |
| **proc_MSS_GetCrawlComponents** | Retrieves list of crawl components. |
| **proc_MSS_GetCrawlComponentsForTopology** | Retrieves list of crawl components that belong to a given crawl topology. |
| **proc_MSS_GetCrawlStoreRefactoringTasks** | Receives list of host names that need to be moved between crawl stores when the specified crawl topology is activated. |
| **proc_MSS_GetCrawlStores** | Retrieves list of crawl stores. |
| **proc_MSS_GetCrawlTopologies** | Retrieves list of crawl topologies. |
| **proc_MSS_GetEndID** | Retrieves the largest document identifier(1) from an ordered set of document identifiers(1) for the specified host name. |
| **proc_MSS_GetFirstId** | Retrieves the smallest document identifier(1) from the set of all documents that have been crawled for the specified host name. |
| **proc_MSS_GetLastId** | Retrieves the largest document identifier(1) from the set of all documents that have been crawled for the specified host name. |
| **proc_MSS_GetListOfHostDistributionRules** | Retrieve list of all administrative host distribution rules that are a part of the currently active crawl topology. |
| **proc_MSS_GetNumberOfDocumentsForHost** | Retrieves the number of documents crawled on the specified host name. |
| **proc_MSS_GetNumberOfDocuments** | Retrieves total number of documents stored in each crawl store. |
| **proc_MSS_GetNumberOfDocumentsInCrawlStore** | Retrieves total number of documents stored in the specified crawl store. |
| **proc_MSS_GetNumberOfDocumentsPerHost** | Retrieves the number of documents crawled on each host name. |
| **proc_MSS_GetOldHostRule** | Determines if in the currently active crawl topology exists a host distribution rule for the specified host name, and if it is associated with the specified crawl store. |
| **proc_MSS_GetPartitions** | Retrieves list of index partitions. |

| Procedure Name | Description |
|---|---|
| **proc_MSS_GetPartitionSchemes** | Retrieves list of query topologies. |
| **proc_MSS_GetPropertyStoreHashesForActiveScheme** | Retrieves association between document distribution identifiers, index partitions and metadata indexes for the active query topology. |
| **proc_MSS_GetPropertyStores** | Retrieves list of metadata indexes. |
| **proc_MSS_GetQueryComponent** | Retrieves properties of a query component (2). |
| **proc_MSS_GetQueryComponents** | Retrieves list of query components(2). |
| **proc_MSS_GetQueryComponentsForActivePartitionScheme** | Retrieves list of query components (2) that belong to the active query topology. |
| **proc_MSS_GetQueryComponentsForPartitionScheme** | Retrieves list of query components (2) that belong to a given query topology. |
| **proc_MSS_GetRefactoringTask** | Retrieves properties of a refactoring tasks. |
| **proc_MSS_GetRefactoringTaskBatches** | Retrieves list of refactoring task batches. |
| **proc_MSS_GetRefactoringTasks** | Retrieves list of refactoring tasks. |
| **proc_MSS_GetRemovedRulesForCrawlStore** | Retrieves the list of host names for which there are administrative host distribution rules in the active crawl topology that have been marked for deletion and are associated with the specified crawl store. |
| **proc_MSS_GetRuleForHost** | Retrieves an administrative host distribution rule for the specified host name. |
| **proc_MSS_GetTopology** | Retrieves properties of the administration component. |
| **proc_MSS_GetTopologyActivationActions** | Retrieves list of topology activation actions. |
| **proc_MSS_MakeCrawlStoreShared** | Sets type of a crawl store to Dedicated (see Section 2.2.1.13). |
| **proc_MSS_MoveHostToDB** | Moves the specified host name to a new crawl store. |
| **proc_MSS_NeedToMoveDataFromDedicatedCrawlStores** | Determines if in the active crawl topology exists an administrative host distribution rule that is marked for deletion and is associated with a crawl store of type Dedicated (see Section 2.2.1.13). |

| Procedure Name | Description |
|---|---|
| **proc_MSS_RegisterCrawlStore** | Adds new crawl store. |
| **proc_MSS_RegisterPropertyStore** | Adds new metadata index. |
| **proc_MSS_RemoveCrawlStoreRefactoringTasks** | Clear the list of host names that need to be moved between crawl stores during activation of the specified crawl topology. |
| **proc_MSS_RemoveHostDistributionRule** | Removes an administrative host distribution rule. |
| **proc_MSS_ReportAdminComponentState** | Updates state of the administration component. |
| **proc_MSS_ReportCrawlComponentState** | Updates state of a crawl component. |
| **proc_MSS_ReportRefactoringTask** | Updates state of a refactoring task. |
| **proc_MSS_ReportRefactoringTaskBatch** | Updates state of a refactoring task batch. |
| **proc_MSS_ReportRefactoringTaskBatchError** | Stores error message for an error that occurred during execution of a refactoring task batch. |
| **proc_MSS_SetConfigurationPropertyEx** | Updates a configuration property record of a search service application. Can be forced to delete the existing record and recreate one if needed. |
| **proc_MSS_SetCrawlTopologyState** | Updates state of a crawl topology. |
| **proc_MSS_SetPartitionPropertyStore** | Associates given metadata index with an index partition. |
| **proc_MSS_SetPartitionSchemeState** | Update state of a query topology. |
| **proc_MSS_SetQueryComponent** | Updates properties of a query component (2). |
| **proc_MSS_SetTopologyIDForUncommittedRules** | Sets the crawl topology identifier for any administrative host distribution rules that have not yet been associated with a crawl topology. |
| **proc_MSS_UpdateCrawlComponent** | Updates properties of a crawl component. |
| **proc_MSS_UpdateRefactoringTaskBatchServer** | Reassigns refactoring task batch to a given server. |
| **proc_MSS_UpdateTopology** | Updates desired server name, desired local storage **path**, and desired type for the administration component. |
| **proc_MSS_UpdateTopologyActivationAction** | Updates state of a topology activation action. |

#### 3.1.5.1 proc_MSS_AddConfigurationProperty

The **proc_MSS_AddConfigurationProperty** stored procedure is called to add a new configuration property to a search service application. Upon successful execution, a configuration property MUST be added if there is no existing one with the specified name and value.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_AddConfigurationProperty (
    @Name                   nvarchar(300),
    @Value                  sql_variant
);
```

**@Name**: The name of this configuration property.

**@Value**: The value corresponding to the name of this property.

**Return Code Values:** This stored procedure MUST return **0** upon completion.

**Result Sets:** MUST NOT return any result sets.

#### 3.1.5.2 proc_MSS_AddCrawlStoreRefactoringTask

The **proc_MSS_AddCrawlStoreRefactoringTask** stored procedure is called to register a host name that needs to be moved during crawl store refactoring.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_AddCrawlStoreRefactoringTask (
    @HostName               nvarchar(256),
    @GthrDBID_from          int,
    @GthrDBID_to            int,
    @CrawlTopologyID        uniqueidentifier
);
```

**@HostName:** The host name of the host distribution rule. Host name data validation MUST occur before it is passed to the stored procedure.

**@GthrDBID_from:** The integer identifier of the crawl store from which the data will be moved.

**@GthrDBID_to:** The integer identifier of the crawl store to which the data will be moved.

**@CrawlTopologyID:** The unique identifier of the crawl topology that is being activated.

**Return Code Values:** This stored procedure MUST return **0** upon completion.

**Result Sets:** MUST NOT return any result sets.

#### 3.1.5.3 proc_MSS_AddNewHostDistributionRule

The **proc_MSS_AddNewHostDistributionRule** stored procedure is called to add a new host distribution rule. Upon successful execution, an administrative host distribution rule MUST be added and it MUST have its crawl topology identifier set to NULL. If the specified host name is not in the Host Set, a new host name MUST be added, and its host name identifier MUST be incremented by 1 above the maximum identifier among the current host names. If a new host name is added, an

automatic host distribution rule MUST be added with the current active crawl topology. The Host Distribution Rule Set and Host Set are described in section 3.1.1.5.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_AddNewHostDistributionRule (
    @HostName           nvarchar(100),
    @GthrDBGuid         uniqueidentifier
);
```

**@HostName:** The host name of the host distribution rule. Host name data validation MUST occur before the it is passed to the stored procedure.

**@GthrDBGuid:** The identifier of the crawl store for the host distribution rule.

**Return Code Values:** An integer which MUST be one of the values listed in the following table.

| Value | Description |
|---|---|
| **0** | The host distribution rule was successfully added. |
| **1** | An administrative host distribution rule was not added. If @GthrDBGuid does not reference any crawl store, no host name or rule was added. If an administrative host distribution rule exists for the host in the current active topology but with a crawl store that is different from the specified crawl store, a host and an automatic host distribution rule were added if the host did not already exist. |
| **2** | The host distribution rule was not added because it already exists. |

**Result Sets**: SHOULD NOT<1> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.4 proc_MSS_AddNewRebalancingRule

The **proc_MSS_AddNewRebalancingRule** stored procedure is called to add a new automatic host distribution rule. Upon successful execution, an automatic host distribution rule MUST be added if there is no existing administrative host distribution rule for the specified host name and crawl topology. In case there is an existing administrative host distribution rule for the specified host name and crawl topology the stored procedure MUST do nothing.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_AddNewRebalancingRule (
    @HostName           nvarchar(100),
    @GthrDBID           int,
    @CrawlTopologyID    uniqueidentifier
);
```

**@HostName:** The host name of the host distribution rule. Host name data validation MUST occur before the data is passed to the stored procedure.

**@GthrDBID:** The ordinal of the crawl store for the host distribution rule.

**@CrawlTopologyID:** The crawl topology identifier of the host distribution rule.

**Return Code Values**: This stored procedure MUST return 0 upon completion.

**Result Sets**: SHOULD NOT<2> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.5 proc_MSS_CheckIfCrawlStoreRefactoringTasksExist

The **proc_MSS_CheckIfCrawlStoreRefactoringTasksExist** stored procedure is called to determine if there is a host name that needs to be moved between crawl stores to activate the specified crawl topology.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CheckIfCrawlStoreRefactoringTasksExist (
    @CrawlTopologyId          uniqueidentifier
);
```

**@CrawlTopologyId:** The unique identifier of the crawl topology.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | The specified crawl topology identifier is not valid or there are no host names that need to be moved for the specified crawl topology; that is, the Crawl Store Refactoring Task Set described in Section 3.1.1.4 does not contain any entries for the specified crawl topology. |
| **1** | There is at least one host name that needs to be moved between crawl stores when the specified crawl topology is activated. |

**Result Sets:** The client MUST ignore any result sets returned by the stored procedure.

### 3.1.5.6 proc_MSS_CheckNumberOfRows

The **proc_MSS_CheckNumberOfRows** stored procedure is called to compare the current number of rows in the table specified by the parameter @TableName and the previously calculated number of rows specified by the parameter @NumOfRows.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CheckNumberOfRows (
    @TableName            nvarchar(256),
    @NumOfRows            int,
    @OriginalNumberOfRows int OUTPUT
);
```

**@TableName:** The name of a SQL table.

**@NumOfRows:** Previously calculated number of rows in the table with the name @TableName.

**@OriginalNumberOfRows:** Upon return from this stored procedure, this parameter MUST be set to the current number of rows in the table with the name @TableName.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | The value of @NumOfRows equals the value of @OriginalNumberOfRows. |
| **1** | The values of @NumOfRows and @OriginalNumberOfRows are different. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.7 proc_MSS_CloneCrawlTopology

The **proc_MSS_CloneCrawlTopology** stored procedure is called to create a copy of the existing crawl topology. A new unique identifier MUST be assigned to the created topology. The state of the created topology MUST be set to 0 and the creation time MUST be set to the UTC time of copying. The new crawl topology MUST have the same set of crawl component as the existing one. The crawl topology and crawl components are described in section 3.1.1.3.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CloneCrawlTopology (
    @CrawlTopologyId          uniqueidentifier,
    @NewCrawlTopologyId       uniqueidentifier OUTPUT
);
```

**@CrawlTopologyId:** The identifier of the crawl topology to be copied.

**@NewCrawlTopologyId:** Upon return from this stored procedure, this parameter MUST be set to the identifier of the created crawl topology.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | The specified crawl topology does not exist. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.8 proc_MSS_ClonePartitionScheme

The **proc_MSS_ClonePartitionScheme** is called to create a query topology and associate index partitions and query components from an existing query topology with the new query topology. The state of the query topology that is created MUST be set to Inactive (see Section 2.2.1.2). The newly created query topology MUST be associated to all index partitions and all query components(2) that are currently associated with the specified existing query topology.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_ClonePartitionScheme (
    @PartitionSchemeID       uniqueidentifier,
    @NewPartitionSchemeID    uniqueidentifier OUTPUT
);
```

**@PartitionSchemeID:** The unique identifier of the query topology index partitions and query components that are being copied from.

**@NewPartitionSchemeID:** Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the created query topology, unless topology was not created because there is no query topology with the unique identifier specified with @PartitionSchemeID.

**Return Code Values:** An integer which MUST be one of the values listed in the following table.

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | There is no query topology with the specified unique identifier. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.9 proc_MSS_CopyRulesForNewTopology

The **proc_MSS_CopyRulesForNewTopology** stored procedure is called to copy all Host Distribution Rules (section 3.1.1.5) that are a part of the currently active Crawl Topology (section 3.1.1.3) to the specified activating Crawl Topology (section 3.1.1.3). For each existing Host Distribution Rule (section 3.1.1.5) that is a part of the current active Crawl Topology (section 3.1.1.3) a new Host Distribution Rule (section 3.1.1.5) MUST be created with the same attributes as the existing rule, except for the crawl topology identifier which MUST be set to the specified identifier. The stored procedure MUST set IsDeleted value to **0** for all administrative host distribution rules that are a part of the currently active Crawl Topology (section 3.1.1.3).

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CopyRulesForNewTopology (
    @ActivatingTopologyID     uniqueidentifier
);
```

**@ActivatingTopologyID:** The identifier of the activating crawl topology.

**Return Code Values**: This stored procedure MUST return **0** upon completion.

**Result Sets**: SHOULD NOT return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.10 proc_MSS_CreateCrawlComponent

The **proc_MSS_CreateCrawlComponent** stored procedure is called to create a new crawl component. A new CrawlComponentID and new CrawlComponentNumber MUST be assigned to the created component. The created crawl component MUST be associated with the specified crawl store and this crawl component MUST be added to the specified inactive crawl topology. The created crawl component MUST be marked as a "master component" if there is no master component in the specified crawl topology; otherwise, it MUST be marked as "no master".

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CreateCrawlComponent (
    @ServerName               nvarchar(256),
    @ServerID                 uniqueidentifier,
```

```
    @LocalStoragePath         nvarchar(260),
    @CrawlTopologyId          uniqueidentifier,
    @CrawlStoreId             uniqueidentifier,
    @DesiredState             int,
    @CrawlComponentId         uniqueidentifier OUTPUT,
    @CrawlComponentNumber     int OUTPUT,
    @Master                   int OUTPUT
);
```

**@ServerName:** The name of the server for the new crawl component.

**@ServerID:** The unique identifier of the server for the new crawl component.

**@LocalStoragePath:** The local storage path for the new crawl component.

**@CrawlTopologyId:** The identifier of the crawl topology where the created crawl component MUST be added.

**@CrawlStoreId:** The identifier of the crawl store which MUST be associated with the new crawl component.

**@DesiredState:** The desired state of the crawl component which MUST be a Crawl Component State data type as specified in section 2.2.1.7.

**@CrawlComponentId:** Upon return from this stored procedure, this parameter MUST be the unique identifier of the created crawl component.

**@CrawlComponentNumber:** Upon return from this stored procedure, this parameter MUST be the integer identifier of the created crawl component.

**@Master:** Upon return from this stored procedure, this parameter MUST an integer that is one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | The created crawl component is not the master component. |
| **1** | The created crawl component is the master component. |

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | The crawl topology the created crawl component MUST be added to does not exist. |
| **2** | The crawl store which the created crawl component MUST be associated with does not exist. |
| **3** | The state of the crawl topology the created crawl component MUST be added to is not Inactive. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.11 proc_MSS_CreateCrawlTopology

The **proc_MSS_CreateCrawlTopology** stored procedure is called to create a new crawl topology. A new unique identifier MUST be assigned to the created topology. The new topology MUST NOT have crawl components associated with it.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CreateCrawlTopology (
    @CrawlTopologyId          uniqueidentifier OUTPUT
);
```

**@CrawlTopologyId:** Upon return from this stored procedure, this parameter MUST be an identifier of the created crawl topology.

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** MUST NOT return any result set.

### 3.1.5.12 proc_MSS_CreatePartitionScheme

The **proc_MSS_CreatePartitionScheme** stored procedure is called to create a query topology. The state of the created query topology must be set to Inactive. The stored procedure MUST also create set of index partitions and register a set of document distribution identifiers to each of these index partitions. The newly created index partitions MUST be associated with the newly created query topology. For each of the created index partitions the stored procedure MUST set the integer ordinal to an integer between 0 and *n*-1 (inclusive), where *n* is the number of index partitions created by the stored proc. The integer ordinal for each of the created index partitions MUST be unique within the set of these index partitions. To each of the created index partitions the stored procedure MUST assign all document distribution identifiers between $i \times 256/n$ and $((i+1) \times 256/n) - 1$ (inclusive), where *i* is the integer ordinal of the index partition and '/' is integer division.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CreatePartitionScheme (
    @PartitionsNumber         smallint,
    @PartitionSchemeID        uniqueidentifier OUTPUT
);
```

**@PartitionsNumber:** The number of index partitions in the new query topology. This value MUST be an integer that is greater than 0 and less than 256.

**@PartitionSchemeID:** Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the created query topology.

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** MUST NOT return any result set.

### 3.1.5.13 proc_MSS_CreateQueryComponent

The **proc_MSS_CreateQueryComponent** stored procedure is called to create a query component (2). To create a query component (2) the specified query topology MUST be in the "Inactive" state. The stored procedure MUST also associate the query component that is being created with the specified query topology and index partition. If the unique identifier of the query topology or the

unique identifier of the index partition passed to the stored procedures are not valid then the stored procedure MUST NOT create the query component (2). State and desired state of the query component (2) created with this stored procedure MUST be set to "Uninitialized" after creation.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CreateQueryComponent (
    @ServerName               nvarchar(256),
    @ServerID                 uniqueidentifier,
    @LocalStoragePath         nvarchar(260),
    @PartitionSchemeID        uniqueidentifier,
    @PartitionID              uniqueidentifier,
    @DesiredState             int,
    @HotSwap                  int,
    @ShareName                nvarchar(260),
    @UsesCustomShare          int,
    @QueryComponentID         uniqueidentifier OUTPUT,
    @QueryComponentNumber     int OUTPUT
);
```

**@ServerName:** The name of the server where the new query component (2) is located.

**@ServerID:** The unique identifier where the new query component (2) is located.

**@LocalStoragePath:** The local storage path for the component being created.

**@PartitionSchemeID:** The unique identifier of the query topology the new query component (2) should be associated with.

**@PartitionID:** The unique identifier of the index partition the new query component (2) should be associated with.

**@DesiredState:** This parameter MUST be set to 0.

**@HotSwap:** The type of the new query component (2). The value MUST be a Query Component Type data type as specified in Section 2.2.1.4.

**@ShareName:** The name of the shared folder used by this component. This parameter MUST be set to NULL if @UsesCustomShare is set to 0.

**@UsesCustomShare:** If set to 1 then the query component (2) MUST use a custom name for the shared folder that is used to copy the full-text index catalog to that query component (2). The name of the shared folder is specified with the @ShareName parameter. If set to 0 then the default shared folder name MUST be used by the new query component (2).

**@QueryComponentID:** Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the newly created query component (2).

**@QueryComponentNumber:** Upon return from this stored procedure, this parameter MUST be set to the integer identifier of the newly created query component (2).

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |

| Value | Description |
|---|---|
| **1** | There is no query topology with the specified unique identifier. |
| **2** | There is no index partition with the specified unique identifier. |
| **3** | The specified index partition is not a part of the specified query topology. |
| **4** | The specified query topology is not in the Inactive state and cannot be changed. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.14 proc_MSS_CreateRefactoringTask

The **proc_MSS_CreateRefactoringTask** stored procedure is called to create a new refactoring task under the specified topology activation action.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CreateRefactoringTask (
    @ActionID                int,
    @TaskType                nvarchar(256),
    @SourceComponentID       uniqueidentifier,
    @DestinationComponentID  uniqueidentifier,
    @StartDocID              bigint,
    @EndDocID                bigint,
    @PartsXml                ntext,
    @TaskID                  int OUTPUT
);
```

**@ActionID:** The unique identifier of the topology activation action this task is a part of.

**@TaskType:** The type of the refactoring task. The value MUST be a Refactoring Task Type data type as specified in Section 2.2.1.10.

**@SourceComponentID:** The unique identifier of the metadata index from which the data is being copied.

**@DestinationComponentID:** The unique identifier of the metadata index to which the data is being copied.

**@StartDocID:** The document identifier of the first document that will be copied by this task. The value MUST be **-1** if the number of documents in the source database is not yet known.

**@EndDocID:** The document identifier of the last document that will be copied by this task. The value MUST be **-1** if the number of documents in the source database is not known yet.

**@PartsXml:** XML document that contains list of refactoring task parts for the new refactoring task. This parameter MUST adhere to the TaskParts Scheme (Section 2.2.6.4.1).

**@TaskID:** Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the newly created refactoring task.

**Return Code Values:** An integer that MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **1** | The activation action identifier specified is not valid. |
| **0** | Successful execution. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.15 proc_MSS_CreateRefactoringTaskBatch

The **proc_MSS_CreateRefactoringTaskBatch** stored procedure is called to create a new refactoring task batch associated the specified refactoring task.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CreateRefactoringTaskBatch (
    @TaskID             int,
    @StartDocID         bigint,
    @EndDocID           bigint,
    @NumOfDocs          int,
    @ServerName         nvarchar(256),
    @BatchID            int OUTPUT
);
```

**@TaskID:** The unique identifier of the refactoring task.

**@StartDocID:** The beginning of the interval of document identifiers that defines a set of documents that need to be processed by this refactoring task batch. If the type of the refactoring task is set to "CrawlStoreMove" then this field can be set to **-1**. If it is set to **-1**, this batch corresponds to the steps that need to be performed to finish the refactoring task (see Section 3.2.5.4).

**@EndDocID:** The end of the interval of document identifiers that defines the set of documents that need to be processed by this refactoring task batch. This value is set to -1 if and only if @StartDocID is set to **-1**.

**@NumOfDocs:** An integer that MUST be set to **-1** for refactoring task batches created for a refactoring task of type *PropertyStoreCopy* or *PropertyStoreDelete*; otherwise, if the type of the refactoring task this refactoring task batch is associated with is set to *CrawlStoreMove*, then this field contains number of documents being copied by this refactoring task batch.

**@ServerName:** The name of the server the refactoring task batch is assigned to.

**@BatchID:** The unique identifier of the refactoring task batch.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | Refactoring task with the specified identifier does not exist. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.16 proc_MSS_CreateTopologyActivationAction

The **proc_MSS_CreateTopologyActivationAction** stored procedure is called to create a new topology activation action. The stored procedure MUST NOT create a new topology activation action if there is another topology activation action with the same name and associated with the same query or crawl topology.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CreateTopologyActivationAction (
    @Name               nvarchar(256),
    @TopologyID         uniqueidentifier,
    @ActionID           int OUTPUT
);
```

**@Name:** The name of the new topology activation action.

**@TopologyID:** The unique identifier of the query topology or the crawl topology the new topology activation action is created for.

**@ActionID:** Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the newly created topology activation action.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | A topology activation action with the same name is already created for the specified query or crawl topology. A duplicate topology activation action cannot be created. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.17 proc_MSS_DeleteCrawlComponent

The **proc_MSS_DeleteCrawlComponent** stored procedure is called to remove a crawl component from the specified inactive crawl topology. The stored procedure MUST delete the crawl component if it not associated with any other crawl topology.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_DeleteCrawlComponent (
    @CrawlTopologyId          uniqueidentifier,
    @CrawlComponentId         uniqueidentifier
);
```

**@CrawlTopologyId:** The identifier of the crawl topology.

**@CrawlComponentId:** The identifier of the crawl component.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | The crawl component to be deleted does not exist. |
| **2** | The crawl topology from which the crawl component MUST be deleted does not exist. |
| **3** | The crawl component is not associated with specified crawl topology. |
| **4** | The crawl topology state is not Inactive. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.18 proc_MSS_DeleteCrawlStore

The **proc_MSS_DeleteCrawlStore** stored procedure is called to delete a crawl store. This procedure MUST NOT delete a crawl store if there is at least one crawl component associated with it.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_DeleteCrawlStore (
    CrawlStoreId       uniqueidentifier
);
```

**CrawlStoreId:** The unique identifier of the crawl store to be deleted.

**Return Code Values:** An integer which MUST be listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | The crawl store has crawl components associated with it. |
| **2** | The crawl store to be deleted does not exist. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.19 proc_MSS_DeleteCrawlTopology

The **proc_MSS_DeleteCrawlTopology** stored procedure is called to delete an inactive crawl topology. This topology to be deleted MUST NOT have any crawl components associated with it. The procedure MUST delete all topology activation actions that are associated with the specified crawl topology together with all refactoring tasks and refactoring task batches created for these topology activation actions. The procedure MUST delete all administrative host distribution rules and all automatic host distribution rules associated with the crawl topology.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_DeleteCrawlTopology (
    @CrawlTopologyId          uniqueidentifier
);
```

**@CrawlTopologyId:** The identifier of the crawl topology.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | The crawl topology has crawl components associated with it. |
| **2** | The crawl topology state is not inactive. |
| **3** | The crawl topology to be deleted does not exist. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.20 proc_MSS_DeletePartitionScheme

The **proc_MSS_DeletePartitionScheme** stored procedure is called to delete a query topology. This stored procedure MUST only delete a query topology if it is in the Inactive state and there is no query component (2) associated with that query topology. The stored procedure MUST delete all existing topology activation actions associated with the query topology as well as all refactoring tasks and refactoring task batches that are part of these topology activation actions.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_DeletePartitionScheme (
    @PartitionSchemeID        uniqueidentifier
);
```

**@PartitionSchemeID:** The unique identifier of the query topology.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | The query topology in not in the Inactive state. |
| **2** | There is a query component (2) associated with the query topology. |
| **3** | There is no query topology with the specified unique identifier. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.21 proc_MSS_DeletePropertyStore

The **proc_MSS_DeletePropertyStore** stored procedure is called to delete a metadata index. It MUST NOT delete a metadata index if there is at least one index partition associated with that metadata index.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_DeletePropertyStore (
    @PropertyStoreID          uniqueidentifier
);
```

**@PropertyStoreID:** The unique identifier of the metadata index.

**Return Code Values:** An integer which MUST be in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | There is an index partition associated with the metadata index. |
| **2** | There is no metadata index with the specified unique identifier. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.22 proc_MSS_DeleteQueryComponent

The **proc_MSS_DeleteQueryComponent** stored procedure is called to remove a query component (2) from the specified query topology. The stored procedure MUST delete the query component (2) if it is not associated with a query topology other than the one it is being removed from. The stored procedure MUST only remove the specified query component (2) from the specified query topology if the query topology is in the Inactive state (The corresponding result value MUST be returned in that case).

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_DeleteQueryComponent (
    @PartitionSchemeID        uniqueidentifier,
    @QueryComponentID         uniqueidentifier
);
```

**@PartitionSchemeID:** The unique identifier of the query topology.

**@QueryComponentID:** The unique identifier of the query component.

**Return Code Values**: An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | There is no query component (2) with the specified unique identifier. |
| **2** | There is no query topology with the specified unique identifier. |
| **3** | The specified query component (2) is not associated with the specified query topology. |
| **4** | The query topology is in the Inactive state. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.23 proc_MSS_GetActiveRefactoringTaskBatches

The **proc_MSS_GetActiveRefactoringTaskBatches** stored procedure is called to retrieve a list of refactoring task batches assigned to the specified server.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetActiveRefactoringTaskBatches (
    @ServerName         nvarchar(256),
    @BatchesCount       int,
    @MaxErrorCount      int
);
```

**@ServerName:** The name of the server the refactoring task batch is assigned to.

**@BatchesCount:** The number of refactoring task batches specified by the caller to return.

**@MaxErrorCount:** The maximum number of errors allowed when executing each of the returned refactoring task batches. The value is not used.

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** MUST return the **Refactoring Task Batches Result Set** as described in section 2.2.4.3. Entries in that result set returned by this stored procedure MUST be sorted by the batch identifier. Number of rows should be limited by the value specified with the @BatchesCount parameter.

### 3.1.5.24 proc_MSS_GetConfigurationPropertyList

The **proc_MSS_GetConfigurationPropertyList** stored procedure is called to retrieve a list of configuration properties of a search service application with the same specified property name.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetConfigurationPropertyList (
    @Name          nvarchar(300)
);
```

**@Name:** The name of the configuration property.

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** This stored procedure MUST return the Configuration Property List Result Set which MUST contain one row if the requested component exists; otherwise, it MUST return zero rows.

#### 3.1.5.24.1 Configuration Property List Result Set

The **Configuration Property List** result set MUST contain zero or more rows, each corresponding to a single configuration property of a search service application.

The T-SQL syntax for the result set is as follows:

```
Name                    nvarchar(300),
Value                   sql_variant,
```

**Name:** The name of the configuration property.

**Value:** The value corresponding to the name of the configuration property.

### 3.1.5.25 proc_MSS_GetCrawlComponent

The **proc_MSS_GetCrawlComponent** stored procedure is called to retrieve a crawl component.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetCrawlComponent (
    @CrawlComponentId         uniqueidentifier
);
```

**@CrawlComponentId:** The identifier of the crawl component.

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** This stored procedure MUST return the Crawl Component Result Set which MUST contain one row if the requested component exists; otherwise, it MUST return zero rows.

### 3.1.5.26 proc_MSS_GetCrawlComponents

The **proc_MSS_GetCrawlComponents** stored procedure is called to retrieve a list of all crawl components.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetCrawlComponents ();
```

**Return Code Values:** An integer value that MUST be ignored.

**Result Sets:** This stored procedure MUST return the Crawl Component Result Set.

### 3.1.5.27 proc_MSS_GetCrawlComponentsForTopology

The **proc_MSS_GetCrawlComponentsForTopology** stored procedure is called to retrieve a list of all crawl components for the specified crawl topology.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetCrawlComponentsForTopology (
    @CrawlTopologyId          uniqueidentifier
);
```

**@CrawlTopologyId:** The unique identifier of the crawl topology.

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** This procedure MUST return the Crawl Component Result Set.

### 3.1.5.28 proc_MSS_GetCrawlStoreRefactoringTasks

The **proc_MSS_GetCrawlStoreRefactoringTasks** stored procedure is called to receive a list of host names that need to be moved between crawl stores when the specified crawl topology is activated.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetCrawlStoreRefactoringTasks (
    @CrawlTopologyID          uniqueidentifier
);
```

**@CrawlTopologyID:** The unique identifier of the crawl topology.

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** MUST return the Crawl Store Refactoring Tasks Result Set as described in section 3.1.5.28.1

#### 3.1.5.28.1 Crawl Store Refactoring Tasks Result Set

The **Crawl Store Refactoring Tasks** result set MUST contain zero or more rows, each corresponding to a single host name.

The T-SQL syntax for the result set is as follows:

```
HostID                int NOT NULL,
GthrDBGuid_from       uniqueidentifier NOT NULL,
GthrDBGuid_to         uniqueidentifier NOT NULL;
```

**HostID:** The ordinal identifier for the host name.

**GthrDBGuid_from:** The ordinal of the crawl store from which the data will be moved.

**GthrDBGuid_to:** The ordinal of the crawl store to which the data will be moved.

### 3.1.5.29 proc_MSS_GetCrawlStores

The **proc_MSS_GetCrawlStores** stored procedure is called to retrieve a list of all crawl stores.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetCrawlStores ();
```

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** This stored procedure MUST return the Crawl Stores Result Set as described in section 3.1.5.29.1.

#### 3.1.5.29.1 Crawl Stores Result Set

The Crawl Stores result set MUST contain zero or more rows, each corresponding to a single crawl store.

The T-SQL syntax for the result set is as follows:

```
CrawlStoreID          uniqueidentifier NOT NULL,
Name                  nvarchar(256) NOT NULL,
Ordinal               int NOT NULL,
NewOrdinal            int NOT NULL,
IsDedicated           int NOT NULL;
```

**CrawlStoreID:** The identifier of the crawl store.

**Name:** The name of the crawl store.

**Ordinal:** The integer crawl store identifier.

**NewOrdinal:** The subsequent integer crawl store identifier (see Section 3.1.1.3).

**IsDedicated:** The crawl store type. The value MUST be a Crawl Store Type data type as specified in section 2.2.1.13.

### 3.1.5.30 proc_MSS_GetCrawlTopologies

The **proc_MSS_GetCrawlTopologies** stored procedure is called to retrieve a list of crawl topologies.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetCrawlTopologies ();
```

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** This stored procedure MUST return the Crawl Topologies Result Set as described in section 3.1.5.30.1.

#### 3.1.5.30.1 Crawl Topologies Result Set

The Crawl Topologies Result Set MUST contain zero or more rows, each corresponding to a single crawl topology.

The T-SQL syntax for the result set is as follows:

```
CrawlTopologyID         uniqueidentifier NOT NULL,
CreationDate            datetime NOT NULL,
State                   smallint NOT NULL;
```

**CrawlTopologyID:** The identifier of the crawl topology.

**CreationDate:** The UTC time when the crawl topology was created.

**State:** The current state of the crawl topology as described in section 2.2.1.6.

### 3.1.5.31 proc_MSS_GetDatabaseSchemaVersion

The **proc_MSS_GetDatabaseSchemaVersion** stored procedure is called to retrieve version of the protocol used by the server. The Client MUST use this function to determine version of the protocol and MUST NOT use the corresponding server if the received version number is not supported by the client.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetDatabaseSchemaVersion (
    @VersionId nvarchar(64),
    @Version nvarchar(64) OUTPUT
```

```
);
```

**@VersionId:** The identifier of the requested version. This parameter SHOULD<3> be set to the value in the following table:

| Value | Description |
|---|---|
| **"A93AE1C8-B85A-45D4-913B-7A68CEEE03B8"** | Version of SQL Administration Protocol and Search Topology Protocol. |
| **"A03EE87B-398E-470B-914B-93148F38A7E5"** | Version of Search Service Database Query Protocol. |
| **"54B868C8-365D-4936-84DF-D6928E872713"** | Version of SQL Gatherer Protocol. |

**@Version:** Upon return from this stored procedure, this parameter MUST be set to the value of the requested protocol version.<4>

**Return Code Values:** An integer that MUST be **0**.

**Result Sets:** SHOULD NOT<5> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.32 proc_MSS_GetEndID

The **proc_MSS_GetEndID** stored procedure is called to retrieve the largest identifier from an ordered set of identifiers for the specified host. The smallest identifier in the set, the size of the set, and the source of identifiers in the set are specified by the stored procedure parameters.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetEndID (
    @HostId             int,
    @StartId            int,
    @NumOfDocs          int,
    @TableIndex         int,
    @OutNumOfDocs       int OUTPUT,
    @EndID              bigint OUTPUT
);
```

**@HostId:** The identifier of the host name.

**@StartId:** The smallest identifier in the ordered set.

**@NumOfDocs:** The number of identifiers in the ordered set.

**@TableIndex:** An integer that MUST be set to one of the values in the following table:

| Value | Description |
|---|---|
| **0** | The ordered set contains document identifiers from Crawl URL History (specified in [MS-SQLPGAT2] section 3.1.1.2). |
| **1** | The ordered set contains link identifiers from Anchor Text Info (specified in [MS-SQLPGAT2] |

| Value | Description |
|---|---|
| | section 3.1.1.8). |
| **2** | The ordered set contains link identifiers from Links( specified in [MS-SQLPGAT2] section 3.1.1.5) |
| **3** | The ordered set contains TrackIDs from Deleted URL (specified in [MS-SQLPGAT2] section 3.1.1.3) |

**@OutNumOfDocs:** Upon return from this stored procedure, this parameter MUST be set to one of the following values, depending on the value of @TableIndex:

| Value of @TableIndex | Value |
|---|---|
| **0** | The number of items in the Crawl URL History (specified in [MS-SQLPGAT2] section 3.1.1.2), minus 1. |
| **1** | The number of items in the Anchor Text Info set (specified in [MS-SQLPGAT2] section 3.1.1.8), minus 1. |
| **2** | The number of Links (specified in [MS-SQLPGAT2] section 3.1.1.5), minus 1. |
| **3** | The number of items in the Deleted URL set (specified in [MS-SQLPGAT2] section 3.1.1.3), minus 1. |

**@EndID:** An integer that is an identifier.

**Result Sets:** SHOULD NOT<6> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.33 proc_MSS_GetFirstId

The **proc_MSS_GetFirstID** stored procedure is called to retrieve the smallest identifier from the set of all identifiers for the specified host. The source of identifiers in the set is specified by the stored procedure parameters.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetFirstID (
    @HostId             int,
    @TableIndex         int,
    @FirstID            bigint OUTPUT
);
```

**@HostId:** The identifier of the host name.

**@TableIndex:** An integer that MUST be value listed in the following table:

| Value | Description |
|---|---|
| **0** | The set contains document identifiers from Crawl URL History (specified in [MS-SQLPGAT2] section 3.1.1.2). |
| **1** | The set contains link identifiers from Anchor Text Info (specified in [MS-SQLPGAT2] section 3.1.1.8). |

| Value | Description |
|---|---|
| **2** | The set contains link identifiers from Links( specified in [MS-SQLPGAT2] section 3.1.1.5). |
| **3** | The set contains TrackIDs from Deleted URL (specified in [MS-SQLPGAT2] section 3.1.1.3). |

**@FirstID:** An integer that is an identifier.

**Result Sets:** SHOULD NOT<7> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.34 proc_MSS_GetLastId

The **proc_MSS_GetLastId** stored procedure is called to retrieve the largest identifier from the set of all identifiers for the specified host. The source of identifiers in the set is specified by the stored procedure parameters.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetLastID (
    @HostId             int,
    @TableIndex         int,
    @LastID             bigint OUTPUT
);
```

**@HostId:** The identifier of the host name.

**@TableIndex:** An integer that MUST be value listed in the following table:

| Value | Description |
|---|---|
| **0** | The set contains document identifiers from Crawl URL History (specified in [MS-SQLPGAT2] section 3.1.1.2). |
| **1** | The set contains link identifiers from Anchor Text Info (specified in [MS-SQLPGAT2] section 3.1.1.8). |
| **2** | The set contains link identifiers from Links( specified in [MS-SQLPGAT2] section 3.1.1.5). |
| **3** | The set contains TrackIDs from Deleted URL (specified in [MS-SQLPGAT2] section 3.1.1.3). |

**@LastID:** An integer that is an identifier.

**Result Sets:** SHOULD NOT<8> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.35 proc_MSS_GetListOfHostDistributionRules

The **proc_MSS_GetListOfHostDistributionRules** stored procedure is called to retrieve the Administrative Host Distribution Rules Result Set (section 3.1.5.35.1) that is a part of the current active crawl topology (section 3.1.1.3). The crawl topology MUST not be in the *Activating* state when this stored procedure is called.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetListOfHostDistributionRules ();
```

**Return Code Values:** This stored procedure MUST return **0** upon completion.

**Result Sets:** This stored procedure MUST return the Host Distribution Rule Result Set.

#### 3.1.5.35.1 Host Distribution Rule Result Set

The **Host Distribution Rule Result Set** returns the list of administrative host distribution rules. The result set MUST contain one row for each administrative host distribution rule that is part of the current active crawl topology (section 3.1.1.3).

The T-SQL syntax for the result set is as follows:

```
HostName               nvarchar(300) NOT NULL,
GthrDBID               int NOT NULL,
CrawlStoreID           uniqueidentifier NOT NULL,
CrawlTopologyID        uniqueidentifier NOT NULL,
IsMarkedForDeletion    int NOT NULL;
```

**HostName:** The host name of the host distribution rule.

**GthrDBID:** The unique identifier of the crawl store of the host distribution rule.

**CrawlStoreID:** The ordinal of the crawl store of the host distribution rule.

**CrawlTopologyID:** The crawl topology identifier of the host distribution rule.

**IsMarkedForDeletion:** An integer which MUST equal **1** if the admin requested to delete this rule; otherwise, it MUST be **0**.

### 3.1.5.36 proc_MSS_GetNumberOfAnchorRowsForHost

The **proc_MSS_GetNumberOfAnchorRowsForHost** stored procedure is called to retrieve the number of URLs discovered during a crawl of the specified host.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetNumberOfAnchorRowsForHost (
    @HostId            int
);
```

**@HostId:** The identifier of the host name.

**Return Code Values:** An integer which MUST be the number of URLs discovered during crawling of the specified host name or **0** if the specified host name doesn't exist.

**Result Sets:** MUST NOT return any result set.

### 3.1.5.37 proc_MSS_GetNumberOfAnchorRowsPerHost

The **proc_MSS_GetNumberOfAnchorRowsPerHost** stored procedure is called to retrieve the number of URLs discovered during a crawl of each host.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetNumberOfAnchorRowsPerHost ();
```

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** This stored procedure MUST return the Number Of Anchor Rows Per Host Result Set.

#### 3.1.5.37.1 Number Of Anchor Rows Per Host Result Set

The Number Of Anchor Rows Per Host Result Set returns information about the number of URLs discovered during crawling of each host.

The T-SQL syntax for the result set is as follows:

```
HostID      int NOT NULL,
NumOfRaws   int NOT NULL;
```

**HostID:** The identifier of the host name.

**NumOfRows:** The total number of URLs discovered during crawling of the corresponding host.

### 3.1.5.38 proc_MSS_GetNumberOfDocumentsForHost

The **proc_MSS_GetNumberOfDocumentsForHost** stored procedure is called to retrieve the number of documents crawled on the specified host.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetNumberOfDocumentsForHost (
    @HostId            int
);
```

**@HostId:** The identifier of the host name.

**Return Code Values:** An integer which MUST be the number of documents crawled on the specified host name or **0** if the specified host name doesn't exist.

**Result Sets:** MUST NOT return any result set.

### 3.1.5.39 proc_MSS_GetNumberOfDocuments

The **proc_MSS_GetNumberOfDocuments** stored procedure is called to retrieve a list of the total number of documents stored in each crawl store. The Crawl Store Set is described in section 3.1.1.3.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetNumberOfDocuments ();
```

**Return Code Values:** This stored procedure MUST return 0 upon completion.

**Result Sets:** This stored procedure MUST return the Crawl Store Document Summary Result Set.

#### 3.1.5.39.1 Crawl Store Document Summary Result Set

The Crawl Store Document Summary Result Set returns information about the number of documents stored in each crawl store. The result set MUST have one row for every crawl store.

The T-SQL syntax for the result set is as follows:

```
CrawlStoreID    uniqueidentifier NOT NULL,
DocCount        int NOT NULL;
```

**CrawlStoreID:** The ordinal of the crawl store.

**DocCount:** The total number of documents in the crawl store.

### 3.1.5.40 proc_MSS_GetNumberOfDocumentsInCrawlStore

The **proc_MSS_GetNumberOfDocumentsInCrawlStore** stored procedure is called to retrieve the number of documents that are stored in the specified crawl store.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetNumberOfDocumentsInCrawlStore (
    @Ordinal           int
);
```

**@Ordinal:** The integer identifier of the crawl store.

**Return Code Value**: An integer value representing the number of documents in the crawl store. If there is no crawl store with the specified integer identifier the stored procedure MUST return NULL.

**Result Sets**: MUST NOT return any result set.

### 3.1.5.41 proc_MSS_GetNumberOfDocumentsPerHost

The **proc_MSS_GetNumberOfDocumentsPerHost** stored procedure is called to retrieve the number of documents crawled on each host.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetNumberOfDocumentsPerHost ();
```

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** This stored procedure MUST return the Number Of Documents Per Host Result Set

#### 3.1.5.41.1 Number Of Documents Per Host Result Set

The Number Of Documents Per Host Result Set returns information about the number of documents crawled on each host.

The T-SQL syntax for the result set is as follows:

```
HostID      int NOT NULL,
```

```
NumOfDocs   int NOT NULL;
```

**HostID:** The identifier of the host name.

**NumOfDocs:** The total number of documents crawled on the corresponding host.

### 3.1.5.42 proc_MSS_GetNumberOfRows

The **proc_MSS_GetNumberOfRows** stored procedure is called to retrieve the number of records in the specified table.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetNumberOfRows (
    @TableName          nvarchar(256)

);
```

**@TableName:** The name of the table. This MUST be one of the following values:

"MSSCrawlHostList"

"MSSCrawlHostsLog"

"MSSUserHosts"

"MSSCrawlUrlUsedContentSourceReport"

"MSSAnchorChangeLog"

"MSSAnchorPendingChangeLog"

"MSSSocialDistance"

"MSSSocialDistanceLinkNew"

"MSSSocialDistanceLinkDelete"

"MSSSocialDistanceLinkUnchanged"

"MSSTranTempTable1"

"MSSAnnotations"

"MSSCrawlChangedCommittedDocs"

"MSSCrawlChangedDeletedDocs"

"MSSCrawlChangedSourceDocs"

"MSSCrawlChangedTargetDocs"

"MSSCrawlLinksLog"

"MSSCrawlURL"

"MSSCrawlURLReport"

"MSSCrawlQueue"

"MSSAnchorDocPropsBlob"

"MSSChangeLogCookies"

"MSSCrawlReportCrawlErrors"

"MSSCrawlUrlChanges"

"MSSUserIDDocIDMap"

"MSSAnchorText"

"MSSTranTempTable0"

"MSSCrawlDeletedURL"

**Return Code Values**: MUST return the number of items in the table that corresponds to the value of @TableName.

| Value of @TableName | Table |
|---|---|
| **MssCrawlHostList** | MSSCrawlHostList (section 2.2.5.10) |
| **MSSCrawlHostsLog** | MSSCrawlHostsLog (section 2.2.5.11) |
| **MSSUserHosts** | MSSUserHosts (section 2.2.5.19) |
| **MSSCrawlUrlUsedContentSourceReport** | MSSCrawlUrlUsedContentSourceReport (section 2.2.5.23) |
| **MSSAnchorChangeLog** | MSSAnchorChangeLog (section 2.2.5.1) |
| **MSSAnchorPendingChangeLog** | MSSAnchorPendingChangeLog (section 2.2.5.15) |
| **MSSSocialDistance** | MSSSocialDistance (section 2.2.5.20) |
| **MSSTranTempTable1** | MSSTranTempTable1 (section 2.2.5.17) |
| **MSSCrawlChangedCommittedDocs** | MSSCrawlChangedCommittedDocs (section 2.2.5.3) |
| **MSSCrawlChangedDeletedDocs** | MSSCrawlChangedDeletedDocs (section 2.2.5.4) |
| **MSSCrawlChangedSourceDocs** | MSSCrawlChangedSourceDocs (section 2.2.5.5) |
| **MSSCrawlChangedTargetDocs** | MSSCrawlChangedTargetDocs (section 2.2.5.6) |
| **MSSCrawlLinksLog** | MSSCrawlLinksLog (section 2.2.5.12) |
| **MSSCrawlURL** | MSSCrawlURL (section 2.2.5.7) |
| **MSSCrawlURLReport** | MSSCrawlURLReport (section 2.2.5.14) |
| **MSSCrawlQueue** | MSSCrawlQueue (section 2.2.5.13) |
| **MSSCrawlReportCrawlErrors** | MSSCrawlReportCrawlErrors (section 2.2.5.21) |
| **MSSCrawlUrlChanges** | MSSCrawlUrlChanges (section 2.2.5.22) |
| **MSSAnchorText** | MSSAnchorText (section 2.2.5.2) |

| Value of @TableName | Table |
|---|---|
| **MSSTranTempTable0** | MSSTranTempTable0 (section 2.2.5.18) |
| **MSSCrawlDeletedURL** | MSSCrawlDeletedURL (section 2.2.5.9) |

**Result Sets**: MUST NOT return any result set.

### 3.1.5.43 proc_MSS_GetOldHostRule

The **proc_MSS_GetOldHostRule** stored procedure is called to determine if there is an existing host distribution rule in the current active crawl topology with the same host name, but different crawl store than the specified host distribution rule. If such an old host distribution rule is present, the output parameters MUST be populated. Otherwise, the protocol client MUST ignore the output parameters.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetOldHostRule (
    @HostName           nvarchar(250),
    @GthrDBID           int,
    @CurGthrDBID        int OUTPUT,
    @HostID             int OUTPUT
);
```

**@HostName:** The host name of the host distribution rule that needs to be verified.

**@GthrDBID:** The crawl store ordinal of the host distribution rule that needs to be verified.

**@CurGthrDBID:** Upon return from this stored procedure, this parameter MUST be set to the crawl store ordinal of the old host distribution rule if it exists.

**@HostID:** Upon return from this stored procedure, this parameter MUST be set to the identifier of the specified host name, if an old host distribution rule is found to exist.

**Return Code Values**: MUST return one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | An old host distribution rule exists for the specified host name. |
| **1** | There is no old host distribution rule for the specified host name. |

**Result Sets**: MUST NOT return any result set.

### 3.1.5.44 proc_MSS_GetPartitions

The **proc_MSS_GetPartitions** stored procedure is called to retrieve list of all index partitions that are associated with a query topology.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetPartitions (
    @PartitionSchemeID        uniqueidentifier
```

```
);
```

**@PartitionSchemeID:** The unique identifier of the query topology.

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** MUST return a result set as described in the section Index Partitions Result Set.

#### 3.1.5.44.1 Index Partitions Result Set

The Index Partitions Result Set MUST contain zero or more rows, each corresponding to a single index partition.

The T-SQL syntax for the result set is as follows:

```
PartitionSchemeID      uniqueidentifier  NOT NULL,
PartitionID            uniqueidentifier NOT NULL,
Ordinal                tinyint NOT NULL,
PropertyStoreID        uniqueidentifier NULL;
```

**PartitionSchemeID:** The unique identifier of the query topology.

**PartitionID:** The unique identifier of the index partition.

**Ordinal:** The integer ordinal of the index partition.

**PropertyStoreID:** The unique identifier of the metadata index the index partition is associated with in the given query topology.

### 3.1.5.45 proc_MSS_GetPartitionsMap

The **proc_MSS_GetPartitionsMap** stored procedure is called to retrieve a list of document distribution identifiers along with the associated index partitions for a query topology.

The T-SQL syntax for the stored procedure is as follows.

```
PROCEDURE proc_MSS_GetPartitionsMap(
    @PartitionSchemeID uniqueidentifier
);
```

**@PartitionSchemeID:** The unique identifier of the query topology.

**Return Code Values:** An integer which MUST be **0**.

**Result Sets:** The procedure MUST return Index Partitions Map Result Set. If there is no query topology with the specified identifier then the result set MUST be empty; otherwise, it MUST contain **256** rows of document distribution identifiers from **0** to **255**.

#### 3.1.5.45.1 Index Partitions Map Result Set

The Index Partitions Map Result Set MUST contain zero or more rows, each corresponding to a single document distribution identifier.

The T-SQL syntax for the result set is as follows:

```
PartitionID             uniqueidentifier NOT NULL,
Hash                    tinyint NOT NULL;
```

**PartitionID:** The unique identifier of the index partition.

**Hash:** The document distribution identifier.

### 3.1.5.46 proc_MSS_GetPartitionSchemes

The **proc_MSS_GetPartitionSchemes** stored procedure is called to retrieve list of all query topologies.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetPartitionSchemes ();
```

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** This procedure MUST return Query Topologies Result Set.

#### 3.1.5.46.1 Query Topologies Result Set

The Query Topologies Result Set MUST contain zero or more rows, each corresponding to a single query topology.

The T-SQL syntax for the result set is as follows:

```
PartitionSchemeID       uniqueidentifier NOT NULL,
CreationDate            datetime NOT NULL,
State                   smallint NOT NULL;
```

**PartitionSchemeID:** The unique identifier of the query topology.

**CreationDate:** The date and time the query topology was created.

**State:** The state of the query topology. The value MUST be a Query Topology State data type as specified in Section 2.2.1.2.

### 3.1.5.47 proc_MSS_GetPropertyStoreHashesForActiveScheme

The **proc_MSS_GetPropertyStoreHashesForActiveScheme** stored procedure is called to retrieve a list of document distribution identifiers for the active query topology along with corresponding identifiers of query topology, index partition and metadata index. The procedure MUST return data related to either the specified query component (2) or all query components (2). The results MUST be document distribution identifiers in ascending order.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetPropertyStoreHashesForActiveScheme (
    @ComponentId       int,
    @ThisPartitionOnly int
);
```

**@ComponentId:** The integer identifier of the query component (2) or NULL if document distribution identifiers for all query components needs to be returned.

**@ThisPartitionOnly:** An integer that defines the result set for the index partition that MUST be included in results. If *@ComponentId* is set to **NULL** then this parameter must be set to **0**; otherwise, this parameter must be set to one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | The result set MUST include information for all index partitions in the query topology the query component (2) belongs to. |
| **1** | The result set MUST include only information for the index partition with which the query component (2) is associated. |

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** MUST return a Document Distribution Identifiers Result Set.

### 3.1.5.47.1 Document Distribution Identifiers Result Set

The Document Distribution Identifiers Result Set MUST contain zero or more rows, each corresponding to a single document distribution identifier of an active query topology:

```
Hash                    tinyint NOT NULL,
Ordinal                 tinyint NOT NULL,
PartitionSchemeID       uniqueidentifier NOT NULL,
PartitionID             uniqueidentifier NOT NULL,
PropertyStoreID         uniqueidentifier NOT NULL;
```

**Hash:** The document distribution identifier.

**Ordinal:** The integer ordinal of the index partition.

**PartitionSchemeID:** The identifier of the query topology.

**PartitionID:** The identifier of the index partition.

**PropertyStoreID:** The identifier of the metadata index.

## 3.1.5.48 proc_MSS_GetPropertyStores

The **proc_MSS_GetPropertyStores** stored procedure is called to retrieve list of all metadata indexes.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetPropertyStores ();
```

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** This procedure MUST return Metadata Indexes Result Set.

#### 3.1.5.48.1 Metadata Indexes Result Set

The Metadata Indexes Result Set MUST contain zero or more rows, each corresponding to a single metadata index.

The T-SQL syntax for the result set is as follows:

```
PropertyStoreID        uniqueidentifier NOT NULL,
Name                   nvarchar(256) NOT NULL;
```

**PropertyStoreID:** The unique identifier of the metadata index.

**Name:** The name of the metadata index.

### 3.1.5.49 proc_MSS_GetQueryComponent

The **proc_MSS_GetQueryComponent** stored procedure is called to receive the current state of a query component (2).

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetQueryComponent (
    @QueryComponentID         uniqueidentifier
);
```

**@QueryComponentID:** The unique identifier of the query component.

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** This procedure MUST return a Query Component Result Set.

### 3.1.5.50 proc_MSS_GetQueryComponentHotSwap

The **proc_MSS_GetQueryComponentHotSwap** stored procedure is called to retrieve a Query Component Data Type (section 2.2.1.4).

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetQueryComponentHotSwap (
    @QueryComponentNumber int
)
```

**@QueryComponentNumber:** The integer identifier of the query component (2).

**Return Code Values:** If there is no query component (2) with the specified integer identifier **0** MUST be returned; otherwise, the value must be a Query Component Data Type (section 2.2.1.4).

**Result Sets:** SHOULD NOT<9> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.51 proc_MSS_GetQueryComponents

The **proc_MSS_GetQueryComponents** stored procedure is called to receive list of all query components(2).

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetQueryComponents ();
```

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** This stored procedure MUST return a Query Component Result Set.

### 3.1.5.52 proc_MSS_GetQueryComponentsForActivePartitionScheme

The **proc_MSS_GetQueryComponenForActivePartitionScheme** stored procedure is called to receive a list of all query components(2) that are associated with the query topology that is in the Active state.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetQueryComponentForActivePartitionScheme();
```

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** MUST return a Query Component Result Set as specified in section 2.2.4.2.

### 3.1.5.53 proc_MSS_GetQueryComponentsForPartitionScheme

The **proc_MSS_GetQueryComponentForPartitionScheme** stored procedure is called to receive a list of all query components(2) that are associated with the specified query topology.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetQueryComponentForPartitionScheme(
    @QueryComponentID         uniqueidentifier
);
```

**@QueryComponentID:** The unique identifier of the query topology.

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** MUST return a Query Component Result Set as described in section 2.2.4.2.

### 3.1.5.54 proc_MSS_GetRefactoringTask

The **proc_MSS_GetRefactoringTask** stored procedure is called to retrieve a refactoring task with the specified refactoring task identifier.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetRefactoringTask (
     @TaskID    int
);
```

**@TaskID:** The unique identifier of the refactoring task.

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** MUST return the **Refactoring Task Result Set** and **Refactoring Task Part Result Set** as specified in section 3.1.5.54.1 and 3.1.5.54.2

### 3.1.5.54.1 Refactoring Task Result Set

Returns a refactoring task with the specified refactoring task identifier. The result set MUST always be returned. If the store contains a refactoring task with the specified identifier, this result set MUST contain one row for that task; otherwise, the result set MUST be empty.

The T-SQL syntax for the result set is as follows:

```
TaskID                      int NOT NULL,
ActionID                    int NOT NULL,
TaskAssignedTime            datetime NOT NULL,
SourceComponentID           uniqueidentifier NOT NULL,
DestinationComponentID      uniqueidentifier NOT NULL,
TaskType                    nvarchar(256) NOT NULL,
CurrentDocID                int NOT NULL,
EndDocID                    int NOT NULL,
SuccessfullyCopied          int NOT NULL,
TotalToCopy                 int NOT NULL,
TaskState                   int NOT NULL,
ErrorDescription            nvarchar(1024) NOT NULL;
```

**TaskID:** The unique identifier of the refactoring task.

**ActionID:** The unique identifier of the topology activation action this task is a part of.

**TaskAssignedTime:** The date and time the refactoring task was assigned.

**SourceComponentID:** The unique identifier of the metadata index where data is being copied from.

**DestinationComponentID:** The unique identifier of the metadata index where data is being copied to.

**TaskType:** The type of the refactoring task. The value MUST be a Refactoring Task Type data type as specified in Section 2.2.1.10.

**CurrentDocID:** The document identifier of the document that was copied last for this refactoring task. MUST be set to -1 if no documents have been copied yet.

**EndDocID:** The document identifier of the last document that will be copied by this task. MUST be set to -1 if number of documents in the source database is not known yet.

**SuccessfullyCopied:** The number of documents that have been successfully processed for this refactoring task.

**TotalToCopy:** The total number of documents that need to be processed for this refactoring task.

**TaskState:** The state of the refactoring task. The value MUST be a Refactoring Task State data type as specified in Section 2.2.1.9.

**ErrorDescription:** Text description of the error occurred during execution of this refactoring task.

#### 3.1.5.54.2 Refactoring Task Part Result Set

Returns an unordered list of task parts with the specified task identifier. The result set MUST always be returned. If the store contains a refactoring task part with the specified identifier, this result set MUST contain one row for that task part; otherwise, the result set MUST be empty.

The T-SQL syntax for the stored procedure is as follows.

```
Part int NOT NULL,
```

**Part:** The task part retrieved be from a XML file.

### 3.1.5.55 proc_MSS_GetRefactoringTaskBatches

The **proc_MSS_GetRefactoringTaskBatches** stored procedure is called to retrieve all refactoring task batches that are associated with the specified refactoring task and for which the **StartDocID** is greater or equal the specified value.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetRefactoringTaskBatches (
    @TaskID            int,
    @StartDocID        bigint
);
```

**@TaskID:** The unique identifier of the refactoring task.

**@StartDocID:** The lower bound for the **StartDocID** of the requested refactoring task batches.

**Return Code Values:** An integer which MUST be one of the values listed in the following table.

| Value | Description |
|---|---|
| **0** | Successful execution. |

**Result Sets:** MUST return the **Refactoring Task Batches Result Set** as described in section 2.2.4.3.

### 3.1.5.56 proc_MSS_GetRefactoringTaskBatchesInfo

The **proc_MSS_GetRefactoringTaskBatchesInfo** stored procedure is called to retrieve information about the refactoring task batch that is associated with the specified refactoring task.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetRefactoringTaskBatchesInfo (
    @TaskID            int,
    @CurrentDocID      bigint OUTPUT,
    @LastScheduled     bigint OUTPUT
);
```

**@TaskID:** The unique identifier of the refactoring task.

**@CurrentDocID:** Upon return from this stored procedure, this parameter MUST be set to the EndDocID value (section 3.1.1.3) of the refactoring task batch that is associated with the specified refactoring task, if the State value (section 3.1.1.3) of that refactoring task batch is *Finished* (section 2.2.1.11). Otherwise this parameter MUST be set to -1.

**@LastScheduled:** Upon return from this stored procedure, this parameter MUST be set to the EndDocID value (section 3.1.1.3) of the refactoring task batch that is associated with the specified refactoring task.

**Return Code Values:** An integer which MUST be one of the values listed in the following table.

| Value | Description |
|---|---|
| **0** | Successful execution. |

**Result Sets:** SHOULD NOT<10> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.57 proc_IsCrawlStoreRefactoringTaskBatchCommitted

The **proc_MSS_IsCrawlStoreRefactoringTaskBatchCommitted** stored procedure is called to retrieve the information if the given refactoring task batch is committed.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_IsCrawlStoreRefactoringTaskBatchCommitted (
    @BatchID          int
    );
```

**@BatchID:** The unique identifier of the refactoring task batch.

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** SHOULD NOT<11> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.58 proc_CommittedCrawlStoreRefactoringTaskBatch

The **proc_MSS_CommittedCrawlStoreRefactoringTaskBatch** stored procedure is called to notify the system that the given refactoring task batch is committed.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CommittedCrawlStoreRefactoringTaskBatch (
    @BatchID          int
    );
```

**@BatchID:** The unique identifier of the refactoring task batch.

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** SHOULD NOT<12> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.59 proc_MSS_GetRefactoringTasks

The **proc_MSS_GetRefactoringTasks** stored procedure is called to retrieve all the refactoring tasks with the specified topology activation action identifier and refactoring task type.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetRefactoringTasks (
    @ActionID          int,
    @TaskType          nchar(256)
);
```

**@ActionID:** The unique identifier of the topology activation action this task is a part of.

**@TaskType:** The type of the refactoring task. The value MUST be a Refactoring Task Type data type as specified in Section 2.2.1.10.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Default return value |

**Result Sets:** MUST return the Refactoring Tasks Result Set as described in section 3.1.5.59.1

#### 3.1.5.59.1 Refactoring Tasks Result Set

Returns an unordered list of refactoring tasks with the specified action identifier and task type. The result set MUST always be returned. If the store contains refactoring tasks with the specified identifier and type, this result set MUST contain one row for each of those tasks; otherwise, the result set MUST be empty.

The T-SQL syntax for the stored procedure is as follows:

```
TaskID                      int NOT NULL,
ActionID                    int NOT NULL,
TaskAssignedTime            datetime NOT NULL,
SourceComponentID           uniqueidentifier NOT NULL,
DestinationComponentID      uniqueidentifier NOT NULL,
TaskType                    nvarchar(256) NOT NULL,
CurrentDocID                int NOT NULL,
EndDocID                    int NOT NULL,
SuccessfullyCopied          int NOT NULL,
TotalToCopy                 int NOT NULL,
TaskState                   int NOT NULL,
ErrorDescription            nvarchar(1024) NOT NULL;
```

**TaskID:** The unique identifier of the refactoring task.

**ActionID:** The unique identifier of the topology activation action this task is a part of.

**TaskAssignedTime:** The UTC time when this refactoring task was created.

**SourceComponentID:** The unique identifier of the metadata index where data is being copied from.

**DestinationComponentID:** The unique identifier of the metadata index where data is being copied to.

**TaskType:** The type of the refactoring task. The value MUST be a Refactoring Task Type data type as specified in Section 2.2.1.10.

**CurrentDocID:** The document identifier of the document that was copied last for this refactoring task. MUST be set to -1 if no documents have been copied yet.

**EndDocID:** The document identifier of the last document that will be copied by this task. MUST be set to -1 if number of documents in the source database is not known yet.

**SuccessfullyCopied:** The number of documents that have been successfully processed by for this refactoring task.

**TotalToCopy:** The total number of documents that need to be processed for this refactoring task.

**TaskState:** The state of the refactoring task. The value MUST be a Refactoring Task State data type as specified in Section 2.2.1.9.

**ErrorDescription:** Text description of the error occurred during execution of this refactoring task.

### 3.1.5.60 proc_MSS_GetRemovedRulesForCrawlStore

The **proc_MSS_GetRemovedRulesForCrawlStore** stored procedure is called to retrieve the list of host names for which there are administrative host distribution rules that have been marked for deletion, and are in the current active Crawl Topology (section 3.1.1.3) and described crawl store.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetRemovedRulesForCrawlStore (
    @Ordinal           int
);
```

**@Ordinal:** The ordinal of the crawl store.

**Return Code Values:** This stored procedure MUST return number of elements in the Host Identifier Result Set.

**Result Sets:** This stored procedure MUST return the Host Identifier Result Set.

#### 3.1.5.60.1 Host Identifier Result Set

The **Host Identifier Result Set** returns information about host names that have been marked for deletion in host distribution rules. The result set MUST have one record specifying the crawl store identifier for each administrative host distribution rule that is marked for deletion and is part of the current active topology. The T-SQL syntax for the result set is as follows:

```
HostID               int NOT NULL;
```

**HostID:** The identifier of the host name.

### 3.1.5.61 proc_MSS_GetRuleForHost

The **proc_MSS_GetRuleForHost** stored procedure is called to retrieve an administrative host distribution rule for the specified host name. The rule can either be a part of the current active crawl topology or not assigned to any crawl topology. Upon successful execution, the output parameters MUST be updated with the host name and crawl store ordinal of the retrieved administrative host distribution rule.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetRuleForHost (
    @HostID             int,
    @HostName           nvarchar(250) OUTPUT,
    @CrawlStoreID       int OUTPUT
);
```

**@HostID:** The identifier of the host name.

**@HostName:** Name of the host with identifier equals to @HostID. Upon successful return from this stored procedure, this parameter MUST be set to the host name of the host distribution rule.

**@CrawlStoreID:** Identifier of the crawl database. Upon successful return from this stored procedure, this parameter MUST be set to the crawl store identifier of the host distribution rule.

**Return Code Values:** MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. An administrative host distribution rule exists for the current active crawl topology. |
| **1** | No administrative host distribution rule exists. @CrawlStoreID MUST be ignored. |
| **2** | Successful execution. An administrative host distribution rule exists with a NULL crawl topology identifier. |
| **10** | The host name does not exist in the Host Set. @HostName and @CrawlStoreID MUST be ignored. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.62 proc_MSS_GetTopology

The **proc_MSS_GetTopology** stored procedure is used to get current state of the administration component.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetTopology ();
```

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** This procedure MUST return Administration Component Result Set

#### 3.1.5.62.1 Administration Component Result Set

The administration component result set MUST contain exactly one row.

The T-SQL syntax for the result set is as follows:

```
TopologyID                      int NOT NULL,
DesiredAdminServerName          nvarchar(256) NULL,
DesiredAdminServerID            uniqueidentifier NULL,
DesiredAdminLocalStoragePath    nvarchar(260) NULL,
DesiredStandalone               int NULL,
AdminServerName                 nvarchar(256) NULL,
AdminServerID                   uniqueidentifier NULL,
AdminLocalStoragePath           nvarchar(260) NULL,
Standalone                      int NULL,
LastLogCleanup                  datetime NOT NULL,
SettingsInRegistry              int NOT NULL;
```

**TopologyID:** This parameter MUST be set to 0, and it MUST be ignored by the client.

**DesiredAdminServerName:** Current value of the desired server name for the administration component as described in Section 3.1.1.1.

**DesiredAdminServerID:** Current value of the desired server identifier for the administration component as described in Section 3.1.1.1.

**DesiredAdminLocalStoragePath:** Current value of the desired local storage path for the administration component as described in Section 3.1.1.1. This value MUST be ignored by the client if the DesiredAdminServerName field contains the same value as AdminServerName.

**DesiredStandalone:** Current value of the desired type of the administration component as described in Section 3.1.1.1. This value MUST be ignored by the client if the DesiredAdminServerName field contains the same value as AdminServerName.

**AdminServerName:** The name of the server where the administration component is currently located. This value MUST be set to NULL if the administration component is not initialized.

**AdminServerID:** The unique identifier of the server where the administration component is currently located. This value MUST be set to NULL if the administration component is not initialized.

**AdminLocalStoragePath:** The local storage path for the administration component. This value MUST be set to NULL if the administration component is not initialized.

**Standalone:** The type of the administration component. This value MUST be set to NULL if the administration component is not initialized, otherwise the value MUST be an Administration Component Type data type as specified in Section 2.2.1.1.

**LastLogCleanup:** This value MUST be ignored by the client.

**SettingsInRegistry:** The value that specifies whether the system MUST use cached values to initialize the administration component. This value MUST be set to 0 if the administration component has never been initialized; otherwise, it MUST be set to 1.

### 3.1.5.63 proc_MSS_GetTopologyActivationActions

The **proc_MSS_GetTopologyActivationActions** stored procedure is called to receive list off all topology activation actions that are associated with a query topology or a crawl topology with the specified unique identifier.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_GetTopologyActivationActions (
    @TopologyID        uniqueidentifier
);
```

**@TopologyID:** The unique identifier of a query topology or a crawl topology.

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** MUST return a Topology Activation Action Result Set as described in section 3.1.5.63.1.

#### 3.1.5.63.1 Topology Activation Action Result Set

The Topology Activation Action Result Set MUST contains zero or more rows, each corresponding to a single topology activation action.

The T-SQL syntax for the result set is as follows:

```
ActionID          int NOT NULL,
Name              nvarchar(256) NOT NULL,
TopologyID        uniqueidentifier NOT NULL,
State             smallint NOT NULL,
StartedTime       datetime NULL,
FinishedTime      datetime NULL;
```

**ActionID:** The unique identifier of the topology activation action.

**Name:** The name of the topology activation action.

**TopologyID:** The unique identifier of the query topology or the crawl topology this topology activation action is associated with.

**State:** The state of the topology activation action. This value MUST be a Topology Activation Action State data type as specified in Section 2.2.1.8.

**StartedTime:** The UTC date when the state of the topology activation action was set to Started (section 2.2.1.8**)**. If execution of the topology activation action hasn't been started this field MUST be set to NULL.

**FinishedTime:** The UTC date when the state of the topology activation action was set to Finished (section 2.2.1.8). If execution of the topology activation action hasn't been finished this field MUST be set to NULL.

### 3.1.5.64 proc_MSS_InitRefactoringTask

The **proc_MSS_InitRefactoringTask** stored procedure is called to update a refactoring task with the specified refactoring task identifier

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_ReportRefactoringTask (
    @TaskID                 int,
    @EndDocID               int,
    @TotalToCopy            int
);
```

**@TaskID:** The unique identifier of the refactoring task.

**@EndDocID:** The document identifier of the last document that will be copied by this task: MUST be set to **-1** if the number of documents in the source database is not yet known.

**@TotalToCopy:** The total number of documents that need to be processed for this refactoring task.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | Refactoring task with the specified identifier does not exist. |

**Result Sets:** SHOULD NOT<13> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.65 proc_MSS_MakeCrawlStoreShared

The **proc_MSS_MakeCrawlStoreShared** stored procedure is called to set the type of a specified crawl store to Dedicated (see Section 2.2.1.13). If there is no crawl store with the specified identifier is not valid or the type of specified crawl store is already set to Dedicated, then the stored procedure MUST NOT change any crawl store.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_MakeCrawlStoreShared (
    @CrawlStoreID      uniqueidentifier
);
```

**@CrawlStoreID:** The identifier of the crawl store.

**Return Code Values:** An integer which MUST be **0**.

**Result Sets:** SHOULD NOT<14> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.66 proc_MSS_MoveHostsWithNoDocuments

This **proc_MSS_MoveHostsWithNoDocuments** stored procedure is called to move all the hosts from the Admin Host Set which do not have a record in either the Automatic Host Distribution Rule Set (section 3.1.1.5) or a record in the Administrative Host Distribution Rule Set (section 3.1.1.5), for the topology specified by the parameter @ActivatingTopologyID, to a new crawl store.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_MoveHostsWithNoDocuments (
    @ActivatingTopologyID     uniqueidentifier,
    @GthrDBID_from            int
);
```

**@ActivatingTopologyID:** The identifier of the current activating crawl topology.

**@GthrDBID_from:** The ordinal of the crawl store from which the data will be moved.

**Return Code Value:** An integer which MUST be **0**.

**Result Sets:** SHOULD NOT<15> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure

### 3.1.5.67 proc_MSS_MoveHostToDB

This **proc_MSS_MoveHostToDB** stored procedure is called to move the specified host name to a new crawl store. The stored procedure MUST add an Automatic Host Distribution Rule Set (section 3.1.1.5) with a new crawl store for the specified host name and crawl topology. If the value of @NeedToRefactor is greater than zero, the stored procedure MUST also add a Crawl Store Refactoring Task Set (section 3.1.1.4) to move data from the described crawl store to the new crawl store.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_MoveHostToDB (
    @HostID                   int,
    @GthrDBID_from            int,
    @ActivatingTopologyID     uniqueidentifier,
    @NumDocs                  int,
    @NeedToRefactor           int
);
```

**@HostID:** The identifier of the host name.

**@GthrDBID_from:** The ordinal of the crawl store from which the data will be moved.

**@ActivatingTopologyID:** The identifier of the current activating crawl topology.

**@NumDocs:** An integer representing the number of documents crawled for the specified host name.

**@NeedToRefactor**: An integer which indicates if a crawl store refactoring is being performed. A value of 0 or less indicates that no refactoring is being done, while a value of 1 or greater indicates that a refactoring is being performed.

**Return Code Value:** The ordinal of the crawl store to which the data is moved.

**Result Sets:** SHOULD NOT<16> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.68 proc_MSS_NeedToMoveDataFromDedicatedCrawlStores

The **proc_MSS_NeedToMoveDataFromDedicatedCrawlStores** stored procedure is called to determine if within the active crawl topology exists an administrative host distribution rule that is marked for deletion and is associated with a crawl store of type *Dedicated* (see Section 2.2.1.13).

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_NeedToMoveDataFromDedicatedCrawlStores ();
```

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | There is no data to move out of a dedicated crawl store. |
| **1** | There is data to move out of a dedicated crawl store. |

**Result Sets**: The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.69 proc_MSS_NumberOfDocumentsForRefactoringTask

The **proc_MSS_NumberOfDocumentsForRefactoringTask** stored procedure is called to retrieve the number of items in the specified set for the specified host.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_NumberOfDocumentsForRefactoringTask (
    @HostId             int,
    @TableIndex         int

);
```

**@HostId:** The identifier of the host name.

**@TableIndex:** An integer that MUST be set to one of the values in the following table:

| Value | Description |
|---|---|
| **0** | The ordered set contains document identifiers from Crawl URL History (specified in [MS-SQLPGAT2] section 3.1.1.2). |
| **1** | The ordered set contains link identifiers from Anchor Text Info (specified in [MS-SQLPGAT2] section 3.1.1.8). |
| **2** | The ordered set contains link identifiers from Links( specified in [MS-SQLPGAT2] section 3.1.1.5) |
| **3** | The ordered set contains TrackIDs from Deleted URL (specified in [MS-SQLPGAT2] section 3.1.1.3) |

**Return Code Values:** This stored procedure MUST return an integer with the following value, depending on the value of @TableIndex:

| Value of @TableIndex | Value |
|---|---|
| **0** | The number of items in Crawl URL History (specified in [MS-SQLPGAT2] section 3.1.1.2) with HostID equal to @HostID. |
| **1** | The number of items of Anchor Text Info (specified in [MS-SQLPGAT2] section 3.1.1.8) with HostID equal to @HostID. |
| **2** | The number of Links( specified in [MS-SQLPGAT2] section 3.1.1.5) with HostID equal to @HostID. |
| **3** | The number of items in Deleted URL (specified in [MS-SQLPGAT2] section 3.1.1.3), with HostID equal to @HostID. |

**Result Sets:** SHOULD NOT<17> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.70 proc_MSS_RegisterCrawlStore

The **proc_MSS_RegisterCrawlStore** stored procedure is called to register a new crawl store in a list of all available crawl stores.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_RegisterCrawlStore (
    @Name               nvarchar(256),
    @CrawlStoreId       uniqueidentifier,
    @IsDedicated        int
);
```

**@Name:** The name of the crawl store.

**@CrawlStoreId:** The identifier of the crawl store.

**@IsDedicated:** The crawl store type as described in section 2.2.1.13.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | The crawl store with the specified identifier is already registered. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.71 proc_MSS_RegisterPropertyStore

The **proc_MSS_RegisterPropertyStore** stored procedure is called to add a new metadata index.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_RegisterPropertyStore (
    @Name               nvarchar(256),
    @PropertyStoreID    uniqueidentifier
```

```
);
```

**@Name:** The name of the metadata index.

**@PropertyStoreID:** The unique identifier for the metadata index.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Metadata index has been added successfully. |
| **1** | Metadata index with the given unique identifier already exists. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.72 proc_MSS_RemoveCrawlStoreRefactoringTasks

The **proc_MSS_RemoveCrawlStoreRefactoringTasks** stored procedure clears the list of host names that need to be moved between crawl stores during activation of the specified crawl topology.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_RemoveCrawlStoreRefactoringTasks (
    @CrawlTopologyID          uniqueidentifier
);
```

**@CrawlTopologyID:** The unique identifier of the crawl topology.

**Return Code Values:** This stored procedure MUST return **0** upon completion.

**Result Sets:** SHOULD NOT<18> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.73 proc_MSS_RemoveHostDistributionRule

The **proc_MSS_RemoveHostDistributionRule** stored procedure is called to remove the specified administrative host distribution rule. If an administrative host distribution rule exists with the specified host name and crawl store identifier, then it MUST be marked for deletion if it is part of the current active crawl topology, and it MUST be deleted if it has a **NULL** crawl topology identifier.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_RemoveHostDistributionRule (
    @HostName          nvarchar(100),
    @GthrDBGuid        uniqueidentifier
);
```

**@HostName:** The host name of the host distribution rule.

**@GthrDBGuid:** The identifier of the crawl store of the host distribution rule.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | The host distribution rule was successfully removed if it existed. |
| **1** | There is no crawl store with the specified identifier, no rules were deleted. |

**Result Sets**: SHOULD NOT<19> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.74 proc_MSS_ReportAdminComponentState

The **proc_MSS_ReportAdminComponentState** stored procedure is called to update the current server, the current local storage path, and the current administration component type (section 2.2.1.1).

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_ReportAdminComponentState (
    @AdminServerName          nvarchar(256),
    @AdminServerID            uniqueidentifier,
    @AdminLocalStoragePath    nvarchar(260),
    @Standalone               int,
    @SettingsInRegistry       int
);
```

**@AdminServerName:** The name of the server where the administration component is currently located. If the administration component is currently uninitialized then this parameter MUST be set to NULL.

**@AdminServerID:** The unique identifier of the server where the administration component is currently located. If the administration component is currently uninitialized then this parameter MUST be set to NULL.

**@AdminLocalStoragePath:** The current local storage path for the administration component. If the administration component is currently uninitialized then this parameter MUST be set to NULL.

**@Standalone:** The current administration component type (section 2.2.1.1).

**@SettingsInRegistry:** The value that specifies whether the system MUST use cached values to initialize the administration component. This value MUST be set to 1.

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** SHOULD NOT<20> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.75 proc_MSS_ReportCrawlComponentState

The **proc_MSS_ReportCrawlComponentState** stored procedure is called to change the state of the crawl component.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_ReportCrawlComponentState (
    @CrawlComponentId         uniqueidentifier,
```

```
    @State                  int
);
```

**@CrawlComponentId:** The identifier of the crawl component.

**@State:** The new state of the crawl component (see 2.2.1.7 for the list of available component states).

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** SHOULD NOT<21> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.76 proc_MSS_ReportCurrentDocID

The **proc_MSS_ReportCurrentDocID** stored procedure is called to update a refactoring task with the specified document identifier information.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_ReportCurrentDocID (
    @TaskID                 int,
    @CurrentDocID           bigint,
    @TaskState              int
);
```

**@TaskID:** The unique identifier of the refactoring task.

**@CurrentDocID:** The document identifier of the document that was copied last for this refactoring task: MUST be set to **-1** if no documents have been copied yet.

**@TaskState:** The state of the refactoring task. The value MUST be a Refactoring Task State data type as specified in Section 2.2.1.9.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | Refactoring task with the specified identifier does not exist. |

**Result Sets:** SHOULD NOT<22> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.77 proc_MSS_ReportRefactoringTask

The **proc_MSS_ReportRefactoringTask** stored procedure is called to update a refactoring task with the specified refactoring task identifier.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_ReportRefactoringTask (
    @TaskID                 int,
```

```
    @CurrentDocID             int,
    @EndDocID                 int,
    @SuccessfullyCopied       int,
    @TotalToCopy              int,
    @TaskState                int
);
```

**@TaskID:** The unique identifier of the refactoring task.

**@CurrentDocID:** The document identifier of the document that was copied last for this refactoring task: MUST be set to **-1** if no documents have been copied yet.

**@EndDocID:** The document identifier of the last document that will be copied by this task: MUST be set to **-1** if number of documents in the source database is not known yet.

**@SuccessfullyCopied:** The number of documents that have been successfully processed by for this refactoring task.

**@TotalToCopy:** The total number of documents that need to be processed for this refactoring task.

**@TaskState:** The state of the refactoring task. The value MUST be a Refactoring Task State data type as specified in Section 2.2.1.9.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | Refactoring task with the specified identifier does not exist. |

**Result Sets:** SHOULD NOT<23> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.78 proc_MSS_ReportRefactoringTaskBatch

The **proc_MSS_ReportRefactoringTaskBatch** stored procedure is called to update the execution state of a refactoring task batch with the specified refactoring task batch identifier. The stored procedure MUST set the HeartbeatTime field in the Refactoring Task Batch Set (section 3.1.1.4) for the specified refactoring task batch to the current UTC time.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_ReportRefactoringTaskBatch (
    @BatchID          int,
    @NewState         smallint
);
```

**@BatchID:** The unique identifier of the refactoring task batch.

**@NewState:** The updated state of the refactoring task batch. The value MUST be a Refactoring Task Batch State data type as specified in Section 2.2.1.11.

If the value of @NewState is *Finished* (section 2.2.1.11), the value of SuccessfullyCopied for the refactoring task identified by TaskID for the refactoring task batch MUST be increased by the NumOfDocs value of the refactoring task batch.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution |
| **1** | Refactoring task batch with the specified identifier does not exist. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.79 proc_MSS_ReportRefactoringTaskBatchError

The **proc_MSS_ReportRefactoringTaskBatchError** stored procedure is called to update the error description of a refactoring task batch with the specified refactoring task batch identifier.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_ReportRefactoringTaskBatchError (
    @BatchID             int,
    @ErrorDescription    nvarchar(1024)
);
```

**@BatchID:** The unique identifier of the refactoring task batch.

**@ErrorDescription:** Text description of the error occurred during execution of this refactoring task batch.

**Return Code Values:**

| Value | Description |
|---|---|
| **1** | Refactoring task batch with the specified identifier does not exist. |
| **0** | Successful execution. |

**Result Sets: MUST NOT return any result sets.**

### 3.1.5.80 proc_MSS_SetAdminComponentServer

The **proc_MSS_SetAdminComponentServer** stored procedure is called to update the server name and the desired server name for the administration component after the server name is changed. The stored procedure MUST update the current server name for the Administration Component (section 3.1.1.1) with the specified value.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_SetAdminComponentServer (
    @AdminServerName         nvarchar(256),
    @AdminServerID           uniqueidentifier,
    @DesiredAdminServerName  nvarchar(256),
    @DesiredAdminServerID    uniqueidentifier
```

```
);
```

**@AdminServerName:** The new name of the server where the Administration Component (section 3.1.1.1) is located.

**@AdminServerID:** MUST be set to **NULL**.

**@DesiredServerName:** The new name of the desired server for the Administration Component (section 3.1.1.1).

**@DesiredAdminServerID:** MUST be set to **NULL**.

**Return Code Values:** An integer which MUST be **0**.

**Result Sets:** SHOULD NOT<24> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.81 proc_MSS_SetConfigurationPropertyEx

The **proc_MSS_SetConfigurationPropertyEx** stored procedure is called to update a configuration property record of a search service application. It can be forced to delete the existing record and recreate one if needed.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_SetConfigurationPropertyEx (
    @Name                   nvarchar(300),
    @Value                  sql_variant,
    @AlwaysRecreateRecord   int
);
```

**@Name:** The name of the configuration property.

**@Value:** The value corresponding to the name of the configuration property.

**@AlwaysRecreateRecord:** An integer indicating if it needs to delete an existing record with the same name and value pair and recreate a new one. This parameter MUST be set to an integer that is one of the values listed in the following table:

| Value | Description |
|---|---|
| **1** | Delete an existing record with the same name and value pair (if there is one) and create a new record. |
| **0** | Update an existing record with the same name and value pair |

**Return Code Values:** This stored procedure returns an integer value that MUST be ignored.

**Result Sets:** MUST NOT return any result set.

### 3.1.5.82 proc_MSS_SetCrawlComponentServer

The **proc_MSS_SetCrawlComponentServer** stored procedure is called to update the server name for a crawl component after the server name is changed. The stored procedure MUST update the current server name for the specified crawl component (section 3.1.1.3) with the specified value.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_SetCrawlComponentServer (
    @CrawlComponentID uniqueidentifier,
    @ServerName nvarchar(256),
    @ServerID uniqueidentifier
);
```

**@CrawlComponentID:** The unique identifier of the crawl component to be updated.

**@ServerName:** The new name of the server where the crawl component is located.

**@ServerID:** MUST be set to **NULL**.

**Return Code Values:** An integer that MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | There is no crawl component with the specified unique identifier. |

**Result Sets:** SHOULD NOT<25> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.83 proc_MSS_SetCrawlTopologyState

The **proc_MSS_SetCrawlTopologyState** stored procedure is called to change the state of a crawl topology. The stored procedure MUST change state of the specified crawl topology only if that is an allowed state change as described in Section 3.1.1.3, otherwise corresponding error code MUST be returned. Beside that the stored procedure MUST follow the following rules:

1. To set the *Activating* state:
    1. There MUST be no crawl topologies or query topologies that are in the *Activating* or *Deactivating* states.
    2. There MUST be at least one crawl component associated with the crawl topology.
    3. All topology activation actions that are associated with the specified crawl topology MUST be deleted together with all refactoring tasks and refactoring task batches created for these topology activation actions.
    4. Once the *Activating* state is set the procedure MUST assign each crawl store associated with at least one crawl component in that crawl topology a temporary (subsequent) unique integer identifier in the range of [0..N-1] where N is a number of crawl stores associated with at least one crawl component this crawl topology.
2. To set the *Active* state:
    1. If there is another crawl topology with the *Active* state then the state of the old *Active* topology MUST be set to *Deactivating*. If there is another crawl topology with the *ActiveToBeRemoved* state then the state of the old *ActiveToBeRemoved* topology MUST be set to *DeactivatingToBeRemoved*. All topology activation actions that are associated with this crawl topology MUST be deleted together with all the refactoring tasks and refactoring task batches created for these topology activation actions.

2. Once the *Active* state is set the procedure MUST copy subsequent integer identifiers of the crawl stores to permanent ones and set subsequent identifiers for all crawl stores to NULL.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_SetCrawlTopologyState (
    @CrawlTopologyId         uniqueidentifier,
    @NewState                int,
    @Force                   bit
);
```

**@CrawlTopologyId:** The identifier of the crawl topology.

**@NewState:** The new state of the crawl topology. The value MUST be a Crawl Topology State data type as specified in section 2.2.1.6.

**@Force:** Value that specified whether the stored procedure MUST change the state of the crawl topology, even if it is not an allowed change according to section 3.1.1.3. A bit which MUST be **0**.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | The crawl topology doesn't exist. |
| **2** | The *Activating* state cannot be set because the current state is not *Inactive*. |
| **3** | There is another crawl topology or query topology that is in an *Activating* or *Deactivating* state. |
| **4** | Cannot set the *Activating* state for the crawl topology because it does not have any crawl components. |
| **6** | The *Active* state cannot be set because the current state is not *Activating*. |
| **7** | The *Inactive* state cannot be set because the current state is neither *Deactivating* nor *DeactivatingToBeRemoved*. |
| **8** | The *Deactivating* state cannot be set because the current state is not *Activating*. |
| **9** | The *ActiveToBeRemoved* state cannot be set because the current state is not *Active*. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.84 proc_MSS_SetNumberOfRows

The **proc_MSS_SetNumberOfRows** stored procedure is called to record the number of rows in the specified table in the MSSRefactoringStatistics table (section 2.2.5.24). This MUST update the MSSRefactoringStatistics table so that the statistic where TableName is equal to @TableName is set to the value of @NumOfRows.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_SetNumberOfRows (
    @TableName               nvarchar(256),
    @NumOfRows               int
```

```
);
```

**@TableName:** The name of the table whose rows will be recorded.

**@NumOfRows:** The number of rows in the named table at the time of the call to this stored procedure is called.

**Return Code:** An integer which MUST be **0**.

**Result Sets:** MUST NOT return any result set.

### 3.1.5.85 proc_MSS_SetPartitionPropertyStore

The **proc_MSS_SetPartitionPropertyStore** stored procedure is called to associate the metadata index with the specified index partition and query topology. The query topology MUST be *Inactive* and associated with the specified index partition.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_SetPartitionPropertyStore (
    @PartitionSchemeId        uniqueidentifier,
    @PartitionId              uniqueidentifier,
    @PropertyStoreId          uniqueidentifier
);
```

**@PartitionSchemeId:** The identifier of the query topology.

**@PartitionId:** The identifier of the index partition.

**@PropertyStoreId:** The identifier of the metadata index.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | The query topology is not *Inactive* or doesn't exist. |
| **2** | The metadata index doesn't exist. |
| **4** | The query topology is not associated with the index partition. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.86 proc_MSS_SetPartitionSchemeState

The **proc_MSS_SetPartitionSchemeState** stored procedure is called to change the state of a query topology. The stored procedure MUST change state of the specified query topology only if that is an allowed state change as defined in Section 3.1.1.2, otherwise corresponding error code MUST be returned. Beside that the stored procedure MUST follow the following rules:

1. To set the *Activating* state:

   - Each index partition MUST have a query component (2) associated with it.

- Each index partition MUST have a metadata index associated with it.
- There MUST be no crawl topologies or query topologies that are in an *Activating* or *Deactivating* state.
- All topology activation actions that are associated with the specified query topology MUST be deleted together with all refactoring tasks and refactoring task batches created for these topology activation actions.

2. To set the *Active* state:
   - If there is another query topology with the *Active* state then the state of the old *Active* topology MUST be set to *Deactivating*. All topology activation actions that are associated with this query topology MUST be deleted together with all refactoring tasks and refactoring task batches created for these topology activation actions.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_SetCrawlTopologyState (
    @PartitionSchemeId        uniqueidentifier,
    @NewState                 int,
    @Force                    bit
);
```

**@PartitionSchemeID:** The identifier of the query topology.

**@NewState:** The new state of the query topology. The value MUST be a Query Topology State data type as described in section 2.2.1.2.

**@Force:** A bit which MUST be **0**.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | The query topology doesn't exist. |
| **2** | The *Activating* state cannot be set because the current state is not *Inactive*. |
| **3** | Not all index partitions have query components (2) associated with them. |
| **4** | Not all index partitions have metadata index associated with them. |
| **5** | There is another crawl topology or query topology that is in *Activating* or *Deactivating* state. |
| **7** | The *Active* state cannot be set because the current state is not *Activating*. |
| **8** | The *Inactive* state cannot be set because the current state is not *Deactivating*. |
| **9** | *Deactivating* state cannot be set because the current state is not *Activating*. |

**Result Sets:** MUST NOT return any result set.

### 3.1.5.87 proc_MSS_SetQueryComponent

The **proc_MSS_SetQueryComponent** stored procedure is called to update the current state of a query component (2).

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_SetQueryComponent (
    @QueryComponentID         uniqueidentifier,
    @LocalStoragePath         nvarchar(260),
    @HotSwap                  int,
    @ShareName                nvarchar(260),
    @UsesCustomShare          int,
    @DesiredState             int,
    @State                    int,
    @TransitionSequenceName   nvarchar(260),
    @TransitionStep           int,
    @TransitionStatus         int,
    @TransitionError          nvarchar(2048),
    @TransitionCancelRequest  int,
    @SourceComponentID        uniqueidentifier,
    @SourceComponentPath      nvarchar(260),
    @PauseRequested           int,
    @SettingsInRegistry       int
);
```

**@QueryComponentID:** The unique identifier of the query component (2) to be updated.

**@LocalStoragePath:** MUST be set to NULL.

**@HotSwap:** If this parameter is not set to NULL then it MUST be a Query Component Type data type (section 2.2.1.4), and the stored procedure MUST update type of the query component.

**@ShareName:** If not set to NULL then the stored procedure MUST update the shared folder name for the query component.

**@UsesCustomShare:** MUST be set to NULL.

**@DesiredState:** If this parameter is not set to NULL then it MUST be a Query Component State data type (section 2.2.1.3), and the stored procedure MUST update the desired state of the query component with the given value.

**@State:** If this parameter is not set to NULL then it MUST be a Query Component State data type (section 2.2.1.3), and the stored procedure MUST update the state of the query component with the given value.

**@TransitionSequenceName:** If this parameter is not set to NULL then the stored procedure MUST set the TransitionSequenceName value of the query component (section 3.1.1.2) to the given value.

**@TransitionStep:** If this parameter is not set to NULL then the stored procedure MUST set the TransitionStep value of the query component (section 3.1.1.2) to the given value.

**@TransitionStatus:** If this parameter is not set to **NULL** then it MUST be a Query Component Transition Status data type (section 2.2.1.5), and the stored procedure MUST set the TransitionStatus value of the query component (section 3.1.1.2) to the given value.

**@TransitionError:** If this parameter is not set to NULL then the stored procedure MUST set the TransitionError value of the query component (section 3.1.1.2) to the given value.

**@TransitionCancelRequest:** If this parameter is not set to NULL then it MUST be set to either 0 or 1, and the stored procedure MUST set the TransitionCancelRequested value of the query component (section 3.1.1.2) to the given value.

**@SourceComponentID:** If this parameter is not set to NULL the stored procedure MUST update the unique identifier of the source query component (section 3.1.1.2) with the given value.

**@SourceComponentPath:** If this parameter is not set to NULL the stored procedure MUST update the source Application directory (section 3.1.1.2) with the given value.

**@PauseRequested:** If this parameter is not set to NULL then it MUST be set to either **0** or **1** and the stored procedure MUST update the **PauseRequested** field (see section 3.1.1.2) for the given query component (2) with the given value.

**@SettingsInRegistry:** MUST be set to NULL.

**Return Code Values:** An integer that MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | There is no query component (2) with the specified unique identifier. |

**Result Sets:** SHOULD NOT<26> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.88 proc_MSS_SetQueryComponentServer

The **proc_MSS_SetQueryComponentServer** stored procedure is called to update the server name for a query component (2) after the server name is changed. The stored procedure MUST update the current server name for the specified query component (2) (section 3.1.1.2) with the specified value.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_SetQueryComponentServer (
    @QueryComponentID uniqueidentifier,
    @ServerName nvarchar(256),
    @ServerID uniqueidentifier
);
```

**@QueryComponentID:** The unique identifier of the query component (2) to be updated.

**@ServerName:** The new name of the server where the query component (2) is located.

**@ServerID:** MUST be set to NULL.

**Return Code Values:** An integer that MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | There is no query component (2) with the specified unique identifier. |

**Result Sets:** SHOULD NOT<27> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.89 proc_MSS_SetTopologyIDForUncommittedRules

The **proc_MSS_SetTopologyIDForUncommittedRules** stored procedure is called to set the crawl topology identifier for any administrative host distribution rules that have not yet been marked as part of a crawl topology. On successful execution, all host distribution rules with a NULL crawl topology identifier MUST have their identifier set to the specified crawl topology identifier. If a host name exists such that it has both an automatic host distribution rule and an administrative host distribution rule for the specified crawl topology, the automatic host distribution rule MUST be deleted. For the specification of the Host Distribution Rule Set see Section 3.1.1.5.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_SetTopologyIDForUncommittedRules (
    @CrawlTopologyID    uniqueidentifier
);
```

**@CrawlTopologyID:** The identifier of the crawl topology.

**Return Code Values**: An integer that MUST return **0** upon completion.

**Result Sets**: SHOULD NOT<28> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.90 proc_MSS_CompleteRulesDeletion

The **proc_MSS_CompleteRulesDeletion** is called to finish the process of deletion of administrative host distribution rules.

The stored procedure MUST create an automatic host distribution rule for every administrative host distribution rule being deleted. The stored procedure MUST then assign the automatic host distribution rule to the same database as the administrative host distribution rule and MUST delete all administrative host distribution rules which are being deleted from the set of administrative host distribution rules.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_CompleteRuleDeletion ();
```

**Return Code Values**: An integer that MUST return **0** upon completion.

**Result Sets:** SHOULD NOT return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.91 proc_MSS_UpdateCrawlComponent

The **proc_MSS_UpdateCrawlComponent** stored procedure is called to set a new desired state and a master property for the specified crawl component.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_UpdateCrawlComponent (
    @CrawlComponentId         uniqueidentifier,
    @Master                   int,
    @DesiredState             int
);
```

**@CrawlComponentId:** The identifier of the crawl component.

**@Master:** An integer which MUST be one of the values listed in the following table.

| Value | Description |
|---|---|
| **0** | Not a master crawl component. |
| **1** | A master crawl component. |

**@DesiredState:** The desired state of the crawl component as described in section 2.2.1.7.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | The specified crawl component doesn't exist. |

**Result Sets:** SHOULD NOT<29> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.92 proc_MSS_UpdateCrawlStoreIdAfterRestore

The **proc_MSS_UpdateCrawlStoreIdAfterRestore** stored procedure is called to change unique identifier and name of a crawl store (section 3.1.1.3).

The T-SQL syntax for the stored procedure is as follows:

```
CREATE_PROC(proc_MSS_UpdateCrawlStoreIdAfterRestore) (
    @CrawlStoreID      uniqueidentifier,
    @NewCrawlStoreID   uniqueidentifier,
    @NewName           nvarchar(256)
);
```

**@CrawlStoreID:** The current unique identifier of the crawl store.

**@NewCrawlStoreID:** The new unique identifier for the crawl store.

**@NewName:** The new name for the crawl store.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | A crawl store with unique identifier that is the same as the value passed in parameter **@NewCrawlStoreID** already exists. |
| **2** | The specified crawl store doesn't exist. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.93 proc_MSS_UpdatePartitionsMap

The **proc_MSS_UpdatePartitionsMap** stored procedure is called to change the mapping between document distribution identifiers and index partitions for one or multiple index partitions. The stored procedure receives an XML document containing the mapping between document distribution identifiers and index partitions. It MUST replace all entries that exist for the specified index partitions in the Index Partition Hash Set (section 3.1.1.2) with the specified values of document distribution identifiers.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_UpdatePartitionsMap (
    @PartitionsMapXml ntext
);
```

**@PartitionsMapXml:** An XML document that contains new mapping between the document distribution identifiers and index partitions. This parameter MUST adhere to the PartitionsMap Schema (Section 2.2.6.4.2).

**Return Code Values:** An integer which MUST be **0**.

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.94 proc_MSS_UpdatePropertyStoreIdAfterRestore

The **proc_MSS_UpdatePropertyStoreIdAfterRestore** stored procedure is called to change the unique identifier and name of a metadata index.

The T-SQL syntax for the stored procedure is as follows:

```
CREATE_PROC(proc_MSS_UpdatePropertyStoreIdAfterRestore) (
    @PropertyStoreID      uniqueidentifier,
    @NewPropertyStoreID   uniqueidentifier,
    @NewName              nvarchar(256)
);
```

**@PropertyStoreID:** The current unique identifier of the metadata index.

**@NewPropertyStoreID:** The new unique identifier for the metadata index.

**@NewName:** The new name for the metadata index.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | A metadata index with the same unique identifier as the value passed into the parameter **@NewPropertyStoreID** already exists. |
| **2** | The specified metadata index doesn't exist. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.95 proc_MSS_ResetMasterRole

The **proc_MSS_ResetMasterRole** stored procedure MUST mark every crawl component as not being a "master component", that means it MUST set the Master property to **0** for all crawl components in the crawl topology; the crawl topology MUST be in the *Active* state.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_ResetMasterRole ();
```

**Return Code Values:** An integer which MUST be **0**.

**Result Sets:** SHOULD NOT<30> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.96 proc_MSS_UpdateRefactoringTaskBatchServer

The **proc_MSS_UpdateRefactoringTaskBatchServer** stored procedure is called to reassign a refactoring task batch to a different server. If the state of the specified refactoring task batch is not set to *Finished*, then the stored procedure MUST reassign the refactoring task batch to the server with the specified name, it MUST set the AssignedTime to current UTC time, ErrorCount to zero, LastErrorDescription to **NULL**, and LastErrorTime to **NULL** for the specified refactoring task batch (see Section 3.1.1.4). If the state of the refactoring task batch is set to "Finished", the stored procedure MUST NOT change the specified refactoring task batch.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_UpdateRefactoringTaskBatchServer (
    @BatchID           int,
    @ServerName        nvarchar(256)
);
```

**@BatchID:** The unique identifier of the refactoring task batch to be updated.

**@ServerName:** The name of the server the refactoring task batch is assigned to.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |

| Value | Description |
|---|---|
| **1** | Refactoring task batch with the specified identifier does not exist. |

**Result Sets:** MUST NOT return any result sets.

### 3.1.5.97 proc_MSS_UpdateTopology

The **proc_MSS_UpdateTopology** stored procedure is called to update the desired server name, desired server identifier, desired local storage path, and desired type for the administration component as defined in Section 3.1.1.1.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_UpdateTopology (
   @DesiredAdminServerName           nvarchar(256),
   @DesiredAdminServerID             uniqueidentifier,
   @DesiredAdminLocalStoragePath     nvarchar(260),
   @DesiredStandalone                int,
   @SettingsInRegistry               int
);
```

**@DesiredAdminServerName:** New value for desired server name for the administration component.

**@DesiredAdminServerID:** New value for desired server identifier for the administration component.

**@DesiredAdminLocalStoragePath:** New value for the desired local storage path for the administration component.

**@DesiredStandalone:** New value for the desired type of the administration component. This parameter MUST be an Administration Component Type data type as specified in Section 2.2.1.1

**@SettingsInRegistry:** This value MUST be set to **NULL**.

**Return Code Values:** An integer that MUST be **0**.

**Result Sets:** SHOULD NOT<31> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

### 3.1.5.98 proc_MSS_UpdateTopologyActivationAction

The **proc_MSS_UpdateTopologyActivationAction** stored procedure is called to update the state of a topology activation action. If the client attempts to call the stored procedure to initiate a prohibited state change (see Section 3.1.1.4) that is not listed in the preceding table, then the stored procedure MUST NOT change the state of the topology activation action, and it MUST return corresponding error code.

The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_MSS_UpdateTopologyActivationAction (
        @ActionID             int,
        @NewState             smallint
```

```
);
```

**@ActionID:** The unique identifier of the topology activation action to be updated.

**@NewState:** The new value for the state of topology activation action. The value MUST be a Topology Activation Action State data type as specified in Section 2.2.1.8.

**Return Code Values:** An integer which MUST be one of the values listed in the following table:

| Value | Description |
|---|---|
| **0** | Successful execution. |
| **1** | There is no topology activation action with the specified unique identifier. |
| **2** | Disallowed state change: @NewState is set to NotStarted. |
| **3** | Disallowed state change: @NewState is set to InProgress, and the current state is not set to NotStarted. |
| **4** | Disallowed state change: @NewState is set to Finished, and the current state is not set to InProgress. |
| **5** | Disallowed state change: @NewState is set to Aborted, and the current state is not set to NotStarted or InProgress. |

**Result Sets:** MUST NOT return any result set.

### 3.1.6 Timer Events

None.

### 3.1.7 Other Local Events

None.

## 3.2 Client Details

### 3.2.1 Abstract Data Model

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

#### 3.2.1.1 Query Component Transitions

To accomplish the long-running operations of initialization or reinitialization with a full-text index catalog, query components(2) undergo transitions which, for the purpose of specifying this protocol, can be minimally specified using the following parameters:

- **Name**: a string which uniquely identifies the transition.

- **BeginState**: a Query Component State (section 2.2.1.3) that is the State value of the query component (2) (section 3.1.1.2) when the transition began.
- **EndState**: a Query Component State (section 2.2.1.3) that is the State value of the query component (2) (section 3.1.1.2) when the transition ends.
- **LastStep**: an integer that specifies the last TransitionStep value of the query component (2) (section 3.1.1.2) before the transition ends.
- **CopyCatalogStep**: the step at which an entire full-text index catalog is copied.
- **CopyRefactoredCatalogStep**: the step at which a refactored full-text index catalog is copied.

The transition parameters are specified in the following table.

| Name | BeginState (query component state, section 2.2.1.3) | EndState (query component state, section 2.2.1.3) | LastStep | CopyCatalogStep | CopyRefactoredCatalogStep |
|---|---|---|---|---|---|
| "initialize with empty catalog" | Uninitialized | Ready | 4 | -1 | -1 |
| "initialize from component" | Uninitialized | Ready | 9 | 7 | -1 |
| "initialize from repartitioning" | Uninitialized | Ready | 10 | -1 | 4 |
| "initialize from restore" | Uninitialized | Ready | 3 | -1 | -1 |
| "recover from component" | Offline | Ready | 9 | 7 | -1 |
| "delete" | Offline | Uninitialized | 4 | -1 | -1 |
| "split indexes transition" | Ready | IndexSplitDone | 0 | -1 | -1 |
| "revert " | IndexSplitDone | Ready | 0 | -1 | -1 |
| "refresh" | Offline | Ready | 0 | -1 | -1 |

The execution of a query component transition always begins at step 0 and proceeds incrementally until the appropriate value of LastStep for that transition. For the complete specification of the execution of a query component transition, see section 3.2.5.2.

#### 3.2.1.2 Server Name

The name of the server that corresponds to this application server.

#### 3.2.1.3 Current Query Component

The query component (2) on whose transition, if any, the application server is currently working. It has the following values:

- **QueryComponentID**: A GUID that uniquely identifies the query component (2).
- **State**: The Query Component State (section 2.2.1.3).
- **DesiredState**: The Query Component State (section 2.2.1.3) toward which a Query Component Transition (section 3.2.1.1) will progress.
- **TransitionSequenceName**: A string that specifies the Query Component Transition (section 3.2.1.1) that the query component (2) is currently executing.
- **TransitionStep**: The number of finished steps of the current Query Component Transition (section 3.2.1.1)
- **TransitionStatus**: The Query Component Transition Status (section 2.2.1.5) of the current Query Component Transition (section 3.2.1.1).
- **TransitionError**: A description of any error that occurred during the execution of the current Query Component Transition (section 3.2.1.1).
- **TransitionCancelRequested**: A Boolean value that specifies whether or not the current Query Component Transition (section 3.2.1.1) should be cancelled.
- **SourceComponentID**: The QueryComponentID (section 3.1.1.2) of the source query component (2). The source query component (2) is used to initialize the index on the given query component (2).
- **PauseRequested**: A Boolean value that specifies whether or not the query component (2) requires a pause of the search service application.

#### 3.2.1.4 Current Transition

The Query Component Transition (section 3.2.1.1) on which the application server is currently working. It has the following values:

- **Name**: A string that uniquely identifies the Query Component Transition (section 3.2.1.1).
- **BeginState**: A Query Component State (section 2.2.1.3) that is the State value of the query component (2) in the Query Topology (section 3.1.1.2) when the transition began.
- **EndState**: A Query Component State (section 2.2.1.3) that is the State value of the query component (2) in the Query Topology (section 3.1.1.2) when the transition ends.
- **LastStep**: An integer that specifies the last TransitionStep value of the query component (2) in the Query Topology (section 3.1.1.2) before the transition ends.
- **CopyCatalogStep**: The step at which an entire Full-Text Index Catalog ([MS-CIFO] section 2.18) is copied.

- **CopyRefactoredCatalogStep**: The step at which a Refactored Full-Text Index Catalog (section 2.2.3.1) is copied.

### 3.2.2 Timers

None.

### 3.2.3 Initialization

None.

### 3.2.4 Higher-Layer Triggered Events

None.

### 3.2.5 Message Processing Events and Sequencing Rules

#### 3.2.5.1 Administration Component Sequence

An application server MUST call the **proc_MSS_GetTopology** stored procedure on a periodic basis (for example, every minute). The particular time interval between executions does not alter the behavior of the protocol. When the application server receives a result set returned by that stored procedure it MUST perform the following actions depending on the values of AdminServerName and DesiredAdminServerName in that result set:

- If DesiredAdminServerName is set to the name of the server that called **proc_MSS_GetTopology** and AdminServerName is set to NULL, then that server MUST initialize administration component and call **proc_MSS_ReportAdminComponentState**. The administration component MUST be initialized using a local storage path and an administration component type (section 2.2.1.1) returned by **proc_MSS_GetTopology** in DesiredAdminLocalStoragePath and DesiredStandalone.
- If AdminServerName is set to the name of the server that called **proc_MSS_GetTopology** and DesiredAdminServerName is set to a different value, then the administration component MUST be uninitialized, and **proc_MSS_ReportAdminComponentState** stored procedure MUST be called with @AdminServerName = NULL, @AdminLocalStoragePath = NULL, @Standalone = 0 and @SettingsInRegistry = 1.

#### 3.2.5.2 Query Component Sequence

The query component transitions described in section 3.2.1.1 are executed by the application server by performing an action on the server and then incrementing the TransitionStep of the query component (section 3.2.1.3). Most of the specific actions correlating to each step of each transition are out of the scope of this document and thus, to follow the protocol correctly, an application server only needs to increment the TransitionStep value (section 3.2.1.3). Those actions for which a server-to-server message must be prepared, sent, or received are described in the following. These correspond to the CopyCatalogStep and CopyRefactoredCatalogStep values of the query component transitions (section 3.2.1.1).

All query component transitions described in section 3.2.1.1 MUST be executed by an application server by performing the following steps on a periodic basis. The particular time interval between executions does not alter the behavior of the protocol.

1. The application server MUST call the **proc_MSS_GetQueryComponents** stored procedure (section 3.1.5.51). The received query component result set (section 2.2.4.2) will be referred to in the following description.

2. For all query components(2) represented in the returned query component result set (section 2.2.4.2), where the value of ServerName (section 2.2.4.2) is equal to Server Name (section 3.2.1.2), steps 3 through 8 of this top-level list MUST be executed.

3. The Current Query Component (section 3.2.1.3) MUST be set so that:

    1. QueryComponentID (section 3.2.1.3) is set to the QueryComponentID value of that result (section 2.2.4.2).

    2. DesiredState (section 3.2.1.3) is set to the DesiredState value of that result (section 2.2.4.2).

    3. State (section 3.2.1.3) is set to the State value of that result (section 2.2.4.2).

    4. TransitionSequenceName (section 3.2.1.3) is set to the TransitionSequenceName value of that result (section 2.2.4.2).

    5. TransitionStep (section 3.2.1.3) is set to the TransitionStep value of that result (section 2.2.4.2).

    6. TransitionStatus (section 3.2.1.3) is set to the TransitionStatus value of that result (section 2.2.4.2).

    7. TransitionError (section 3.2.1.3) is set to the TransitionError value of that result (section 2.2.4.2).

    8. TransitionCancelRequested (section 3.2.1.3) is set to the TransitionCancelRequested value of that result (section 2.2.4.2).

    9. SourceComponentID (section 3.2.1.3) is set to the SourceComponentID value of that result (section 2.2.4.2).

    10. PauseRequested (section 3.2.1.3) is set to the PauseRequested value of that result (section 2.2.4.2).

4. The values of the current transition (section 3.2.1.4) MUST be set to the values of the query component transition (section 3.2.1.1) whose Name value is equal to the TransitionSequenceName value of the Current Query Component (section 3.2.1.3):

    1. Name (section 3.2.1.4) is set to the Name value of the Query Component Transition (section 3.2.1.1).

    2. BeginState (section 3.2.1.4) is set to the BeginState value of the Query Component Transition (section 3.2.1.1).

    3. EndState (section 3.2.1.4) is set to the EndState value of the Query Component Transition (section 3.2.1.1).

    4. LastStep (section 3.2.1.4) is set to the LastStep value of the Query Component Transition (section 3.2.1.1).

    5. CopyCatalogStep (section 3.2.1.4) is set to the CopyCatalogStep value of the Query Component Transition (section 3.2.1.1).

6. CopyRefactoredCatalogStep (section 3.2.1.4) is set to the CopyRefactoredCatalogStep value of the Query Component Transition (section 3.2.1.1).

5. To execute the query component transition sequence, the application server MUST perform the following steps:

    1. If the TransitionCancelRequested value of the Current Query Component (section 3.2.1.3) is true, the application server MUST call the **proc_MSS_SetQueryComponent** stored procedure (section 3.1.5.87) with @QueryComponentID set to the QueryComponentID value of the Current Query Component (section 3.2.1.3), @TransitionStatus set to *RollingBack* (section 2.2.1.5), and all other parameters set to **NUL**L. It MUST also set the TransitionStatus value of the Current Query Component (section 3.2.1.3) to *RollingBack* (section 2.2.1.5).

    2. If the TransitionStatus value of the current query component (section 3.2.1.3) is RollingBack (section 2.2.1.5), it MUST then skip the remaining steps listed at this level and continue at step 6 of the top-level list.

    3. If the TransitionStep value of the current query component (section 3.2.1.3) is greater than the LastStep value of the current transition (section 3.2.1.1), the application server MUST skip the remaining steps listed at this level and continue at step 7 of the top-level list.

    4. Repeat the following steps any number of times until step 4 is reached without error:

        1. If the TransitionStep value of the Current Query Component (section 3.2.1.3) is equal to the CopyCatalogStep value (section 3.2.1.1) of the current transition, two full-text index catalogs ([MS-CIFO] section 2.18) MUST be retrieved as specified in section 3.2.5.2.1, using the SourceComponentID value of the result. One full-text index catalog MUST be named "Portal_Content" and the other MUST be named "AnchorProject".

        2. If the TransitionStep value of the Current Query Component (section 3.2.1.3) is equal to the CopyRefactoredCatalogStep value (section 3.2.1.1) of the current transition, a Refactored Full-Text Index Catalog (section 2.2.3.1) MUST be received as specified in section 3.2.5.2.2, using the SourceComponentID value of the result. One Refactored Full-Text Index Catalog (section 2.2.3.1) MUST be named "Portal_Content" and the other MUST be named "AnchorProject".

        3. If the Name value of the Current Transition (section 3.2.1.4) is "split indexes", and the TransitionStep value of the Current Query Component (section 3.2.1.3) is **1**, then the Refactored Full-Text Index Catalogs (section 2.2.3.1) must be produced for both the Main Catalog ([MS-CIFO] section 2.18.1) and Anchor Text Catalog ([MS-CIFO] section 2.18.2), in the directories specified in section 3.2.5.2.2. The application server MUST create the same number of Refactored Full-Text Index catalogs (section 2.2.3.1) as there are index partitions in the new Query Topology (section 3.1.1.2). Each new Refactored Full-Text Index Catalog (section 2.2.3.1) is created using a partition number **0**, **1**, **2**, and so on, up to one less than the number of index partitions in the new query topology. The full-text index catalog data for an item MUST be copied from the existing full-text index catalog into the new Refactored Full-Text Index Catalog (section 2.2.3.1) if and only if its document identifier(1), modulo **256**, multiplied by the number of index partitions, and integer-divided (that is, leaving no fractional component) by **256**, is equal to the partition number of the Refactored Full-Text Index Catalog (section 2.2.3.1) being created. For an example see section 4.2.2.

        4. If any error occurs in step 1, 2, or 3, or any internal errors occur, the application server MUST:

1. Call the **proc_MSS_SetQueryComponent** stored procedure (section 3.1.5.87) with @QueryComponentID set to the QueryComponentID value of the Current Query Component (section 3.2.1.3), @TransitionError set to any message, and all other parameters set to **NULL, and**

2. Set the TransitionError value of the Current Query Component (section 3.2.1.3) to the message sent in step 3, setting the Current Query Component.

   At such time the application server MAY also choose to abort the query component transition sequence by:

   1. Calling the **proc_MSS_SetQueryComponent** stored procedure (section 3.1.5.87) with @QueryComponentID set to the QueryComponentID value of the Current Query Component (section 3.2.1.3) @TransitionStatus set to *RollingBack* (section 2.2.1.5), and all other parameters set to **NULL, and**

   2. Setting the @TransitionStatus value of the Current Query Component (section 3.2.1.3) to *RollingBack* (section 2.2.1.5).

5. If no error occurred in step 1, 2, or 3, the application server MUST call the **proc_MSS_SetQueryComponent** stored procedure (section 3.1.5.87) with @QueryComponentID set to the QueryComponentID value of the current query component (section 3.2.1.3), @TransitionStep set to the TransitionStep value of the Current Query Component (section 3.2.1.3) plus one, and all other parameters set to **NULL**. It MUST also set the TransitionStep value of the Current Query Component (section 3.2.1.3) to one higher than its current value.

6. Go back to step 1. of this list.

6. To roll back the query component transition sequence, the application server MUST perform the following steps.

   1. If the TransitionStep value of the Current Query Component (section 3.2.1.3) is less than **0**, the application server MUST skip the remaining steps listed at this level and continue at step 7 of the top-level list.

   2. The application server MUST call the **proc_MSS_SetQueryComponent** stored procedure (section 3.1.5.87) with @QueryComponentID set to the QueryComponentID value of the current query component (section 3.2.1.3), @TransitionStep set to the TransitionStep value of the current query component (section 3.2.1.3) minus one, and all other parameters set to NULL. It MUST also set the TransitionStep value of the current query component (section 3.2.1.3) to one lower than its current value.

   3. Go back to step 1 of this list.

7. At this step, the query component transition sequence is finished, and therefore the application server MUST perform the following steps:

   1. If the TransitionStatus value of the current query component (section 3.2.1.3) is *Executing* (section 2.2.1.5), it MUST call the **proc_MSS_SetQueryComponent** stored procedure (section 3.1.5.87) with @QueryComponentID set to the QueryComponentID value of the Current Query Component (section 3.2.1.3), @DesiredState set to the EndState value of the current transition (section 3.2.1.4), @State set to the EndState value of the current transition (section 3.2.1.4), @TransitionStep set to **-1**, @TransitionCancelRequested set to **0**, @PauseRequested set to **0**, @TransitionStatus set to *Completed* (section 2.2.1.5), and all other parameters set to **NULL**.

2. Otherwise, if the TransitionCancelRequested value of the Current Query Component (section 3.2.1.3) is true, it MUST call the **proc_MSS_SetQueryComponent** stored procedure (section 3.1.5.87) with @QueryComponentID set to the QueryComponentID value of the current query component (section 3.2.1.3), @DesiredState set to the BeginState value of the current transition (section 3.2.1.4), @State set to the BeginState value of the current transition (section 3.2.1.4), @TransitionStep set to **-1**, @TransitionCancelRequested set to **0**, @PauseRequested set to **0**, @TransitionStatus set to *Canceled* (section 2.2.1.5), and all other parameters set to **NULL**.

3. Otherwise, it MUST call the **proc_MSS_SetQueryComponent** stored procedure (section 3.1.5.87) with @QueryComponentID set to the QueryComponentID value of the Current Query Component (section 3.2.1.3), @DesiredState set to the BeginState value of the Current Transition (section 3.2.1.4), @State set to the BeginState value of the Current Transition (section 3.2.1.4), @TransitionStep set to **-1**, @TransitionCancelRequested set to **0**, @PauseRequested set to **0**, @TransitionStatus set to *Failed* (section 2.2.1.5), and all other parameters set to **NULL**.

8. If the TransitionSequenceName value of the current query component (section 3.2.1.3) is empty and the PauseRequested value of the current query component (section 3.2.1.3) is true, the application server MUST:

    1. Call the **proc_MSS_SetQueryComponent** stored procedure (section 3.1.5.87) with @QueryComponentID set to the QueryComponentID value of the Current Query Component (section 3.2.1.3), @PauseRequested set to **0**, and all other parameters set to NULL.

    2. Set the TransitionStep value of the Current Query Component (section 3.2.1.3) to false.

#### 3.2.5.2.1 Copying a Full-Text Index Catalog

To copy a Full-Text Index Catalog ([MS-CIFO] section 2.18) from a source query component (2), an application server MUST copy each one of the files that make up that Full-Text Index Catalog ([MS-CIFO] section 2.18). The files are located at a path relative to the shared directory in the file system of the source query component (2). The directory path containing these files MUST be generated as follows:

```
<directory path>=\\<server name>\<share name>\<query component guid>-query-<query
component number>\Projects\<catalog name>\Indexer\CiFiles
```

Where:

- `<server name>` is the ServerName value (section 3.1.1.2) of the query component
- `<share name>` is the ShareName value (section 3.1.1.2) of the query component
- `<query component guid>` is the QueryComponentID value (section 3.1.1.2) of the query component
- `<query component number>` is the QueryComponentNumber value (section 3.1.1.2) of the query component
- `<catalog name>` is the name of the catalog to be copied. The valid names are "Portal_Content" and "AnchorProject"

#### 3.2.5.2.2 Copying a Refactored Full-Text Index Catalog

To copy a Refactored Full-Text Index Catalog (section 2.2.3.1) from a source query component (2), an application server MUST copy each one of the files that make up that Refactored Full-Text Index Catalog (section 2.2.3.1). The files are located at a path relative to the shared directory in the file system of the source query component (2). The directory path containing these files MUST be generated as follows:

```
<directory path>=\\<server name>\<share name>\<query component guid>-query-<query
component number>\Projects\<catalog name>\Indexer\CiFiles
```

Where:

- `<server name>` is the ServerName value (section 3.1.1.2) of the query component (2)
- `<share name>` is the ShareName value (section 3.1.1.2) of the query component (2)
- `<query component guid>` is the QueryComponentID value (section 3.1.1.2) of the query component (2)
- `<query component number>` is the QueryComponentNumber value (section 3.1.1.2) of the query component (2)
- `<catalog name>` is the name of the catalog to be copied

Refer to section 4.2.2 for an example.

### 3.2.5.3 Crawl Component Sequence

An application server MUST call the **proc_MSS_GetCrawlComponents** stored procedure (section 3.1.5.26) on a periodic basis (for example, every minute). The particular time interval between executions does not alter the behavior of the protocol. When the application server receives the crawl components result set (section 2.2.4.1), it MUST perform the following actions for each result, depending on the values of ServerName, State and DesiredState:

- If the ServerName value of the result (section 2.2.4.1) is the name of the server that called **proc_MSS_GetCrawlComponents**, State is Uninitialized (section 2.2.1.7), and DesiredState is Ready (section 2.2.1.7), then the application server MUST call the **proc_MSS_ReportCrawlComponentState** procedure (section 3.1.5.75) with @CrawlComponentID set to the value of CrawlComponentID of the result, and @State set to Ready (section 2.2.1.7).
- If the ServerName value of the result (section 2.2.4.1) is the name of the server that called **proc_MSS_GetCrawlComponents**, and DesiredState is Uninitialized (section 2.2.1.7), then the application then the server MUST call the **proc_MSS_ReportCrawlComponentState** procedure (section 3.1.5.75) with @CrawlComponentID set to the value of CrawlComponentID in the result, and @State set to Uninitialized (section 2.2.1.7).

### 3.2.5.4 Database Refactoring Sequence

An application server MUST call **proc_MSS_GetActiveRefactoringTaskBatches** stored procedure on periodic basis (for example, every minute) when a query topology or a crawl topology is being activated. The particular time interval between executions does not alter the behavior of the protocol. The @ServerName parameter of this stored procedure MUST be set to the name of the application server that calls that stored procedure. After the application servers receives results set that contains list of refactoring task batches, the server MUST execute each of these refactoring task

batches and report their status using **proc_MSS_ReportRefactoringTaskBatch** stored procedure. How each refactoring task batch is executed is determined by the type of the corresponding refactoring task as described in the following.

For refactoring tasks of type "**PropertyStoreCopy**" information that is stored in the following tables is copied from the source metadata index to the destination metadata index:

- **MSSDocSdids**
- **MSSDefinitions**
- **MSSDuplicateHashes**
- **MSSDocResults**
- **MSSDocProps**

These tables are documented in [MS-SQLPQ2].The source and destination metadata indexes are defined by **SourceComponentID** and **DestinationComponentID** parameters of the refactoring task. Rows that correspond to the documents that satisfy both of the following two conditions MUST be copied:

- Document identifiers(1) is in the range defined by **StartDocID** and **EndDocID** parameters of the refactoring task batch,
- Document distribution identifier is in the set defined by the **Refactoring Task Part Result Set** returned from **proc_MSS_GetRefactoringTask**.

For refactoring tasks of type "**PropertyStoreDelete**" information that is stored in the following tables is deleted:

- **MSSDocSdids**
- **MSSDefinitions**
- **MSSDuplicateHashes**
- **MSSDocResults**
- **MSSDocProps**

These tables are documented in [MS-SQLPQ2]. The metadata index from which information MUST be deleted is defined by **SourceComponentID** parameter of the refactoring task. Rows that correspond to the documents that satisfy both of the following two conditions MUST be deleted:

- Document identifiers(1) is in the range defined by **StartDocID** and **EndDocID** parameters of the refactoring task batch
- Document distribution identifier is in the set defined by the **Refactoring Task Part Result Set** returned from **proc_MSS_GetRefactoringTask**

For refactoring tasks of type "**CrawlStoreMove**":

1. When **StartDocID** and **EndDocID** parameters of the refactoring task batch are not set to **-1**, information that is stored in the following tables is moved from the source crawl store to the destination crawl store:
   - MSSAnchorChangeLog

- MSSAnchorPendingChangeLog
- MSSAnchorText
- MSSAnnotations
- MSSCrawlChangedCommittedDocs
- MSSCrawlChangedDeletedDocs
- MSSCrawlChangedSourceDocs
- MSSCrawlChangedTargetDocs
- MSSCrawlDeletedURL
- MSSCrawlLinksLog
- MSSCrawlURLLog
- MSSCrawlQueue
- MSSTranTempTable0
- MSSCrawlURLReport
- MSSCrawlURL

  The source and destination crawl stores are defined by **SourceComponentID** and **DestinationComponentID** parameters of the refactoring task. Rows that correspond to the documents that satisfy both of the following two conditions MUST be moved:

- Document identifiers(1) is in the range defined by StartDocID and EndDocID parameters of the refactoring task batch
- Identifier of the host name for the document is in the set defined by the **Refactoring Task Part Result Set** returned from **proc_MSS_GetRefactoringTask**

2. When **StartDocID** and **EndDocID** parameters of the refactoring task batch are set to **-1**, information that is stored in the following tables is moved from the source crawl store to the destination crawl store:

   - MSSCrawlHostList
   - MSSCrawlHostsLog
   - MSSUserHosts

The source and destination crawl stores are defined by **SourceComponentID** and **DestinationComponentID** parameters of the refactoring task. In these tables only the rows that correspond to the host names that are in the set defined by **Refactoring Task Part Result Set** returned from **proc_MSS_GetRefactoringTask** must be moved.

If an error is encountered during execution of a refactoring task batch, that error MUST be reported using **proc_MSS_ReportRefactoringTaskBatchError** stored procedure.

### 3.2.6 Timer Events

None.

### 3.2.7 Other Local Events

None.

# 4 Protocol Examples

This section provides specific example scenarios for search topology administration tasks. In all of these examples the administration server is an application server that controls the execution of the sequence. Any application server in the farm can play a role of the administration server.

## 4.1 Administration Component Initialization

The following diagram shows sequence of actions that is executed when the administration component is initialized. The name of the application server shown on this diagram is 'server0'. The administration component is initialized on that server.

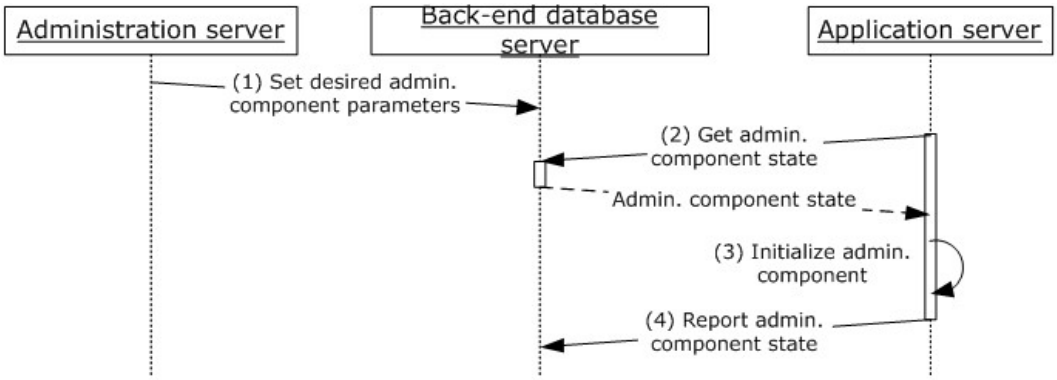


**Figure 7: Administration Component Initialization Sequence**

The sequence contains following actions:

1. The administration server calls the **proc_MSS_UpdateTopology** stored procedure with the following data

   - @DesiredAdminServerName = **'server0'**
   - @DesiredAdminLocalStoragePath = **'C:\Index'**
   - @DesiredStandalone = **0**
   - @SettingsInRegistry = **NULL**

   The stored procedure returns **0**.

2. The application server calls the **proc_MSS_GetTopology** stored procedure, and receives a result set that contains one row with the following data:

   - TopologyID = **0**
   - DesiredAdminServerName = **'server0'**
   - DesiredAdminLocalStoragePath = **'C:\Index'**
   - DesiredStandalone = **0**
   - AdminServerName = **NULL**
   - AdminLocalStoragePath = **NULL**

- Standalone = **NULL**
- SettingsInRegistry = **0**

3. The administration component is initialized on the given application server (see Section 3.2.5.1) because the received result set contained DesiredAdminServerName = **'server0'** and AdminServerName = **NULL**
4. The application server calls the **proc_MSS_ReportAdminComponentState** stored procedure with the following:
   - @AdminServerName = **'Server0'**
   - @AdminLocalStoragePath = **'C:\Index'**
   - @Standalone = **0**
   - @SettingsInRegistry = **1**

     The stored procedure returns **0**.

## 4.2 Query Topology Activation

This example shows the process of creation and activation of a new query topology.

**Initial state:**

- One query topology with the following:
  - QueryTopologyID = **'2D4EB671-D3E0-4233-95BB-9490058A260E'**
  - State = **Active**
- One metadata index with the following:
  - MetadataIndexID = **'1C35208B-6F92-4af4-AA5C-74D714A66D17'**
  - Name = **"PropertyStore1"**
- One index partition for the active query topology with the following:
  - PartitionID = **'0FB5791F-0255-4426-90DE-B338F208B3CF'**
  - QueryTopologyID = **'2D4EB671-D3E0-4233-95BB-9490058A260E'**
  - MetadataIndexID = **'1C35208B-6F92-4af4-AA5C-74D714A66D17'**
  - Ordinal = **0**
- One query component (2) associated with the active query topology with the following:
  - QueryComponentNumber = **0**
  - QueryComponentID = **'AEC819CB-4CE8-4382-9E58-2CAD65ACDA99'**
  - ServerName = **'Server0'**
  - LocalStoragePath = **'C:\Index'**

- PartitionID = **'0FB5791F-0255-4426-90DE-B338F208B3CF'**
- State = **Ready**

The following diagram shows a sequence of actions that is executed when a new query topology is created and activated. In this example a query topology with two index partitions is created, each index partition has one query component (2) assigned to it. This sequence can be invoked from the **topology** administration UI.

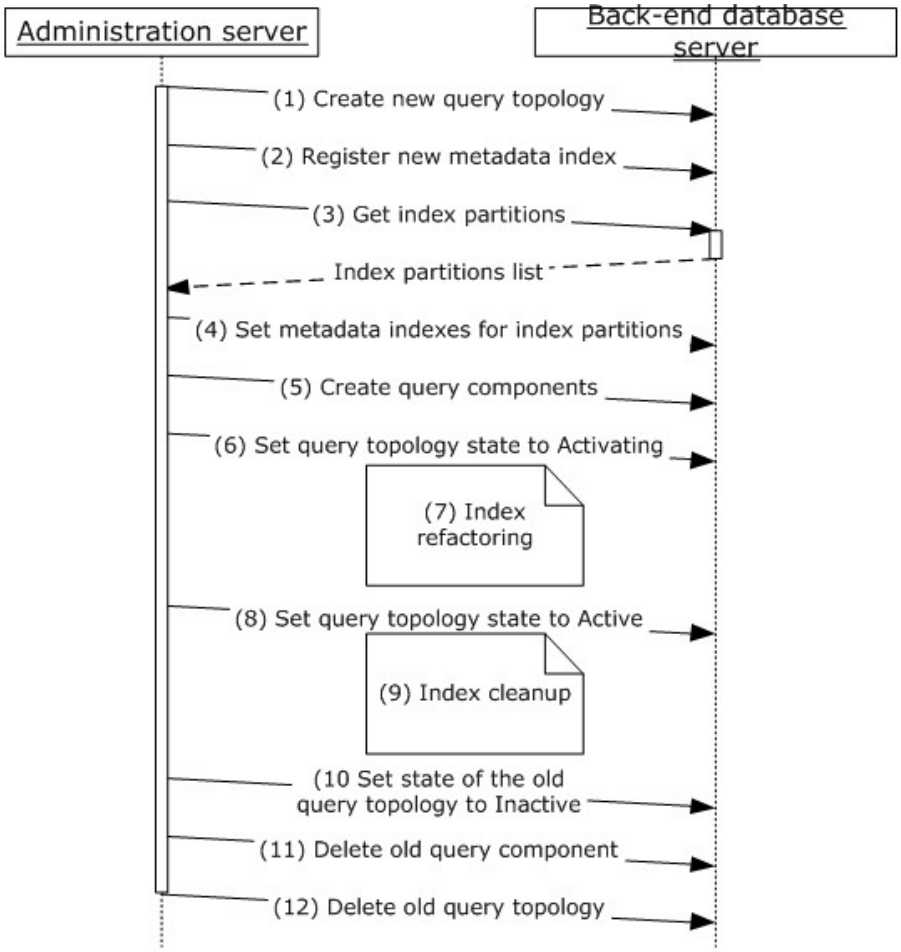


**Figure 8: Query Topology Activation Sequence**

The sequence consists of the following steps:

- Administration server calls the **proc_MSS_CreatePartitionScheme** stored procedure to create a new query topology. @PartitionsNumber is set to **2**. The store procedure sets:
  - @PartitionSchemeID to **'F51D68EC-EAA9-4525-B709-D501B9148482'**

and returns **0**.

1. To register a new metadata index the **proc_MSS_RegisterPropertyStore** store procedure is called with:

- @Name = **'PropertyStore2'**
- @PropertyStoreID = **'260C2399-5ADB-43A8-BF54-6BBA0237AA24'**

2. The **proc_MSS_GetPartitions** stored procedure is called with:
   - @PartitionSchemeID = 'F51D68EC-EAA9-4525-B709-D501B9148482' to get the list of index partitions in the newly create query topology
   - The back-end server returns result set with two rows:
     1. Row 1:
        1. PartitionSchemeID = 'F51D68EC-EAA9-4525-B709-D501B9148482'
        2. PartitionID = '0DE05E5E-D9A3-495d-9ACE-A15AE9664036'
        3. Ordinal = **0**
        4. PropertyStoreID = **NULL**
     2. Row 2
        1. PartitionSchemeID = 'F51D68EC-EAA9-4525-B709-D501B9148482'
        2. PartitionID = 'DF51A997-FB76-4378-9E2D-8B3101C9FA29'
        3. Ordinal = **1**
        4. PropertyStoreID = **NULL**
3. To assign a metadata index to the index partition the **proc_MSS_SetPartitionPropertyStore** stored procedure is called for each of these index partitions. It is called for the first index partition with:
   - @PartitionSchemeID = 'F51D68EC-EAA9-4525-B709-D501B9148482'
   - @PartitionID = '0DE05E5E-D9A3-495d-9ACE-A15AE9664036'
   - @PropertyStoreID = '1C35208B-6F92-4af4-AA5C-74D714A66D17'

   For the second index partition the stored procedure is called with the following parameters:
   - @PartitionSchemeID = **'F51D68EC-EAA9-4525-B709-D501B9148482'**
   - @PartitionID = **'DF51A997-FB76-4378-9E2D-8B3101C9FA29'**
   - @PropertyStoreID = **'260C2399-5ADB-43A8-BF54-6BBA0237AA24'**
4. Two new query components(2) are created using the **proc_MSS_CreateQueryComponent** stored procedure. For the first query component (2) the stored procedure is called with:
   - @ServerName = **'server1'**
   - @LocalStoragePath = **'C:\Index'**
   - @PartitionSchemeID = **'F51D68EC-EAA9-4525-B709-D501B9148482'**
   - @PartitionID = **'0DE05E5E-D9A3-495d-9ACE-A15AE9664036'**

- @DesiredState = **0**
- @HotSwap = **0**
- @ShareName = **NULL**
- @UsesCustomShare = **0**

  The stored procedure sets:

- @QueryComponentID to **'0AD33931-9B1B-4c29-885E-A1E951DA8B59'**
- @QueryComponentNumber to **1**

  For the second query component (2) the stored procedure is called with:

- @ServerName = **'server2'**
- @LocalStoragePath = **'C:\Index'**
- @PartitionSchemeID **= 'F51D68EC-EAA9-4525-B709-D501B9148482'**
- @PartitionID = **'DF51A997-FB76-4378-9E2D-8B3101C9FA29'**
- @DesiredState = **0**
- @HotSwap = **0**
- @ShareName = **NULL**
- @UsesCustomShare = **0**

  The stored procedure sets:

- @QueryComponentID to '**B1699847-F435-4541-8D66-968813731961'**
- @QueryComponentNumber to **2**

5. The **proc_MSS_SetPartitionSchemeState** stored procedure is called with:

   - @PartitionSchemeID = **'F51D68EC-EAA9-4525-B709-D501B9148482'**
   - @NewState = **Activating**
   - @Force = **0**

     The stored procedure changes state of the query topology to **Activating** and returns **0**.

6. On this step the full-text index catalogs and metadata indexes are being repartitioned. An example of metadata index repartitioning is described in section 4.2.1. Refactoring of the full-text index catalogs is discussed in section 4.2.2.

7. After index refactoring has been finished, the **proc_MSS_SetPartitionSchemeState** stored procedure is called with:

   - @PartitionSchemeID = **'F51D68EC-EAA9-4525-B709-D501B9148482'**
   - @NewState = **Active**
   - @Force = **0**

The stored procedure changes the state of the query topology to **Active** and returns **0**. This stored procedure also changes the state of the previously active query topology to **Deactivating**.

8. On this step old query components(2) are deactivated and metadata indexes are cleaned of the information that was used by the old query topology. The process of removing data from the old metadata indexes is similar to the sequence described in section 4.2.1 (the main difference is that TaskType for the refactoring task is set to **'PropertyStoreDelete'** instead of **'PropertyStoreCopy'**).

9. The **proc_MSS_SetPartitionSchemeState** stored procedure is called with:

   - @PartitionSchemeID = **'2D4EB671-D3E0-4233-95BB-9490058A260E'**
   - @NewState = **Inactive**
   - @Force = **0**

   The stored procedure changes the state of the old query topology to **Inactive** and returns **0**.

10. To delete old query component (2) the **proc_MSS_DeleteQueryComponent** stored procedure is called with:

    - @PartitionSchemeID = **'2D4EB671-D3E0-4233-95BB-9490058A260E'**
    - @QueryComponentID **= 'AEC819CB-4CE8-4382-9E58-2CAD65ACDA99'**

11. The **proc_MSS_DeletePartitionScheme** stored procedure is called with:

    - @PartitionSchemeID = **'2D4EB671-D3E0-4233-95BB-9490058A260E'**

to delete the old query topology.

### 4.2.1 Metadata Index Refactoring

Metadata index refactoring happens as a part of query topology activation. During this process data is copied from metadata indexes that are used by the old query topology to the metadata indexes used by the new query topology.

**The following diagram** shows an example of the metadata index refactoring process. For simplicity in this example all refactoring task batches are executed on the application server with name 'server1'. The refactoring task batches can be distributed among multiple application servers depending on the implementation.

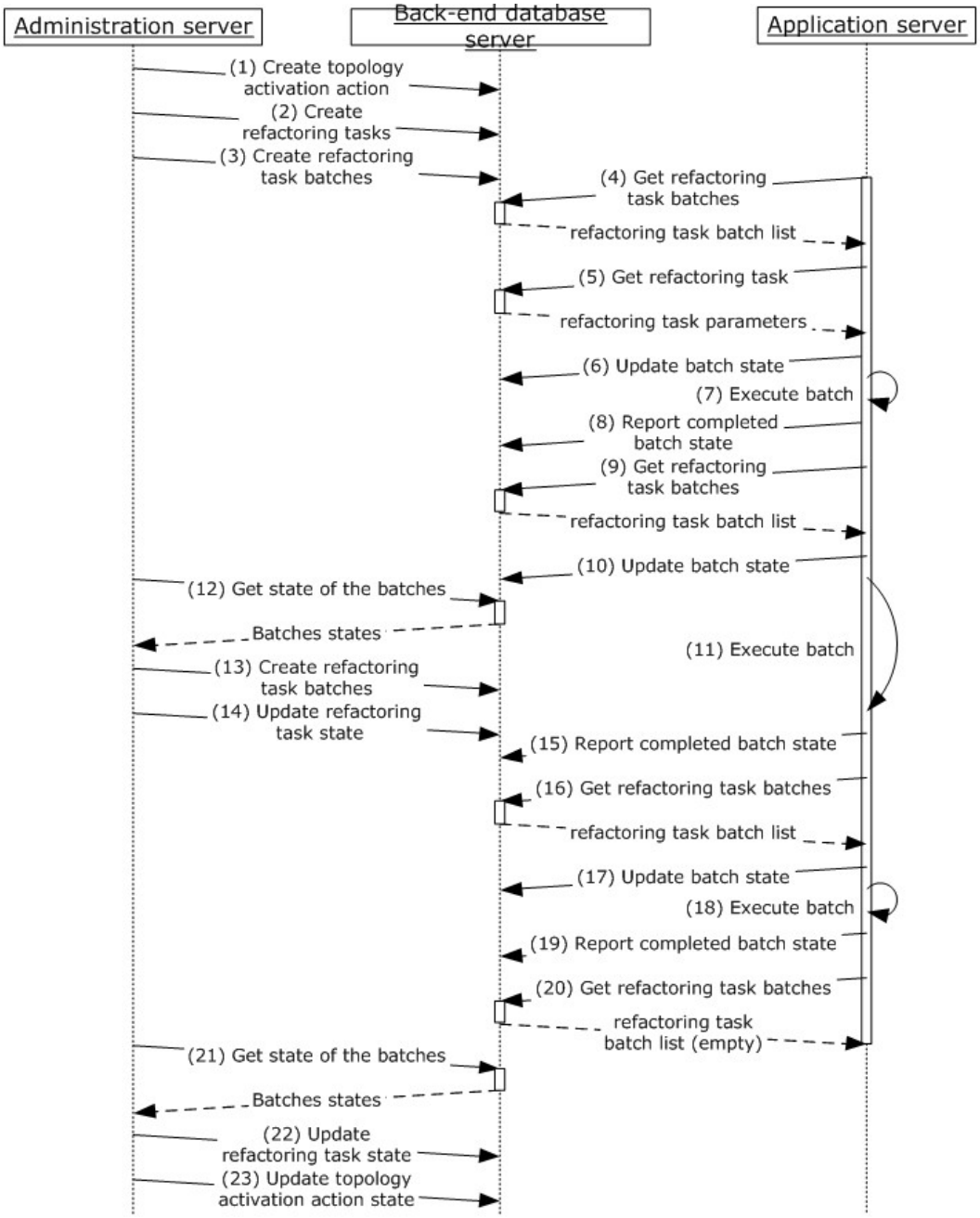


**Figure 9: Metadata Index Refactoring Sequence**

1. A new topology activation action is created by calling the **proc_MSS_CreateTopologyActivationAction** stored procedure with

   - @Name = **'PropertyStoreRefactoring'**
   - @TopologyID = **'F51D68EC-EAA9-4525-B709-D501B9148482'**

The stored procedure sets @ActionID to 1 and returns 0. After that stored procedure **proc_MSS_UpdateTopologyActivationAction** is called with

- @ActionID = **1**
- @NewState = **1** (InProgress)

2. Administration server calls **proc_MSS_CreateRefactoringTask** with
   - @ActionID = **1**
   - @TaskType = **"PropertyStoreCopy"**
   - @SourceComponentID = **'1C35208B-6F92-4af4-AA5C-74D714A66D17'**
   - @DestinationComponentID = **'260C2399-5ADB-43A8-BF54-6BBA0237AA24'**
   - @StartDocID = **0**
   - @EndDocID = **768**
   - @PartsXml is set to the following XML blob:

```
<root>
<part>128</part>
<part>129</part>
<part>130</part>
.........
<part>253</part>
<part>254</part>
<part>255</part>
</root>
```

   The stored procedure sets @TaskID to **1** and returns **0**.

3. Stored procedure **proc_MSS_CreateRefactoringTaskBatch** is called twice to create two refactoring task batches. For the first batch:
   - @TaskID = **1**
   - @StartDocID = **0**
   - @EndDocID = **256**
   - @NumOfDocs = **-1**
   - @ServerName = **'server0'**

   The stored procedure sets @BatchID = **0**. For the second batch the stored procedure is called with:
   - @TaskID = **1**
   - @StartDocID = **256**
   - @EndDocID = **512**
   - @NumOfDocs = **-1**

- @ServerName = **'server0'**

  The stored procedure sets @BatchID = **1**.

4. The application server calls **proc_MSS_GetActiveRefactoringTaskBatches** with @ServerName = 'server0', @BatchesCount = 10, @MaxErrorCount = 0. The stored procedure returns result set that contains two rows with the following values:

   - First row:
     - BatchID = **0**
     - TaskID = **1**
     - StartDocID = **0**
     - EndDocID = **256**
     - ServerName = **'server0'**
     - AssignedTime – time when the batch was created
     - State = **0** (NotStarted)
     - StartedTime = **NULL**
     - FinishedTime = **NULL**
     - LastErrorDescription = **NULL**
     - LastErrorTime = **NULL**
     - ErrorCount = **0**
     - NumOfDocs = **-1**
   - Second row:
     - BatchID = **1**
     - TaskID = **1**
     - StartDocID = **256**
     - EndDocID = **512**
     - ServerName = **'server0'**
     - AssignedTime – time when the batch was created
     - State = **0** (NotStarted)
     - StartedTime = **NULL**
     - FinishedTime = **NULL**
     - LastErrorDescription = **NULL**
     - LastErrorTime = **NULL**

- ErrorCount = **0**
- NumOfDocs = **-1**

5. Stored procedure **proc_MSS_GetRefactoringTask** is called with @TaskID = **1** to receive information about refactoring task. The stored procedure returns two result sets the first result set contains one row with:
   - TaskID = **1**
   - ActionID = **1**
   - TaskAssignedTime – time when the refactoring task was created
   - SourceComponentID = **'1C35208B-6F92-4af4-AA5C-74D714A66D17'**
   - DestinationComponentID = **'260C2399-5ADB-43A8-BF54-6BBA0237AA24'**
   - TaskType = **'PropertyStoreCopy'**
   - CurrentDocID = **0**
   - EndDocID = **768**
   - SuccessfullyCopied = **0**
   - TotalToCopy = **0**
   - TaskState = **0** (NotStarted)
   - ErrorDescription = **NULL**

   The second result set returned by the stored procedure contains 128 rows; each row contains one column Part with values 128, 129, 130, …, 253, 254, 255.
6. Stored procedure **proc_MSS_ReportRefactoringTaskBatch** is called with:
   - @BatchID = **0**
   - @NewState = **1** (InProgress)

   The stored procedure returns 0.
7. The application server executes refactoring task batch by copying items with document identifiers in range [128, 255] from the first metadata index to the second.
8. Stored procedure **proc_MSS_ReportRefactoringTaskBatch** is called with:
   - @BatchID = **0**
   - @NewState = **2** (Finished)

   The stored procedure returns 0.
9. The application server calls **proc_MSS_GetActiveRefactoringTaskBatches** with:
   - @ServerName = **'server0'**
   - @BatchesCount = **10**

- @MaxErrorCount = **0**

  The stored procedure returns result set that contains one row with the following values:

- BatchID = **1**
- TaskID = **1**
- StartDocID = **256**
- EndDocID = **512**
- ServerName = **'server0'**
- AssignedTime – time when the batch was created
- State = **0** (NotStarted)
- StartedTime = **NULL**
- FinishedTime = **NULL**
- LastErrorDescription = **NULL**
- LastErrorTime = **NULL**
- ErrorCount = **0**
- NumOfDocs = **-1**

10. Stored procedure **proc_MSS_ReportRefactoringTaskBatch** is called with:

    - @BatchID = **1**
    - @NewState = **1** (InProgress)

      The stored procedure returns **0**.

11. The application server executes refactoring task batch by copying items with document identifiers in range [384, 511] from the first metadata index to the second.

12. The administration server calls **proc_MSS_GetRefactoringTaskBatches** with:

    - @TaskID = **1**
    - @StartDocID = **0**

      The stored procedure returns result set that contains two rows with the following values:

    - First row:
      - BatchID = **0**
      - TaskID = **1**
      - StartDocID = **0**
      - EndDocID = **256**
      - ServerName = **'server0'**

- AssignedTime – time when the batch was created
- State = **2** (Finished)
- StartedTime – time when the batch was started
- FinishedTime = time when the batch was finished
- LastErrorDescription = **NULL**
- LastErrorTime = **NULL**
- ErrorCount = **0**
- NumOfDocs = **-1**

- Second row:
  - BatchID = **1**
  - TaskID = **1**
  - StartDocID = **256**
  - EndDocID = **512**
  - ServerName = **'server0'**
  - AssignedTime – time when the batch was created
  - State = **1** (InProgress)
  - StartedTime – time when the batch was started
  - FinishedTime = **NULL**
  - LastErrorDescription = **NULL**
  - LastErrorTime = **NULL**
  - ErrorCount = **0**
  - NumOfDocs = **-1**

13. Stored procedure **proc_MSS_CreateRefactoringTaskBatch** is called to create next refactoring task batches. The stored procedure is called with the following parameters:
    - @TaskID = **1**
    - @StartDocID = **512**
    - @EndDocID = **768**
    - @NumOfDocs = **-1**
    - @ServerName = **'server0'**

      The stored procedure sets @BatchID to 2.

14. State of the refactoring task is stored using **proc_MSS_ReportRefactoringTask**. That stored procedure is called with:

    - @TaskID = **1**
    - @CurrentDocID = **256**
    - @EndDocID = **768**
    - @SuccessfullyCopied = **128**
    - @TotalToCopy = **384**
    - @TaskState = **1** (InProgress)

    The stored procedure updates state of the refactoring task and return 0.

15. Stored procedure **proc_MSS_ReportRefactoringTaskBatch** is called with:

    - @BatchID = **1**
    - @NewState = **2** (Finished)

    The stored procedure returns **0**.

16. The application server calls **proc_MSS_GetActiveRefactoringTaskBatches** with @ServerName = 'server0', @BatchesCount = 10, @MaxErrorCount = 0. The stored procedure returns result set that contains one row with the following values:

    - BatchID = **2**
    - TaskID = **1**
    - StartDocID = **512**
    - EndDocID = **768**
    - ServerName = **'server0'**
    - AssignedTime – time when the batch was created
    - State = **0** (NotStarted)
    - StartedTime = **NULL**
    - FinishedTime = **NULL**
    - LastErrorDescription = **NULL**
    - LastErrorTime = **NULL**
    - ErrorCount = **0**
    - NumOfDocs = **-1**

17. Stored procedure **proc_MSS_ReportRefactoringTaskBatch** is called with:

    - @BatchID = **2**
    - @NewState = **1** (InProgress)

The stored procedure returns **0**.

18. The application server executes refactoring task batch by coping items with document identifiers in range [640, 767] from the first metadata index to the second (see Section 3.2.5.4).

19. Stored procedure **proc_MSS_ReportRefactoringTaskBatch** is called with:

- @BatchID = **2**
- @NewState = **2** (Finished)

  The stored procedure returns 0.

20. The application server calls **proc_MSS_GetActiveRefactoringTaskBatches** with:

- @ServerName = **'server0'**
- @BatchesCount = **10**
- @MaxErrorCount = **0**

  The stored procedure returns empty result set.

21. The administration server calls **proc_MSS_GetRefactoringTaskBatches** with:

- @TaskID = **1**
- @StartDocID = **256**

  The stored procedure returns result set that contains two rows with the following values:

- First row:
  - BatchID = **1**
  - TaskID = **1**
  - StartDocID = **256**
  - EndDocID = **512**
  - ServerName = **'server0'**
  - AssignedTime – time when the batch was created
  - State = **2** (Finished)
  - StartedTime – time when the batch was started
  - FinishedTime – time when the batch was finished
  - LastErrorDescription = **NULL**
  - LastErrorTime = **NULL**
  - ErrorCount = **0**
  - NumOfDocs = **-1**
- Second row:

- BatchID = **2**
- TaskID = **1**
- StartDocID = **512**
- EndDocID = **768**
- ServerName = **'server0'**
- AssignedTime – time when the batch was created
- State = **2** (Finished)
- StartedTime – time when the batch was started
- FinishedTime – time when the batch was finished
- LastErrorDescription = **NULL**
- LastErrorTime = **NULL**
- ErrorCount = **0**
- NumOfDocs = **-1**

22. State of the refactoring task is stored using **proc_MSS_ReportRefactoringTask**. That stored procedure is called with:
    - @TaskID = **1**
    - @CurrentDocID = **768**
    - @EndDocID = **768**
    - @SuccessfullyCopied = **384**
    - @TotalToCopy = **384**
    - @TaskState = **2** (Finished)

    The stored procedure updates state of the refactoring task and return **0**.

23. Stored procedure **proc_MSS_UpdateTopologyActivationAction** is called with:
    - @ActionID = **1**
    - @NewState = **2** (Finished)

    The stored procedure updates state of the topology activation action and returns **0**.

### 4.2.2 Full-Text Index Refactoring

Full-text index refactoring happens as part of query topology activation. During this process, the query component (2) in the active query topology copies a full-text index catalog to each of the query components (2) in the new, activating query topology. The activating query components (2) are then initialized to bring them to the Ready state.

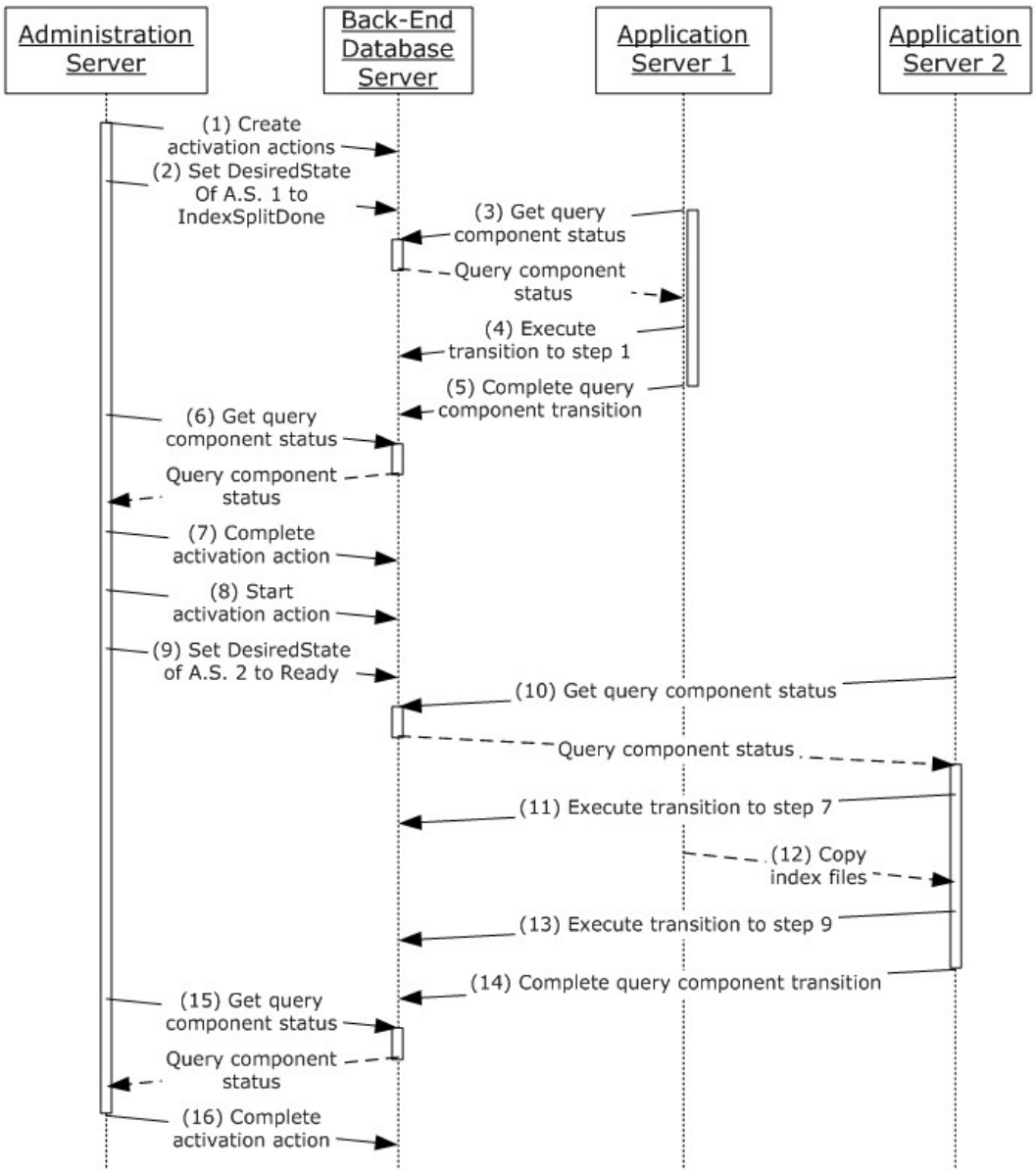


**Figure 10: Full-Text Index Refactoring Sequence**

1. Administration Server creates topology activation actions "IndexSplitting" and "InitializeAfterRepartition" and starts the "IndexSplitting" action.

   - Calls **proc_MSS_CreateTopologyActivationAction** stored procedure with
     - @Name = **"IndexSplitting"**
     - @TopologyID = **{F51D68EC-EAA9-4525-B709-D501B9148482}**
     - Back-End Database Server creates the topology activation action and returns **0**. After the call completes, @ActionID = **2**.

- Calls **proc_MSS_CreateTopologyActivationAction** stored procedure with
  - @Name = **"InitializeAfterRepartitioning"**
  - @TopologyID = **{F51D68EC-EAA9-4525-B709-D501B9148482}**
  - Back-End Database Server creates the topology activation action and returns **0**. After the call completes, @ActionID = **3**.
- Calls **proc_MSS_UpdateTopologyActivationAction** stored procedure with
  - @ActionID = **2**
  - @NewState = **1** (InProgress)
  - Back-End Database Server updates the topology activation action's State value to **InProgress** and returns **0**.

2. Administration Server starts the "split indexes" transition on the query component (2) that will act as a source for the full-text index catalog data for the newly created query components (2).
   - Calls **proc_MSS_GetQueryComponents**, which returns a result set containing one result, with the following values:
     - QueryComponentNumber = **0**
     - QueryComponentID = **{AEC819CB-4CE8-4382-9E58-2CAD65ACDA99}**
     - ServerName = **"Server0"**
     - LocalStoragePath = **"C:\Index"**
     - PartitionID = **{0FB5791F-0255-4426-90DE-B338F208B3CF}**
     - DesiredState = **1** (Ready)
     - DesiredStateSetTime = **August 1, 2009 1:21:34 AM**
     - HotSwap = **0**
     - ShareName = **"aec819cb-4ce8-4382-9e58-2cad65acda99-query-0"**
     - UsesCustomShare = **0**
     - State = **1** (Ready)
     - LastPropagationTime = **August 2, 2009 3:35:01 PM**
     - TransitionSequenceName = **"initialize first"**
     - TransitionStep = **-1**
     - TransitionStepStartTime = **August 1, 2009 2:11:47 AM**
     - TransitionStatus = **1** (Complete)
     - TransitionError = **""**
     - TransitionCancelRequested = **0**

- SourceComponentID = **NULL**
- SourceComponentPath = **NULL**
- PauseRequested = **0**
- SettingsInRegistry = **0**
- ScopeCompilationID = **7**

- Chooses a query component from the result set that is in the Ready state, of which there is only one: the active query component whose QueryComponentID is **{AEC819CB-4CE8-4382-9E58-2CAD65ACDA99}**.
- Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except
  - @QueryComponentID = **{AEC819CB-4CE8-4382-9E58-2CAD65ACDA99}**
  - @DesiredState = **103** (IndexSplitDone)
  - @TransitionSequenceName = **"split indexes"**
  - @TransitionStep = **0**
  - @TransitionStatus = **0** (Executing)
  - @TransitionError = **""**
  - @TransitionCancelRequested = **0**
  - Back-End Database Server sets the query component's DesiredState value to **IndexSplitDone**, TransitionStep value to **0**, TransitionStatus value to **Executing**, TransitionError value to **""**, and TransitionCancelRequested value to **false**, and finally returns **0**.

3. Application Server 1 polls the Back-End Database Server until it finds that one of the query components (2) should be performing a transition.
   - Calls **proc_MSS_GetQueryComponents**, which returns a result set containing three results, one of which has the following values representing the active query component (2):
     - QueryComponentNumber = **0**
     - QueryComponentID = **{AEC819CB-4CE8-4382-9E58-2CAD65ACDA99}**
     - ServerName = **"server0"**
     - LocalStoragePath = **"C:\Index"**
     - PartitionID = **{0FB5791F-0255-4426-90DE-B338F208B3CF}**
     - DesiredState = **103** (IndexSplitDone)
     - DesiredStateSetTime = **August 1, 2009 1:21:34 AM**
     - HotSwap = **0**
     - ShareName = **"aec819cb-4ce8-4382-9e58-2cad65acda99-query-0"**

- UsesCustomShare = **0**
- State = **1** (Ready)
- LastPropagationTime = **August 2, 2009 3:35:01 PM**
- TransitionSequenceName = **"split indexes"**
- TransitionStep = **0**
- TransitionStepStartTime = **August 1, 2009 2:11:47 AM**
- TransitionStatus = **0** (Executing)
- TransitionError = **""**
- TransitionCancelRequested = **0**
- SourceComponentID = **NULL**
- SourceComponentPath = **NULL**
- PauseRequested = **0**
- SettingsInRegistry = **0**
- ScopeCompilationID = **7**

4. Application Server 1 executes the "split indexes" query component transition sequence of the active query component (2), increasing the value of TransitionStep to the LastStep value (**1**) of the query component transition sequence.
   - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
     - @QueryComponentID = **{AEC819CB-4CE8-4382-9E58-2CAD65ACDA99}**
     - @TransitionStep = **1**
     - Back-End Database Server sets the query component's TransitionStep value to **1** and returns 0.
   - Creates two refactored full-text index catalogs each (one for each index partition in the new query topology) for both the main catalog and anchor text catalog. The refactored full-text index catalogs for the main catalog includes the following files and content:
     - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.ci, containing all data for items whose document identifiers (1) modulo 256 are between 0 and 127, inclusive
     - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.cix, containing all data for items whose document identifiers(1) modulo 256 are between 0 and 127, inclusive
     - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.dir
     - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.bsi, containing all data for items whose document identifiers (1) modulo 256 are between 0 and 127, inclusive

- \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.bsd
- \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.00000001.csi, containing all data for items whose document identifiers (1) modulo 256 are between 0 and 127, inclusive
- \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.00000001.csd
- \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.wid
- \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\02020001.ci, containing all data for items whose document identifiers (1) modulo 256 are between 128 and 255, inclusive
- \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\02020001.cix, containing all data for items whose document identifiers (1) modulo 256 are between 128 and 255, inclusive
- \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\02020001.dir
- \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\02020001.bsi, containing all data for items whose document identifiers (1) modulo 256 are between 128 and 255, inclusive
- \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\02020001.bsd
- \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\02020001.00000001.csi, containing all data for items whose document identifiers (1) modulo 256 are between 128 and 255, inclusive
- \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\02020001.00000001.csd

5. Application Server 1 completes the query component transition sequence.
   - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
     - @QueryComponentID = **{AEC819CB-4CE8-4382-9E58-2CAD65ACDA99}**
     - @DesiredState = **103** (IndexSplitDone)
     - @State = **103** (IndexSplitDone)
     - @TransitionStep = **-1**
     - @TransitionStatus = **1** (Complete)
     - @PauseRequested = **0**
     - @TransitionCancelRequested = **0**
     - Back-End Database Server sets the query component's DesiredState value to **IndexSplitDone**, State value to **IndexSplitDone**, TransitionStep value to **-1**,

TransitionStatus value to **Complete**, PauseRequested value to **false**, and TransitionCancelRequested value to **false**, and finally returns **0**.

6. Administration Server polls Back-End Database Server until it finds that the active query component (2) has reached the IndexSplitDone state.

   - Calls **proc_MSS_GetQueryComponents**, which returns a result set containing three results, one of which has the following values:

     - QueryComponentNumber = **0**
     - QueryComponentID = **{AEC819CB-4CE8-4382-9E58-2CAD65ACDA99}**
     - ServerName = **"server0"**
     - LocalStoragePath = **"C:\Index"**
     - PartitionID = **{0FB5791F-0255-4426-90DE-B338F208B3CF}**
     - DesiredState = **103** (IndexSplitDone)
     - DesiredStateSetTime = **August 2, 2009 11:39:01 PM**
     - HotSwap = **0**
     - ShareName = **"aec819cb-4ce8-4382-9e58-2cad65acda99-query-0"**
     - UsesCustomShare = **0**
     - State = **103** (IndexSplitDone)
     - LastPropagationTime = **August 2, 2009 3:35:01 PM**
     - TransitionSequenceName = **""**
     - TransitionStep = **-1**
     - TransitionStepStartTime = **August 2, 2009 11:40:13 PM**
     - TransitionStatus = **1** (Complete)
     - TransitionError = **""**
     - TransitionCancelRequested = **0**
     - SourceComponentID = **NULL**
     - SourceComponentPath = **NULL**
     - PauseRequested = **0**
     - SettingsInRegistry = **0**
     - ScopeCompilationID = **7**

7. Administration Server finishes the "SplittingIndexes" topology activation action.

   - Calls stored procedure **proc_MSS_UpdateTopologyActivationAction** with:

     - @ActionID = **2**

- @NewState = **2** (Finished)
- Back-End Database Server updates the State value of the topology activation action to **Finished** and returns **0**.

8. Administration Server starts the "InitializeAfterRepartition" topology activation action.
   - Calls **proc_MSS_UpdateTopologyActivationAction** stored procedure with:
     - @ActionID = **3**
     - @NewState = **1** (InProgress)
     - Back-End Database Server updates the State value of the topology activation action to **InProgress** and returns **0**.
9. Administration Server starts the "initialize from repartitioning" transition on both query components (2) in the new query topology. The activities of only one query component (2) (Application Server 2) are demonstrated here. Steps 10-14 are also performed on the server which hosts the second query component (2) (for example, Application Server 3).
   - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except
     - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
     - @DesiredState = **1** (Ready)
     - @TransitionSequenceName = **"initialize from repartitioning"**
     - @TransitionStep = **0**
     - @TransitionStatus = **0** (Executing)
     - @TransitionError = **""**
     - @TransitionCancelRequested = **0**
     - Back-End Database Server sets the query component's DesiredState value to **Ready**, TransitionSequenceName value to **"initialize from repartitioning"**, TransitionStep value to **0**, TransitionStatus value to **Executing**, TransitionError value to **""**, and TransitionCancelRequested value to **false**, and finally returns **0**.
   - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except
     - @QueryComponentID = **{B1699847-F435-4541-8D66-968813731961}**
     - @DesiredState = **1** (Ready)
     - @TransitionSequenceName = **"initialize from repartitioning"**
     - @TransitionStep = **0**
     - @TransitionStatus = **0** (Executing)
     - @TransitionError = **""**
     - @TransitionCancelRequested = **0**

- Back-End Database Server sets the query component's DesiredState value to **Ready**, TransitionSequenceName value to **"initialize from repartitioning"**, TransitionStep value to **0**, TransitionStatus value to **Executing**, TransitionError value to **""**, and TransitionCancelRequested value to **false**, and finally returns **0**.

10. Application Server 2 polls Back-End Database Server until it finds that one of the query components (2) should be performing a query component transition.

    - Calls **proc_MSS_GetQueryComponents**, which returns a result set containing three results, one of which has the following values:

      - QueryComponentNumber = **1**
      - QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
      - ServerName = **"server1"**
      - LocalStoragePath = **"C:\Index"**
      - PartitionID = **{0DE05E5E-D9A3-495d-9ACE-A15AE9664036}**
      - DesiredState = **1** (Ready)
      - DesiredStateSetTime = **August 2, 2009 11:41:51 PM**
      - HotSwap = **0**
      - ShareName = **"0ad33931-9b1b-4c29-885e-a1e951da8b59-query-1"**
      - UsesCustomShare = **0**
      - State = **0** (Uninitialized)
      - LastPropagationTime = **January 1, 1900 12:00:00 AM**
      - TransitionSequenceName = **"initialize from repartitioning"**
      - TransitionStep = **0**
      - TransitionStepStartTime = **August 2, 2009 11:41:51 PM**
      - TransitionStatus = **0** (Executing)
      - TransitionError = **""**
      - TransitionCancelRequested = **0**
      - SourceComponentID = **{AEC819CB-4CE8-4382-9E58-2CAD65ACDA99}**
      - SourceComponentPath = **NULL**
      - PauseRequested = **NULL**
      - SettingsInRegistry = **0**
      - ScopeCompilationID = **NULL**
      - TransitionSequenceName = **NULL**

- TransitionCancelRequested = **NULL**

11. Application Server 2 executes the "initialize from repartitioning" query component transition sequence until its **TransitionStep** is equal to the **CopyRefactoredCatalogStep** value (4) of the "initialize from repartitioning" transition.

    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @TransitionStatus = **0** (Executing)
        - Back-End Database Server sets the query component's TransitionStatus value to **Executing** and returns **0**.
    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @TransitionStep = **1**
        - Back-End Database Server sets the query component's TransitionStep value to **1** and returns **0**.
    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @TransitionStep = **2**
        - Back-End Database Server sets the query component's TransitionStep value to **2** and returns **0**.
    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @TransitionStep = **3**
        - Back-End Database Server sets the query component's TransitionStep value to **3** and returns **0**.
    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @TransitionStep = **4**
        - Back-End Database Server sets the query component's TransitionStep value to **4** and returns **0**.

12. Application Server 2 copies the following refactored full-text index catalog files from Application Server 1:

    - Portal_Content catalog
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.ci

        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.cix
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.dir
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.bsi
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.bsd
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.00000001.csi
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.00000001.csd
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\Portal_Content\Indexer\CiFiles\01020001.wid
    - AnchorProject catalog
        - \server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\AnchorProject\Indexer\CiFiles\01020001.ci
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\AnchorProject\Indexer\CiFiles\01020001.cix
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\AnchorProject\Indexer\CiFiles\01020001.dir
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\AnchorProject\Indexer\CiFiles\01020001.bsi
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\AnchorProject\Indexer\CiFiles\01020001.bsd
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\AnchorProject\Indexer\CiFiles\01020001.00000001.csi
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\AnchorProject\Indexer\CiFiles\01020001.00000001.csd
        - \\server0\aec819cb-4ce8-4382-9e58-2cad65acda99-query-0\Projects\AnchorProject\Indexer\CiFiles\01020001.wid
13. Application Server 2 executes the "initialize from repartitioning" query component transition sequence until its **TransitionStep** is equal to the **LastStep** value (10) of the "initialize from repartitioning" transition.
    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @TransitionStep = **5**

        - Back-End Database Server sets the query component's TransitionStep value to **5** and returns **0**.
    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @TransitionStep = **6**
        - Back-End Database Server sets the query component's TransitionStep value to **6** and returns **0**.
    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @TransitionStep = **7**
        - Back-End Database Server sets the query component's TransitionStep value to **7** and returns **0**.
    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @TransitionStep = **8**
        - Back-End Database Server sets the query component's TransitionStep value to **8** and returns **0**.
    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @TransitionStep = **9**
        - Back-End Database Server sets the query component's TransitionStep value to **9** and returns **0**.
    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @TransitionStep = **10**
        - Back-End Database Server sets the query component's TransitionStep value to **10** and returns **0**.

14. Application Server 2 completes the query component transition sequence.

    - Calls **proc_MSS_SetQueryComponent** with all parameters set to **NULL** except:
        - @QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
        - @DesiredState = **1** (Ready)
        - @State = **1** (Ready)
        - @TransitionStep = **-1**

- @TransitionStatus = **1** (Complete)
- @PauseRequested = **0**
- @TransitionCancelRequested = **0**
- Back-End Database Server sets the query component's DesiredState value to **Ready**, State value to **Ready**, TransitionStep value to **-1**, TransitionStatus value to **Complete**, PauseRequested value to **false**, and TransitionCancelRequested value to **false**, and finally returns **0**.

15. Administration Server polls Back-End Database Server until it finds that both activating query components have reached the IndexSplitDone state.

- Calls **proc_MSS_GetQueryComponents**, which returns a result set containing three results, including one with the following values:
  - QueryComponentNumber = **1**
  - QueryComponentID = **{0AD33931-9B1B-4C29-885E-A1E951DA8B59}**
  - ServerName = **"server1"**
  - LocalStoragePath = **"C:\Index"**
  - PartitionID = **{0DE05E5E-D9A3-495d-9ACE-A15AE9664036}**
  - DesiredState = **1** (Ready)
  - DesiredStateSetTime = **August 2, 2009 11:39:01 PM**
  - HotSwap = **0**
  - ShareName = **"0ad33931-9b1b-4c29-885e-a1e951da8b59-query-1"**
  - UsesCustomShare = **0**
  - State = **1** (Ready)
  - LastPropagationTime = **January 1, 1900 12:00:00 AM**
  - TransitionSequenceName = **"initialize from repartitioning"**
  - TransitionStep = **-1**
  - TransitionStepStartTime = **August 2, 2009 11:42:37 PM**
  - TransitionStatus = **1** (Completed)
  - TransitionError = **""**
  - TransitionCancelRequested = **0**
  - SourceComponentID = **{AEC819CB-4CE8-4382-9E58-2CAD65ACDA99}**
  - SourceComponentPath = **NULL**
  - PauseRequested = **0**

- SettingsInRegistry = **0**
- ScopeCompilationID = **NULL**

- and another with the following values:
  - QueryComponentNumber = **2**
  - QueryComponentID = **{B1699847-F435-4541-8D66-968813731961}**
  - ServerName = **"server2"**
  - LocalStoragePath = **"C:\Index"**
  - PartitionID = **{DF51A997-FB76-4378-9E2D-8B3101C9FA29}**
  - DesiredState = **1** (Ready)
  - DesiredStateSetTime = **August 2, 2009 11:39:01 PM**
  - HotSwap = **0**
  - ShareName = **"0ad33931-9b1b-4c29-885e-a1e951da8b59-query-1"**
  - UsesCustomShare = **0**
  - State = **1** (Ready)
  - LastPropagationTime = **January 1, 1900 12:00:00 AM**
  - TransitionSequenceName = **"initialize from repartitioning"**
  - TransitionStep = **-1**
  - TransitionStepStartTime = **August 2, 2009 11:42:41 PM**
  - TransitionStatus = **1** (Completed)
  - TransitionError = **""**
  - TransitionCancelRequested = **0**
  - SourceComponentID = **{AEC819CB-4CE8-4382-9E58-2CAD65ACDA99}**
  - SourceComponentPath = **NULL**
  - PauseRequested = **0**
  - SettingsInRegistry = **0**
  - ScopeCompilationID = **NULL**

16. Administration Server finishes the "InitializeAfterRepartition" action.

- Calls stored procedure **proc_MSS_UpdateTopologyActivationAction** with
  - @ActionID = **3**
  - @NewState = **2** (Finished)

- Back-End Database Server updates the state of the topology activation action and returns **0**.

# 5 Security

## 5.1 Security Considerations for Implementers

Security for this protocol is controlled by the access rights to the databases on the back-end database server, which is negotiated as part of the TDS protocol, as described in [MS-TDS].

To call stored procedures administration servers and application servers run as an account that has read and write permissions on the back-end database server.

To copy files from other servers, application servers run as an account that is a member of the local security **group(2)** named "WSS_WPG".

## 5.2 Index of Security Parameters

None.

# 6 Appendix A: Product Behavior

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

- Microsoft® FAST™ Search Server 2010
- Microsoft® SharePoint® Foundation 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.

<1> Section 3.1.5.3: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<2> Section 3.1.5.4: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<3> Section 3.1.5.31: In SharePoint Foundation 2010 the @VersionID parameter is always set to "E54BBEDA-DB65-4F86-AAD7-E37C28026C2B". In this case the version returned by this stored procedure identifies version of all search protocols.

<4> Section 3.1.5.31: Following version values are returned in all products except SharePoint Foundation 2010:

| Protocol | Version |
|---|---|
| SQL Administration Protocol and Search Topology Protocol. | "13.0.214.0" |
| Search Service Database Query Protocol | "13.0.33.0" |
| SQL Gatherer Protocol | "13.0.63.0" |

In SharePoint Foundation 2010 the stored procedure returns version "4.0.191.0".

<5> Section 3.1.5.31: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<6> Section 3.1.5.32: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<7> Section 3.1.5.33: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<8> Section 3.1.5.34: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<9> Section 3.1.5.50: If a given stored procedure does an INSERT, UDPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<10> Section 3.1.5.56: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<11> Section 3.1.5.57: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<12> Section 3.1.5.58: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<13> Section 3.1.5.64: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<14> Section 3.1.5.65: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<15> Section 3.1.5.66: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<16> Section 3.1.5.67: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<17> Section 3.1.5.69: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<18> Section 3.1.5.72: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<19> Section 3.1.5.73: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<20> Section 3.1.5.74: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<21> Section 3.1.5.75: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<22> Section 3.1.5.76: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<23> Section 3.1.5.77: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<24> Section 3.1.5.80: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<25> Section 3.1.5.82: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<26> Section 3.1.5.87: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<27> Section 3.1.5.88: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<28> Section 3.1.5.89: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<29> Section 3.1.5.91: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<30> Section 3.1.5.95: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<31> Section 3.1.5.97: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

# 7 Change Tracking

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

# 8 Index

**E**



**F**



**G**



**H**



**I**



**L**



**M**

**N**



**O**



**P**

## Q



## R

## S

## T

## V



## X