

[MS-SQLPGAT]: SQL Gatherer Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft's Open Specification Promise (available here: <http://www.microsoft.com/interop/osp>) or the Community Promise (available here: <http://www.microsoft.com/interop/cp/default.msp>). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplq@microsoft.com.
- **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
04/04/2008	0.1		Initial Availability
06/27/2008	1.0	Major	Revised and edited the technical content
10/06/2008	1.01	Editorial	Revised and edited the technical content
12/12/2008	1.02	Editorial	Revised and edited the technical content
07/13/2009	1.03	Major	Changes made for template compliance
08/28/2009	1.04	Editorial	Revised and edited the technical content
11/06/2009	1.05	Editorial	Revised and edited the technical content
02/19/2010	2.0	Editorial	Revised and edited the technical content
03/31/2010	2.01	Editorial	Revised and edited the technical content
04/30/2010	2.02	Editorial	Revised and edited the technical content
06/07/2010	2.03	Editorial	Revised and edited the technical content
06/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	2.04	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1	Introduction	6
1.1	Glossary	6
1.2	References.....	7
1.2.1	Normative References.....	7
1.2.2	Informative References	8
1.3	Protocol Overview (Synopsis)	8
1.4	Relationship to Other Protocols.....	16
1.5	Prerequisites/Preconditions	17
1.6	Applicability Statement.....	17
1.7	Versioning and Capability Negotiation.....	17
1.8	Vendor-Extensible Fields.....	18
1.9	Standards Assignments	18
2	Messages.....	19
2.1	Transport.....	19
2.2	Common Data Types	19
2.2.1	Simple Data Types and Enumerations	19
2.2.2	Simple Data Types	19
2.2.2.1	Project Identifier.....	19
2.2.2.2	Crawl Type	19
2.2.2.3	Crawl Status	19
2.2.2.4	Transaction Type	20
2.2.2.5	Transaction Scope	20
2.2.2.6	Transaction Flags.....	20
2.2.2.7	Full Incremental Interval	21
2.2.2.8	Delete On Error Interval	21
2.2.2.9	Index Type	21
2.2.2.10	Filter Behavior	21
2.2.2.11	Rule Type	21
2.2.2.12	UrlRule Type	21
2.2.2.13	Compilation State Type	22
2.2.2.14	Managed Type	22
2.2.3	Bit Fields and Flag Structures.....	22
2.2.4	Binary Structures	22
2.2.5	Common Result Sets	23
2.2.5.1	Scopes Result Set.....	23
2.2.6	Tables and Views	23
2.2.6.1	MSSAnchorChangeLog.....	23
2.2.6.2	MSSAnchorText	23
2.2.6.3	MSSAnchorTransactions	24
2.2.6.4	MSSCrawledPropSamples	24
2.2.6.5	MSSSessionDefinitions/MSSSessionDefinitionsAlt	25
2.2.6.6	MSSSessionDocProps/MSSSessionDocPropsAlt	25
2.2.6.7	MSSSessionDocSdIds/MSSSessionDocSdIdsAlt	26
2.2.6.8	MSSSessionDocSignatures/MSSSessionDocSignaturesAlt	26
2.2.6.9	MSSSessionDuplicateHashes/MSSSessionDuplicateHashesAlt	27
2.2.6.10	MSSSessionExistingDocs/MSSSessionExistingDocsAlt	27
2.2.6.11	MSSTranTempTable0.....	27
2.2.7	XML Structures	30

3 Protocol Details	31
3.1 MOSS Server Details	31
3.1.1 Abstract Data Model	31
3.1.1.1 Metadata Functionality	33
3.1.1.2 Search Scopes Functionality	33
3.1.2 Timers	35
3.1.3 Initialization	35
3.1.4 Message Processing Events and Sequencing Rules	35
3.1.4.1 proc_MSS_AddAndReturnCrawledProperty	38
3.1.4.2 proc_MSS_AddCrawledPropertyCategoryWithDefaults	39
3.1.4.3 proc_MSS_CommitAnchorTextCrawl	40
3.1.4.4 proc_MSS_Crawl	40
3.1.4.5 proc_MSS_FlushTemp0	42
3.1.4.6 proc_MSS_GetBigConfigurationProperty	43
3.1.4.6.1 Big Configuration Property Result Set	43
3.1.4.7 proc_MSS_GetCompiledScopeRules	43
3.1.4.7.1 Compiled Scope Rules Result Set	44
3.1.4.8 proc_MSS_GetCompiledScopes	44
3.1.4.8.1 Scopes Result Set	44
3.1.4.9 proc_MSS_GetCompletedScopesCompilationID	45
3.1.4.10 proc_MSS_GetConfigurationProperty	45
3.1.4.11 proc_MSS_GetCrawledPropertyUpdates	45
3.1.4.11.1 GetCrawledPropertyUpdates Result Set	46
3.1.4.12 proc_MSS_GetCrawledPropMappingUpdates	46
3.1.4.12.1 GetCrawledPropMappingUpdates Result Set	47
3.1.4.13 proc_MSS_GetCrawls	47
3.1.4.13.1 GetCrawls Result Set	48
3.1.4.14 proc_MSS_GetDeletedScopesForCompilation	48
3.1.4.14.1 Deleted Search Scopes Result Set	48
3.1.4.15 proc_MSS_GetDocStatus	48
3.1.4.15.1 Document Status Result Set	49
3.1.4.16 proc_MSS_GetManagedProperties	49
3.1.4.16.1 GetManagedProperties Result Set	49
3.1.4.17 proc_MSS_GetNextCrawlBatch	51
3.1.4.17.1 GetNextCrawlBatch Result Set	52
3.1.4.18 proc_MSS_GetSampleExtremes	54
3.1.4.18.1 GetSampleExtremes Result Set	54
3.1.4.19 proc_MSS_GetSchemaHighLevelInfo	54
3.1.4.19.1 GetSchemaHighLevelInfo Result Set	55
3.1.4.20 proc_MSS_GetSchemaMappings	56
3.1.4.20.1 Schema Mappings Result Set	56
3.1.4.21 proc_MSS_GetSchemaParameters	56
3.1.4.21.1 Schema Parameters Result Set	56
3.1.4.22 proc_MSS_GetScopeRulesForCompilation	57
3.1.4.22.1 Pending Scope Rules Result Set	57
3.1.4.23 proc_MSS_GetScopesForCompilation	58
3.1.4.24 proc_MSS_GetSDID	58
3.1.4.24.1 GetSDID Result Set	58
3.1.4.25 proc_MSS_InsertFromSession	59
3.1.4.26 proc_MSS_OnDocDelete	59
3.1.4.27 proc_MSS_OnEndCrawl	59
3.1.4.28 proc_MSS_OnStartCrawl	60
3.1.4.29 proc_MSS_PrepareAnchorTextCrawl	61

3.1.4.30	proc_MSS_ProcessCommitted.....	61
3.1.4.31	proc_MSS_PushSD.....	64
3.1.4.32	proc_MSS_ReportScopesCompilationBegin.....	64
3.1.4.33	proc_MSS_ReportScopesCompilationEnd.....	65
3.1.4.34	proc_MSS_ResetCatalog.....	65
3.1.4.35	proc_MSS_SetBigConfigurationProperty.....	65
3.1.4.36	proc_MSS_SetConfigurationProperty.....	66
3.1.4.37	proc_MSS_SetCrawledPropertyIsSampleCacheFull.....	66
3.1.4.38	proc_MSS_ShareScopesCompilationInfo.....	67
3.1.4.39	proc_MSS_TruncateCleanupTable.....	67
3.1.4.40	proc_MSS_Recompile.....	68
3.1.5	Timer Events.....	68
3.1.6	Other Local Events.....	68
3.2	WSS Server Details.....	68
3.2.1	Abstract Data Model.....	68
3.2.2	Timers.....	68
3.2.3	Initialization.....	68
3.2.4	Message Processing Events and Sequencing Rules.....	68
3.2.4.1	proc_MSS_GetConfigurationProperty.....	70
3.2.5	Timer Events.....	71
3.2.6	Other Local Events.....	71
4	Protocol Examples.....	72
4.1	Full Crawl.....	72
4.2	Incremental Crawl.....	73
4.3	Delete Crawl.....	74
5	Security.....	76
5.1	Security Considerations for Implementers.....	76
5.2	Index of Security Parameters.....	76
6	Appendix A: Product Behavior.....	77
7	Change Tracking.....	79
8	Index.....	80

1 Introduction

This document specifies the communication sequences that are used by the Client (Web and application servers) to perform data query and update commands on the Server in relation to search crawl (content indexing) functions.

1.1 Glossary

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

access control list (ACL)
GUID
HRESULT
language code identifier (LCID)

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

access URL
anchor text
back-end database server
big configuration property
binary large object (BLOB)
change log
compact URL
configuration property
content source
crawl
crawl queue
crawl status
crawl URL history
crawled property
crawled property category
crawled property set identifier
datetime
delete crawl
display URL
excluded item
farm
folder
full crawl
full-text index catalog
host hop
host name
incremental crawl
index server
item
managed property
mapping order
metadata index
metadata schema
page hop
parent item
pluggable security authentication
portal content
property oriented rank

protocol
query independent rank
query server
result set
return code
search application
search catalog
search database
search scope
search scope compilation
search scope compilation identifier
search scope index
search scope rule
search scopes system
search security descriptor
search shared application object
start address
stored procedure
subdomain
transaction
Transact-Structured Query Language (T-SQL)
Uniform Resource Locator (URL)
variant type
vector configuration property

The following terms are specific to this document:

alternate access mapping: A mapping of URLs to Web applications. Incoming alternate access mappings are used to provide multiple URL entry points for the same set of content. Outgoing alternate access mappings are used to ensure that content is rendered in the correct URL context.

authority page: A Web page that a site collection administrator designated as more relevant than other Web pages. This is typically the URL of the home page for the intranet of an organization. The higher the authority level assigned to a page, the higher the page appears in search results. Also referred to as authoritative page.

high priority folder: A container that contains items that need to be processed before other items that are already in a queue.

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

1.2 References

1.2.1 Normative References

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

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

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

[MS-ERREF] Microsoft Corporation, "[Windows Error Codes](#)".

[MS-SQL] 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://msdn.microsoft.com/en-us/library/dd631854\(v=SQL.10\).aspx](http://msdn.microsoft.com/en-us/library/dd631854(v=SQL.10).aspx)

[MS-SQLPADM] Microsoft Corporation, "[SQL Administration Protocol Specification](#)"

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

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

1.2.2 Informative References

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

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

1.3 Protocol Overview (Synopsis)

This protocol specifies the communication between the **index server** and **back-end database server** used to satisfy requests for search **crawl** tasks. This server-to-server protocol uses the Tabular Data Stream Protocol (as specified in [\[MS-TDS\]](#)) as its transport between the index server and the back-end database server. Two distinct roles are served by the back-end database server in the protocol:

- Microsoft Office SharePoint Server (MOSS) search crawl administration role: This role serves the crawl requests for a MOSS **search application**.
- Windows® SharePoint® Services search crawl administration role.

This protocol is used by the index server to do full, incremental, delete and anchor text crawls. The diagram in the following diagram specifies the data flow between the protocol client, the index server and the protocol server, and the back-end database server with regards to performing a full crawl.

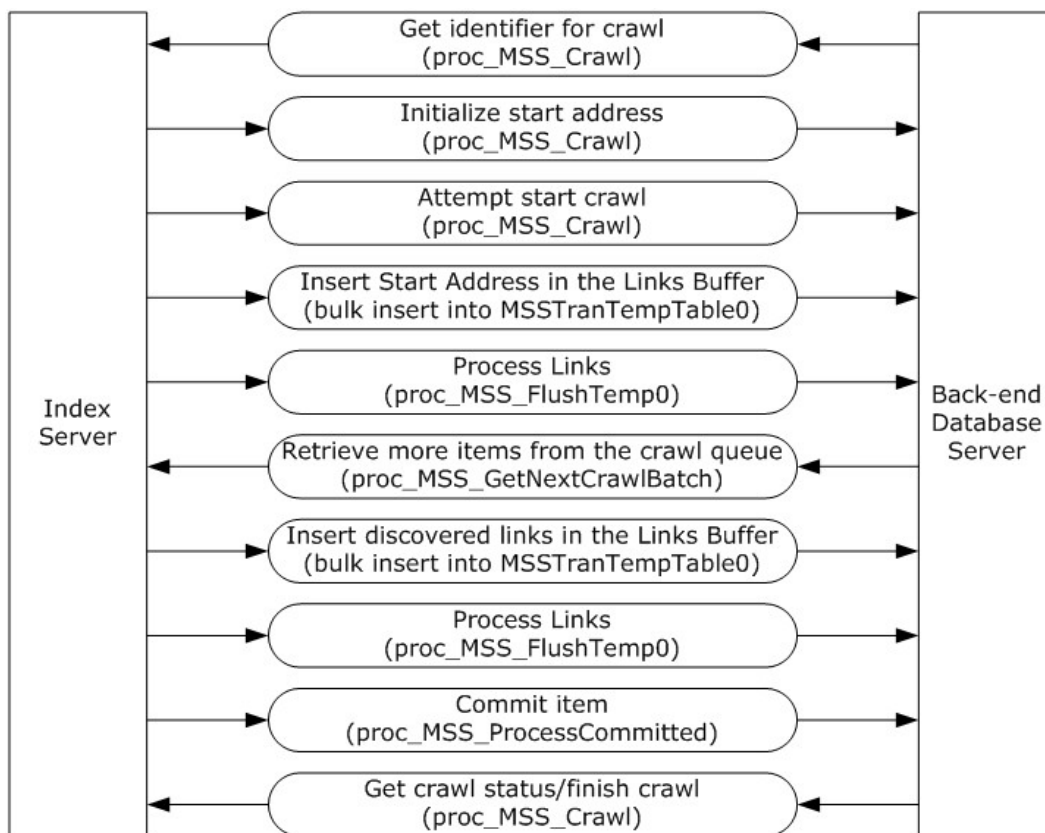


Figure 1: Flow of data during a full crawl operation

The preceding diagram specifies the data flow for a **full crawl**, the protocol client (index server) creates a crawl by getting an identifier for the crawl from the server. The client then makes a call to initialize each start address for the **content source**. The client then attempts to start the crawl, and ignores the request if there is another crawl of the same content source already in progress. The client prepares the crawl by inserting the start addresses into the links buffer, as specified in **Abstract Data Model** (section [3.1.1](#)). Then the client initiates links processing by calling the server to put the start addresses into the **crawl queue** and update the **crawl URL history**.

The index server periodically calls the server to retrieve **items** to crawl from the crawl queue. For each item, the links discovered by the index server are inserted into the links buffer, as specified in **Abstract Data Model** (section [3.1.1](#)). These links are later processed by the server:

- If the links are not in the crawl URL history then they are added to it.
- If the links have not yet been crawled, they are inserted into the crawl queue.
- The links are persisted in the anchor text information structure, as specified in **Abstract Data Model** (section [3.1.1](#)).

After processing the links, the index server calls the server to commit the item, setting the item status as completed and removing the item from the **crawl queue**, as specified in **Abstract Data Model** (section [3.1.1](#)).

The index server periodically calls the server to retrieve the **crawl status**. If the crawl queue is empty, and items need to be deleted by this crawl, then those items are inserted into the crawl queue to be deleted. If the crawl queue is empty and there are no items that need to be deleted, the server completes the crawl and returns a crawl complete status.

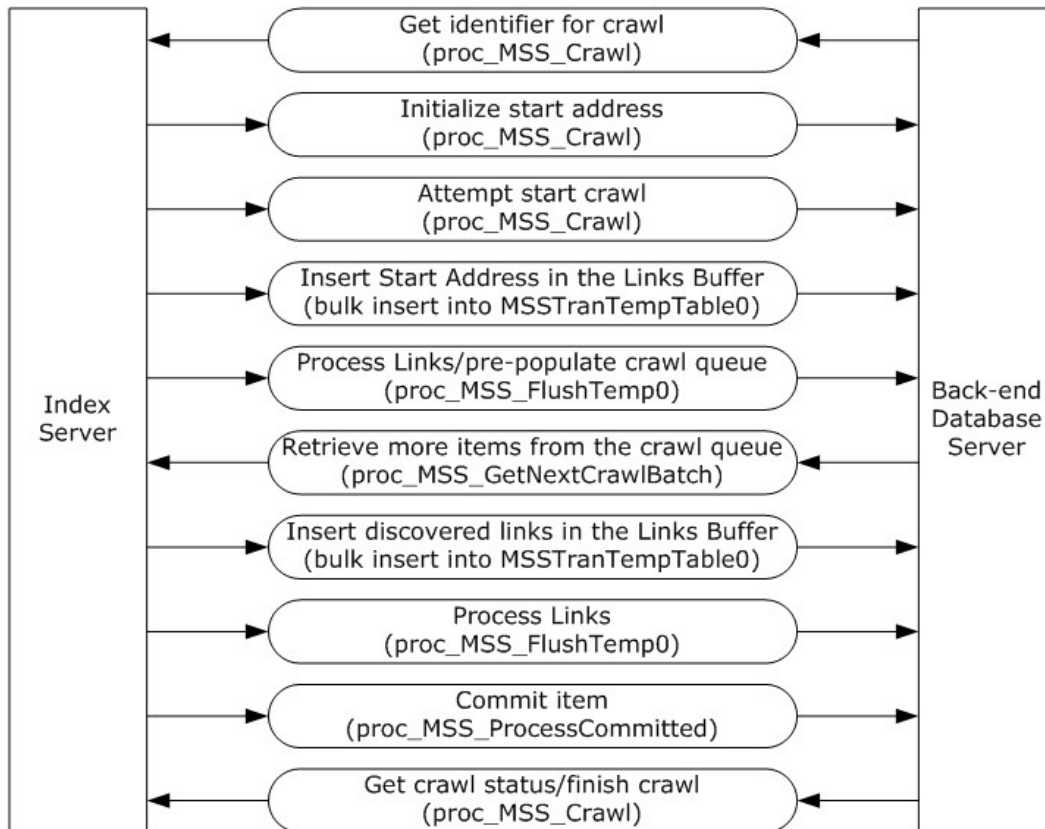


Figure 2: Flow of data during an incremental crawl operation

The data flow, shown in the preceding diagram, for an **incremental crawl** is similar to the **full crawl**. The main difference is that after the start addresses are inserted into the **links buffer**, as specified in **Abstract Data Model** (section [3.1.1](#)), the process links action can pre-populate the **crawl queue** based on the **crawl url history**, as specified in **Abstract Data Model** (section [3.1.1](#)).

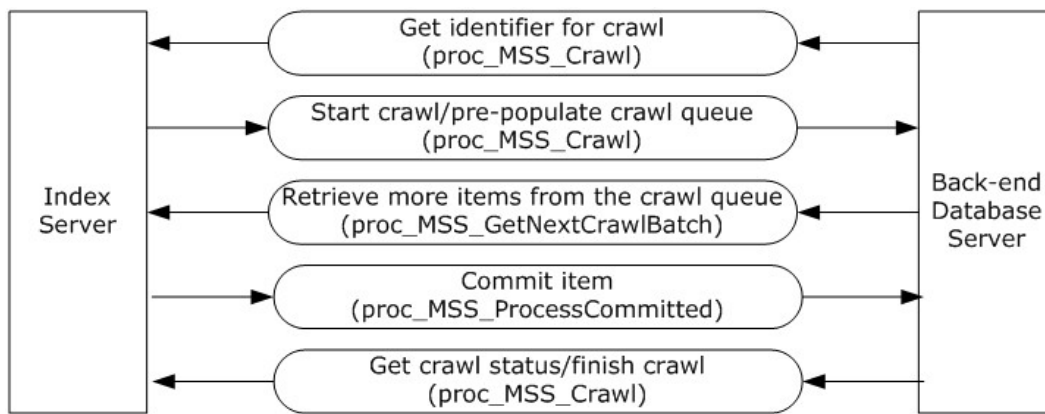


Figure 3: Flow of data during a delete crawl operation

The **delete crawl** operation removes items which are associated with a deleted content source or start address from the full-text and metadata indices. The protocol client (index server) creates a crawl by getting an identifier for the crawl from the server, as it does with other types of crawls. The client then makes a call to pre-populate the **crawl queue** to delete the items associated with a given content source or start address. As with other types of crawls, the index server periodically calls the server to retrieve items to crawl from the **crawl queue**. The index server periodically calls the server to retrieve the **crawl status**. If the crawl queue is empty, the server completes the crawl and returns a crawl complete status.

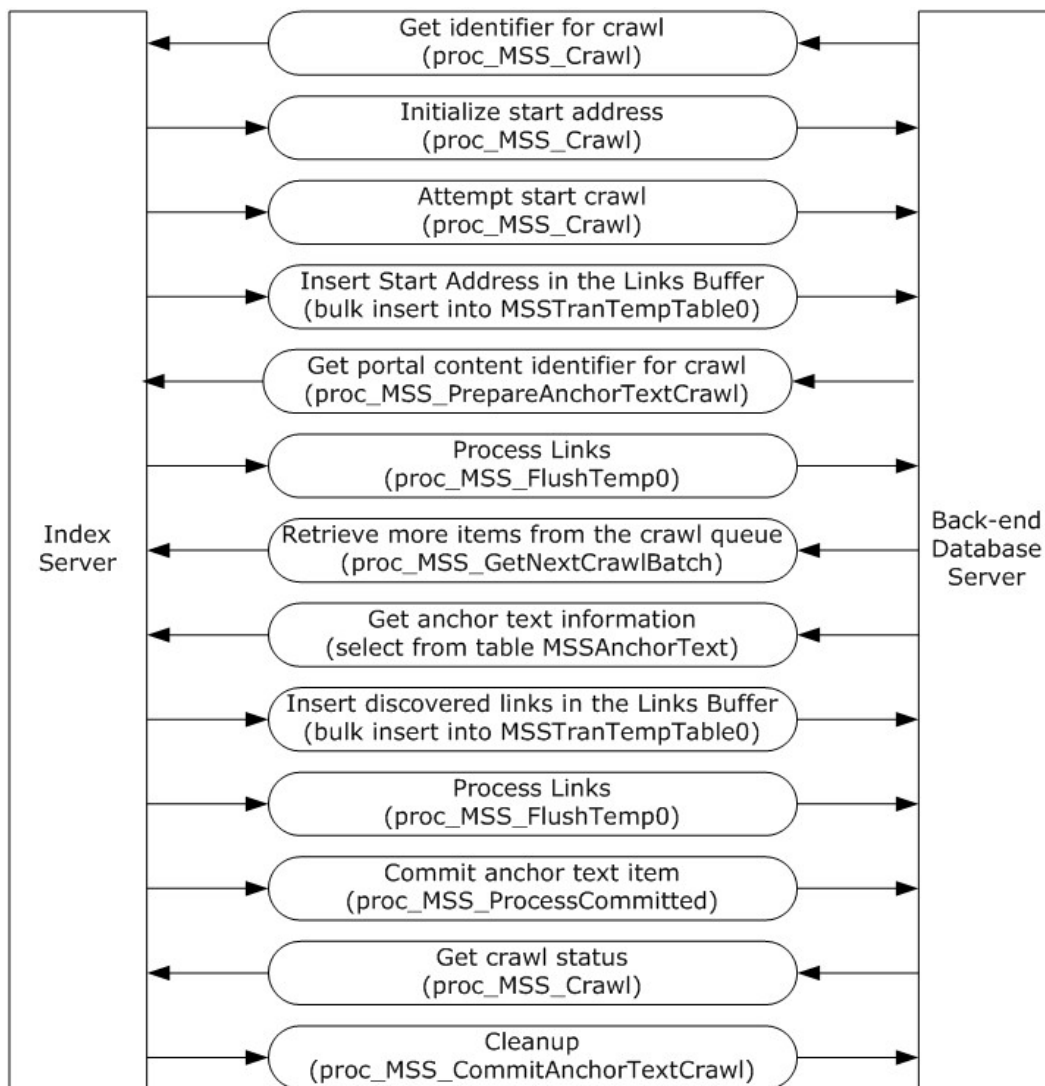


Figure 4: Flow of data during an anchor text crawl operation

In the **anchor text** crawl, the client creates a crawl by getting an identifier for the crawl from the server. The client then makes a call to initialize the internal **start address**. The client then attempts to start the crawl, aborting if there is another crawl of the same content source already in progress. The client prepares the crawl by inserting the start address into the links buffer, as specified in **Abstract Data Model** (section [3.1.1](#)). The client calls the server to prepare the anchor text crawl and gets the oldest portal content crawl identifier that has unprocessed anchor text information. Then the client initiates links processing by calling the server to put the start address into the **crawl queue**, as specified in **Abstract Data Model** (section [3.1.1](#)).

The index server periodically calls the server to retrieve items to crawl from the **crawl queue** and remove from the crawl queue all items in the anchor text completed items structure, as specified in **Abstract Data Model** (section [3.1.1](#)). The client retrieves the anchor text information from the server and inserts internal links into the **links buffer**, as specified in **Abstract Data Model** (section [3.1.1](#)). For each item that generates links, the index server makes the call to process the links and then sets the item status as completed by committing the anchor text item. If the item doesn't

produce a link, the index server inserts the item into the anchor text completed items structure, as specified in **Abstract Data Model** (section [3.1.1](#)). The index server periodically calls the server to retrieve the **crawl status**. If the crawl queue is empty, the server completes the crawl and returns a crawl complete status.

When the crawl is done, the index server cleans up internal data structures created at the beginning of the crawl.

As part of the index server crawl flows given in this section, two alternate types of data flow occur:

- Metadata information is collected in the **metadata index**.
- Search scope information is collected in the **search scope index**.

The following subsections specify the Metadata and Search Scope data flow.

▪ **Metadata Crawl Data Flow**

The following diagram specifies the data flow of metadata during full and incremental crawls:

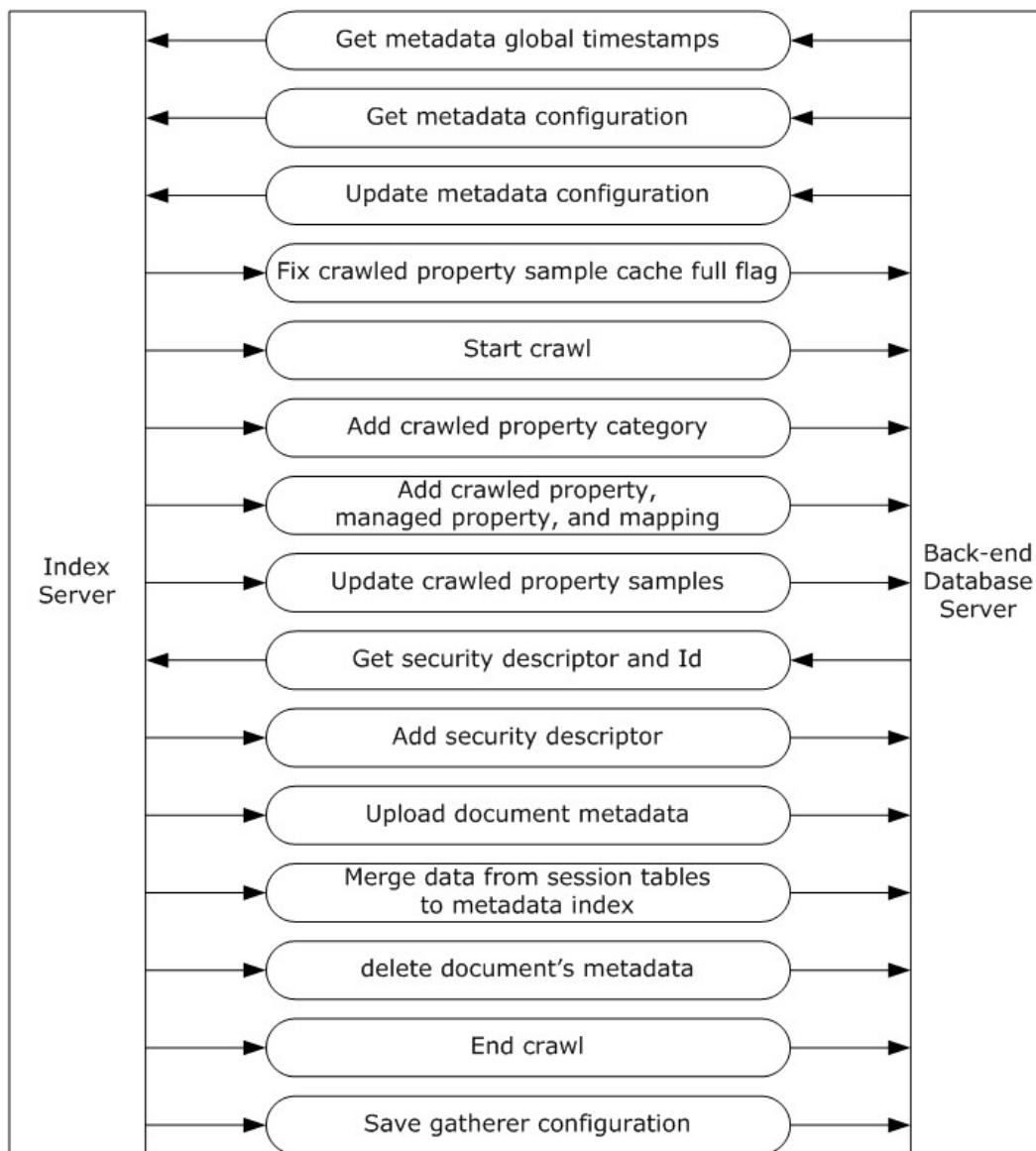


Figure 5: Flow of metadata during full and incremental crawl operations

On startup, the crawl process retrieves configuration data from the backend database server (using `proc_MSS_GetConfigurationProperty` in section 3.1.4.10 and `proc_MSS_GetBigConfigurationProperty` in section 3.1.4.6). The metadata global timestamps are retrieved in **datetime** format and stored for comparison later (using `proc_MSS_GetSchemaHighLevelInfo` in section 3.1.4.19). Then metadata configuration such as **crawled property categories** (using `proc_MSS_GetCrawledPropertyCategoriesBasic` in [MS-SQLPADM], section 3.1.4.54), **crawled properties** (using `proc_MSS_GetCrawledPropertiesBasic` in [MS-SQLPADM], section 3.1.4.53), **managed properties** (using `proc_MSS_GetManagedProperties` in section 3.1.4.16), and mappings are retrieved (using `proc_MSS_GetSchemaMappings` in section 3.1.4.20). In addition, ranking parameters and configuration properties are retrieved (using `proc_MSS_GetSchemaParameters` in section 3.1.4.21).

The metadata global timestamps are retrieved multiple times each minute and used to update the metadata configuration information within the index server. These timestamps are used to retrieve updates for crawled property categories (using `proc_MSS_GetCrawledPropertyCategoriesBasic` in [MS-SQLPADM], section 3.1.4.54), crawled properties (using `proc_MSS_GetCrawledPropertyUpdates` in section 3.1.4.11), managed properties (using `proc_MSS_GetManagedProperties` in section 3.1.4.16), and mappings between crawled and managed properties (using `proc_MSS_GetCrawledPropMappingUpdates` in section 3.1.4.12).

Crawled property sample counts are checked to be within tolerance as well (using `proc_MSS_GetSampleExtremes` in section 3.1.4.18). If the count is below the "lower limit" parameter, the sample cache full flag is set to 1, or if the count is above the "higher limit" parameter, the sample cache full flag is set to 0, (using `proc_MSS_SetCrawledPropertyIsSampleCacheFull` in section 3.1.4.37). The cleanup is then executed to refresh structures that are used to calculate the sample counts (using `proc_MSS_TruncateCleanupTable` in section 3.1.4.39).

When a crawl is started, a start crawl procedure is executed on the backend database server (using `proc_MSS_OnStartCrawl` in section 3.1.4.28). Data from crawled items are then processed. Each chunk of data is tagged with a **crawled property set identifier** and a property name. If the crawled property set identifier is not found in the crawled property category set, a crawled property category is added (using `proc_MSS_AddCrawledPropertyCategoryWithDefaults` in section 3.1.4.2). If a new crawled property is discovered, it is added (using `proc_MSS_AddAndReturnCrawledProperty` in section 3.1.4.1). If the newly discovered crawled property is configured to be automatically mapped to a managed property, a managed property mapping between them is also created. If applicable, crawled property samples are then uploaded (using a bulk upload to `MSSCrawledPropSamples` in section 2.2.6.4) and, if applicable, the crawled property sample cache full flag is set to 0 (using `proc_MSS_SetCrawledPropertyIsSampleCacheFull` in section 3.1.4.37).

If the item's security descriptor is not cached, the security descriptor is retrieved (using `proc_MSS_GetSDID` in section 3.1.4.24). If not found, the security descriptor is added (using `proc_MSS_PushSD` in section 3.1.4.31).

The item's metadata properties are then uploaded to a temporary repository which includes the session tables described in section 2.2.5. The upload toggles between the session tables with and without the "alt" suffix. The data is then merged into the metadata index from the session tables (using `proc_MSS_InsertFromSession` in section 3.1.4.25).

When the crawl detects that an item should be deleted, the item's metadata is deleted from the metadata index (using `proc_MSS_OnDocDelete` in section 3.1.4.26).

When the crawl is finished, an end crawl procedure is executed (using `proc_MSS_OnEndCrawl` in section 3.1.4.27). The index server saves its configuration to the server either before shutdown or after a configuration change (using `proc_MSS_SetBigConfigurationProperty` in section 3.1.4.35).

▪ Search Scope Data Flow During Crawler Startup and Search Scope Update

The following diagram specifies the flow of all search scope-related metadata between the index server and the back-end database server during startup and search scope update.

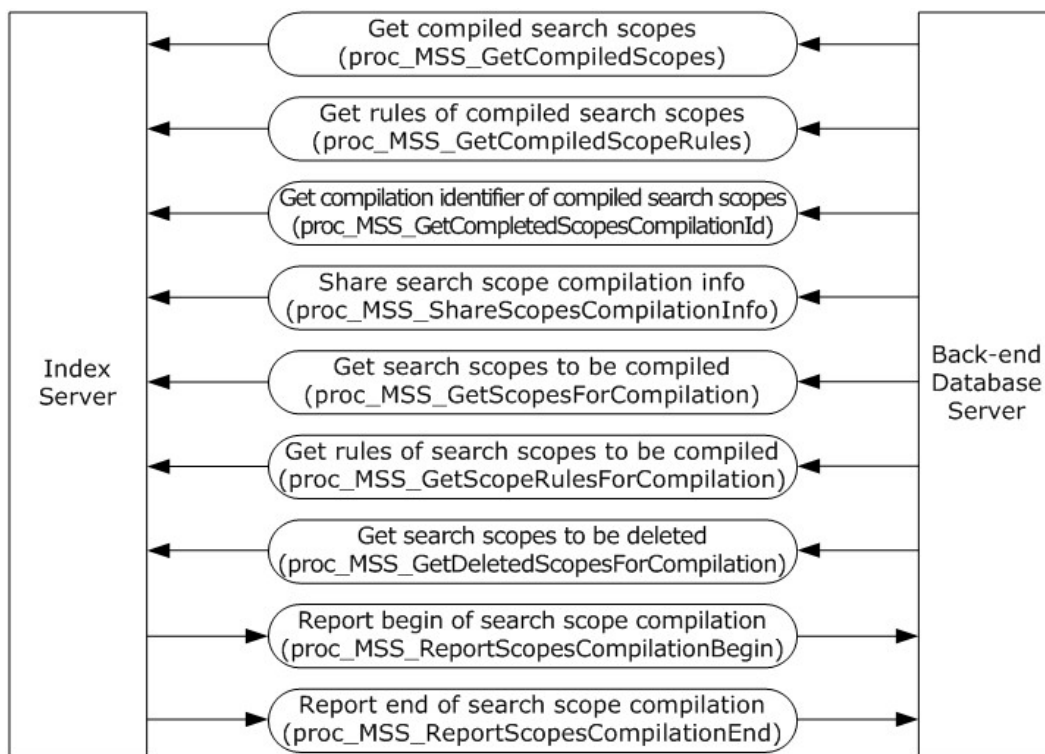


Figure 6: Flow of search scope metadata during crawler startup and search scope update

When the **search catalog** is reset or **Office SharePoint Server Search (Osearch) service** is started, the back-end database server sends information about compiled search scopes and compiled search scope rules to the index server. The stored procedure that gets the **search scope compilation identifier** of compiled search scopes is called only when the search catalog is reset. The **stored procedure** that shares search scope compilation information is called every 30 seconds to send search scope compilation information from database server to index server. When search scopes are updated, the back-end database server sends information about search scopes to be compiled, search scope rules to be compiled, and search scopes to be deleted to the index server. The index server informs the back-end database server about the beginning and end of search scope compilation.

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:

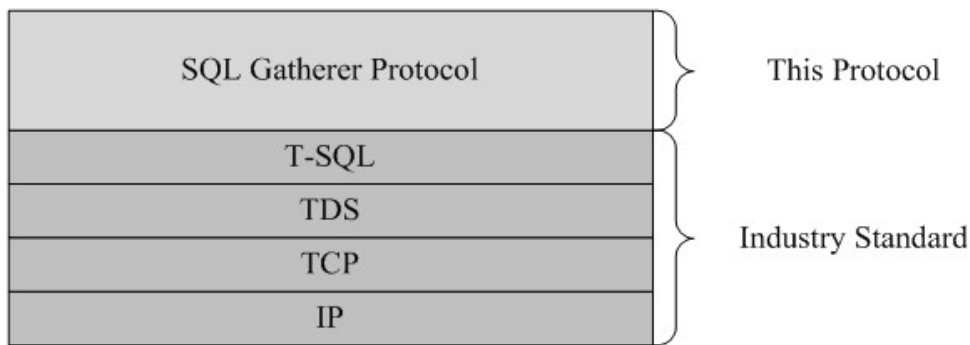


Figure 7: This protocol in relationship to other protocols

1.5 Prerequisites/Preconditions

This protocol requires that a **farm** is installed and configured. The operations described by the protocol operate between a client that is a part of a farm and a back-end database server on which the databases of the farm are stored.

Unless otherwise specified, the stored procedures and any related tables are present in the database that is being queried on the back-end database server. The tables in the database contain valid data in a consistent state in order to be queried successfully by the stored procedures.

The user that calls the stored procedures has adequate permission to access the databases that contain the stored procedures.

1.6 Applicability Statement

This protocol is only applicable to index servers when communicating with the back-end database server to satisfy requests for common search crawl tasks.

This protocol is intended for use by protocol clients and protocol servers that are both connected by high-bandwidth, low latency network connections.

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. If the stored procedures are not provided the expected calling parameters or return code values, the results of the call are indeterminate.

This document covers versioning issues in the following areas:

Security and Authentication Methods

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

1.8 Vendor-Extensible Fields

This protocol uses HRESULT values as defined in [\[MS-ERREF\]](#), section 2.1. Vendors can define their own HRESULT values, provided they set the C bit (0x20000000) for each vendor-defined value, indicating that the value is a customer code.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

[\[MS-TDS\]](#) is the transport protocol used to call the stored procedures, SQL Tables and return result codes and result sets.

2.2 Common Data Types

The following sections define the common data types that are used in this protocol.

2.2.1 Simple Data Types and Enumerations

2.2.2 Simple Data Types

2.2.2.1 Project Identifier

A unique identifier of a crawl. Valid values are listed in the table below:

Value	Description
1	Portal Content
2	Anchor Text
3	People Profile

2.2.2.2 Crawl Type

The type of the crawl. The value MUST be in the following table:

Value	Description
1	Full crawl
2	Incremental crawl
6	Delete crawl

2.2.2.3 Crawl Status

The state of the crawl. The value MUST be in the following table:

Value	Description
1	Initializing
2	Start addresses are being initialized
4	Started
5	Forbidden. The crawl cannot start either because another crawl of the same content is already in progress, or there are no valid start addresses in the content source that got the crawl start request.

Value	Description
11	Done
9	Paused
12	Stopped

2.2.2.4 Transaction Type

The type of action for the current item. The value MUST be in the following table:

Value	Description
0	Add item
1	Delete item
2	Modify item
3	Move or Rename item

2.2.2.5 Transaction Scope

The scope of the given **transaction**. The value MUST be in the following table:

Value	Description
1	Only the current item is processed in the current crawl.
2	This item and all children are processed in the current crawl.
4	Only the item access control list (ACL) is updated. The item content is not processed. This item and all children are processed in the current crawl.

2.2.2.6 Transaction Flags

A bit mask of flags. The value MUST be in the following table:

Value	Description
0x0004	The item is a container.
0x0008	If the item is a container, the links in the item have to be processed by the current crawl.
0x0040	The item has a valid UTC last modified timestamp.
0x0200	When the item is discovered as a result of enumerating the parent container, the last modified time of the item is always available.
0x0400	The item lives in a root container that provides a UTC last modified time. This time is updated after any item in the container is added, modified or deleted.
0x0800	The incremental crawl processes the item only if the difference between the current time and the last processed time is greater than FullIncrementalInterval .

Value	Description
0x2000000	The current crawl updates only the item Access Control List (ACL); the item content is not processed.

2.2.2.7 Full Incremental Interval

The time interval (in minutes) between 2 consecutive incremental complete crawls.

2.2.2.8 Delete On Error Interval

The maximum number of consecutive errors allowed in the system for any given item. If the number of consecutive errors equals the value of this parameter, the item is removed from the crawl history and from the index.

2.2.2.9 Index Type

The value representing the crawl status for a given item where index type MUST be 1 if the item is indexed. Otherwise, it MUST be 0.

2.2.2.10 Filter Behavior

An integer that identifies how a **search scope rule** will filter items in the search catalog. The value MUST be in the following table:

Value	Description
0	Include: Items matching this rule will be included in the search scope .
1	Require: Items that don't match this rule will be excluded from the search scope.
2	Exclude: Items matching this rule will be excluded from this search scope.

2.2.2.11 Rule Type

An integer that **identifies the type** of the **search scope rule**. The value MUST be in the following table:

Value	Description
0	All Content: The search scope rule includes all items.
1	Url: The search scope rule includes items whose folder , host name , or subdomain matches the UserValueString attribute value as defined in Search Scope Rule Set in [MS-SQLPADM] , Section 3.1.1.3.
2	Property Query: The search scope rule includes items whose managed property value matches the UserValueString attribute value and whose managed property identifier matches PropertyId value. UserValueString and PropertyId attributes are defined in Search Scope Rule Set in [MS-SQLPADM] , Section 3.1.1.3.

2.2.2.12 UrlRule Type

An integer that identifies the type of the **search scope rule**. The value MUST be in the following table:

Value	Description
0	Folder
1	Host name
2	Subdomain

2.2.2.13 Compilation State Type

An integer that identifies the state of **search scope compilation** for the given search scope. The value MUST be in the following table:

Value	Description
0	Empty: There are no search scope rules.
1	Invalid: The search scope rules are invalid .
2	Query Expanded: There are not enough search scope rules (fewer than 25) and the compilation rules have been set to compile only if 25 or more search scope rules exist, so that compilation will not happen.
3	Needs Compilation.
4	Compiled.
5	Needs to be recompiled.

2.2.2.14 Managed Type

An integer that identifies the data type of the managed property. The value MUST be in the following table:

Value	Description
1	String which is a Unicode character array of arbitrary length.
2	64 bit integer.
3	64 bit decimal.
4	64 bit UTC date/time representing the number of 100-nanosecond intervals since January 1, 1601.
5	A boolean value, where -1 which is TRUE and everything else which is FALSE.
6	Binary large object (BLOB).

2.2.3 Bit Fields and Flag Structures

None.

2.2.4 Binary Structures

None.

2.2.5 Common Result Sets

2.2.5.1 Scopes Result Set

The Scopes result set returns a list of **search scopes**. The result set MUST contain zero or more rows, each corresponding to a single **search scope**.

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

```
ScopeID          int,  
CompilationState smallint;
```

ScopeID: An integer that uniquely identifies the **search scope**.

CompilationState: The **search scope** compilation state for the given search scope. The value MUST be a **Compilation State** Data Type, as specified in section [2.2.2.13](#).

2.2.6 Tables and Views

Tables with two different titles are duplicates that are used in an alternating sequence in the product.

2.2.6.1 MSSAnchorChangeLog

The MSSAnchorChangeLog table is used in the implementation of **anchor text information**, as specified in Abstract Data Model (Section [3.1.1](#)).

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 uniquely identifier of the **crawl**.

TargetDocId: An identifier of an item that the current crawl determines has changed since the previous crawl.

ChangeType: Indicator of link status for the given TargetDocId. The value MUST be in the following table.

Value	Description
1	There are no links pointing to this item.
2	There is at least one link that points to this item.

2.2.6.2 MSSAnchorText

The MSSAnchorText table is used in the implementation of **anchor text information** as specified in **Abstract Data Model**.

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

```
TABLE MSSAnchorText (
    SourceDocID          int NULL,
    TargetDocID          int NULL,
    Link                 nvarchar(2048) NULL,
    LCID                 int NULL,
    LinkId               int NOT NULL,
    LinkId               bigint NOT NULL,
    AnchorText           nvarchar(1024) NULL
);
```

SourceDocID: The identifier of the item that contains the link.

TargetDocID: The identifier of the item to which the link points.

Link: The link represented as a **URL**.

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

LinkId: The unique identifier of the link<1>.

2.2.6.3 MSSAnchorTransactions

The MSSAnchorTransactions table implements the **anchor text completed documents** structure, as specified in **Abstract Data Model** (section [3.1.1](#)).

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

```
TABLE MSSAnchorTransactions (
    DocID                int NOT NULL
);
```

DocID: The unique identifier of the item.

2.2.6.4 MSSCrawledPropSamples

The MSSCrawledPropSamples table keeps a list of **crawled properties** and some of the items that contain them.

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

```
TABLE MSSCrawledPropSamples (
    DocId                int NOT NULL,
    CrawledPropertyId    int NOT NULL,
    AccessHash           int NOT NULL
);
```

DocId: The unique identifier of an item.

CrawledPropertyId: The identifier of a crawled property.

AccessHash: The identifier for the **access URL** of the item.

2.2.6.5 MSSSessionDefinitions/MSSSessionDefinitionsAlt

The MSSSessionDefinitions and MSSSessionDefinitionsAlt tables are temporary repositories for definition metadata from items before it is merged into the **metadata index**.

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

```
TABLE MSSSessionDefinitions(  
    DocId                int NOT NULL,  
    Term                  nvarchar(40) NOT NULL,  
    Sentence              nvarchar(255) NOT NULL,  
    TermOffset            int NOT NULL,  
    TermLength            int NOT NULL  
);
```

DocId: The unique identifier of **an item**.

Term: The term from an item that is being defined.

Sentence: The sentence from which the term appears.

TermOffset: The offset (in characters) where the *Term* appears in the *Sentence*.

TermLength: The length of the *Term* in characters.

2.2.6.6 MSSSessionDocProps/MSSSessionDocPropsAlt

The MSSSessionDocProps and MSSSessionDocPropsAlt tables are temporary repositories for metadata from items before it is merged into the **metadata index**.

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

```
TABLE MSSSessionDocProps(  
    CatalogId            smallint NOT NULL,  
    DocId                int NOT NULL,  
    Pid                  int NOT NULL,  
    RowId                smallint NOT NULL,  
    llVal                bigint NULL,  
    strVal               nvarchar(64) NULL,  
    binVal               image NULL,  
    fltVal               float NULL  
);
```

CatalogId: **MUST be 1.**

DocId: The unique identifier of **an item**.

Pid: The unique identifier of **a managed property**.

RowId: The unique identifier for rows with the same DocId and Pid.

llVal: The numeric value of the managed property. It **MUST** be NULL if the managed property is not of type integer, Boolean or string. It holds a hash of the *strVal* column, if *strVal* is not NULL.

strVal: The string value of the managed property. If the value is greater than the size of *strVal*, then the overflow is stored in *binVal*. It **MUST** be NULL if the managed property is not of type string.

binVal: Binary metadata from items. It contains the overflow from the *strVal* if the managed property is of type string.

fltVal: The floating-point numeric value of the managed property. It MUST be NULL if the managed property is not of type floating-point.

2.2.6.7 MSSessionDocSdIds/MSSessionDocSdIdsAlt

The MSSessionDocSdIds table is a temporary repository for security-related metadata from **items** before it is merged into the **metadata index**.

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

```
TABLE MSSessionDocSdIds (
    DocId                int NOT NULL,
    Type                 smallint NOT NULL,
    Sdid                 int NOT NULL,
    HasPluggableSecurityTrimming bit NOT NULL
);
```

DocId: The unique identifier of an item.

Type: A number that MUST be 1 when the **search security descriptor** is in the format defined in [\[MS-DTYP\]](#), section 2.4.6. Otherwise, it MUST be 0.

Sdid: A unique number that identifies the search security descriptor.

HasPluggableSecurityTrimming: A number that MUST be 1 when the item uses **pluggable security authentication**. Otherwise, it MUST be 0.

2.2.6.8 MSSessionDocSignatures/MSSessionDocSignaturesAlt

The MSSessionDocSignatures and MSSessionDocSignaturesAlt tables are temporary repositories containing metadata that describes how an item has changed in the last crawl. These temporary repositories are later merged into the **metadata index**.

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

```
TABLE MSSessionDocSignatures (
    DocId                int NOT NULL,
    UrlSignature          bigint NULL,
    ContentSignature      bigint NULL,
    SchemaSignature       int NULL,
    CrawlTime             bigint NULL
);
```

DocId: The unique identifier of an item.

UrlSignature: An identifier of the access URL.

ContentSignature: An identifier of the data and metadata properties of the **item**.

SchemaSignature: An identifier of the set of metadata properties.

CrawlTime: The FILETIME in Coordinated Universal Time (UTC) of the point when the **item** was crawled.

2.2.6.9 MSSSessionDuplicateHashes/MSSSessionDuplicateHashesAlt

The MSSSessionDuplicateHashes and MSSSessionDuplicateHashesAlt tables are temporary repositories containing metadata that describes an item's uniqueness before it is merged into the **metadata index**.

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

```
TABLE MSSSessionDuplicateHashes(  
    DocId                int NOT NULL,  
    HashVal              bigint NOT NULL  
) ;
```

DocId: The unique identifier of an item.

HashVal: An identifier of an item's data and metadata.

2.2.6.10 MSSSessionExistingDocs/MSSSessionExistingDocsAlt

The MSSSessionExistingDocs and MSSSessionExistingDocsAlt tables are temporary repositories that list which of the items in the MSSSessionDocProps table (section [2.2.6.6](#)) are expected to exist already in the **metadata index**; those not listed are new.

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

```
TABLE MSSSessionExistingDocs(  
    DocId                int NOT NULL  
) ;
```

DocId: The unique identifier of an item.

2.2.6.11 MSSTranTempTable0

The MSSTranTempTable0 table implements the **link buffer** data structure, as specified in **Abstract Data Model** (section [3.1.1](#)).

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(1500) NOT NULL,  
    AccessHash           int NOT NULL,  
    CompactURL           nvarchar(40) NULL,  
    CompactHash          int NULL,  
    ParentCompactURL     nvarchar(40) NULL,  
    ParentCompactHash    int NULL,  
    DisplayURL           nvarchar(1500) NOT NULL,
```

```

DisplayHash          int NOT NULL,
Host                 nvarchar(300) NOT NULL,
hrResult            int NOT NULL,
AnchorText          nvarchar(512) NULL,
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,
FolderHighPriority  int NOT NULL,
ItemHighPriority     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
);

```

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

SourceDocID: An identifier of the item that generates the links.

DocID: The unique identifier of the item when *@ProjectID* is 1. Otherwise, it MUST be -1.

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

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

ProjectID: See Project Identifier in **Project Identifier** (section [2.2.2.1](#)) .

AccessURL: The item's access URL.

AccessHash: The identifier of the *@AccessURL* string.

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

CompactHash: The identifier of the *@CompactURL* string.

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

ParentCompactHash: The identifier of the *@ParentCompactURL* string.

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

DisplayHash: The identifier of the *@DisplayURL* string.

Host: The **host name** for the current item.

hrResult: MUST be 0x80040d07 if the item is an **excluded item**. Otherwise, it MUST be 0.

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

FirstLink: A number used to determine the order in which the **links** are discovered for an **item**. The first link has a value of 0, the next one has a value of 1, and so on.

TransactionType: See transaction type in **Transaction Type** (section [2.2.2.4](#)).

Scope: The scope of the transaction. See transaction scope in **Transaction Scope** (section [2.2.2.5](#)).

ItemType: The type of link. Its value MUST be in the following table.

Value	Description
1	The link is a start address .
2	Link discovered in the portal content crawl ; the crawl will follow this link.
4	Link discovered in the anchor text crawl .
6	Link discovered in the portal content crawl ; the crawl will not follow this link.

TransactionFlags: The transaction flags. See transaction flags in **Transaction Flags** (section [2.2.2.6](#)).

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.

UseChangeLog: MUST be either 1 or 0. If 1, the item belongs to a site that supports **incremental crawl** based on a **change log**. Otherwise, it MUST be 0.

IndexType: A number which specifies whether the item can be returned in search results. Its value MUST be in the following table.

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

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

FolderHighPriority: A number which MUST be either 0 or 1. If set to 1 then the item is a container of high priority items; otherwise 0. See below the explanation of *@ItemHighPriority*.

ItemHighPriority: If set to 1 then the item will be processed before other items. Otherwise, it MUST be 0.

SeqID: An identifier of the item in the current **crawl**.

LCID: **Language code identifier (LCID)**.

EndPathFlag: This value is a bit mask of flags. Its value MUST be in the following table:

Value	Description
0x0001	The access URL] ends with a slash.

Value	Description
0x0002	The display URL ends with a slash.

PropMD5: An identifier of the item metadata. In the **incremental crawl**, if the value of the parameter is different than the existing value, the **item** and any children will be re-crawled.

LastModifiedTime: A UTC **FILETIME** that indicates when the item was modified.

ProtocolSwitch: 1 if the parent item and the link have a different **protocol**. Otherwise, it MUST be 0.

2.2.7 XML Structures

None.

3 Protocol Details

3.1 MOSS Server Details

The Microsoft Office SharePoint Server role is described in this section.

3.1.1 Abstract Data Model

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

The index server uses multiple data structures to track its state. The following objects are defined here:

- Crawl status
- Crawl URL history
- Crawl queue
- Temporary storage for links discovered during the crawl
- Anchor text information

The following paragraphs define each of these data structures:

Crawl status: The crawl status structure maintains the lists of crawls in progress and recently completed crawls. Each crawl status has the following attributes:

- **crawl** unique identifier
- Project Identifier. For details see **Project Identifier** as specified in section [2.2.2.1](#).
- Crawl Type. For details see **crawl type** as specified in section [2.2.2.2](#).
- Crawl Status. For details see **crawl status** as specified in section [2.2.2.3](#).
- Crawl start time
- Crawl end time
- Identifier of the **content source** associated with the **crawl**.
- Identifiers of all **start addresses** that belong to the **content source** above.

Crawl URL history: The crawl URL history keeps the state of each URL processed by the index server. Each crawl URL has the following attributes:

- Unique item identifier
- Identifier of the **start address**
- Identifier of the **content source**
- Project identifier. See the explanation of the **project identifier** as specified in section [2.2.2.1](#).

- Identifier of the crawl that is going to process a item
- Identifier of the last crawl that is done with processing the item
- Access URL
- Display URL
- Transaction flags. See the explanation of the **transaction flags** as specified in section [2.2.2.6](#).

Crawl queue: The crawl queue contains items which need to be processed by the active crawls. Each item in the crawl queue has the following attributes:

- **Crawl** identifier
- Unique item identifier
- **Start address** identifier
- **Transaction type**. See the explanation of the **transaction type** as specified in section [2.2.2.4](#).
- **Transaction scope**. See the explanation of the transaction scope as specified in section [2.2.2.5](#).
- Identifier of the parent item

Link buffer: This data structure is used to temporarily store all links discovered during the crawl. The main attributes of a link are:

- Crawl identifier
- Source item identifier
- **Start address** identifier
- **Content source** identifier
- Project identifier. See the explanation of the **project identifier** as specified in section [2.2.2.1](#).
- **Access URL**
- **Display URL**
- Transaction flags. See the explanation of the **transaction flags** as specified in section [2.2.2.6](#).
- Transaction type. See the explanation of the **transaction type** as specified in section [2.2.2.4](#).
- Transaction scope. See the explanation of the **transaction scope** as specified in section [2.2.2.5](#).

Anchor text information: This data structure contains all the links discovered during the **crawl**. Each link has the following attributes:

- Link represented as a URL.
- Source Item identifier.
- Anchor text.
- Identifier of the anchor text language.

Anchor Text Completed Items: This data structure contains items marked as processed by the anchor text crawl. Each item has only one attribute: the document identifier.

In addition to the crawl data structures, two other sets of data structures are used by this protocol:

- Metadata-related functionality.

Search scope functionality.

The following subsections describe the metadata and search scope data structures.

3.1.1.1 Metadata Functionality

Most data structures used by metadata-related functionality are specified in [\[MS-SQLPADM\]](#), section 3.1.1.

However, certain metadata-related functionality results in configuration-related events using configuration properties. A configuration property is a property that stores a setting for the Office SharePoint Server Search (Osearch) service. There are three types of configuration properties:

- Regular **configuration property**
- **Big configuration property**
- **vector configuration property**

Regular configuration property

A regular configuration property stores a value of `sql_variant` type.

Big configuration property

A big configuration property stores a value that can't be stored using `sql_variant` type, for example, a data Binary large object (BLOB). Big configuration properties differ from regular configuration properties in the following two respects:

- Big configuration properties hold a value of the T-SQL type `image` instead of `sql_variant`.
- Values of big configuration properties are versioned. This means that each value has an integer version, and each time the value of the property is updated, the version number could either increase by 1 or remain unchanged, but it never decreases.

Vector configuration property

A vector configuration property has multiple values. These values are not ordered. All vector configuration properties have integer types. Other types are not supported.

3.1.1.2 Search Scopes Functionality

Search scope compilation uses multiple data objects to track different compilation states and compilation rules of the search scopes. The following objects are defined here:

- Compiled search scope set
- Compilation search scope set
- Deletion search scope set

The following paragraphs define each of these sets.

Compiled Search Scope Set

The compiled search scope set contains all the information about each search scope to be compiled as well as the rule information used by the search scope. The compiled search scope set has the following attributes:

- The search scope identifier
- The name of the search scope
- The state of the search scope compilation
- The filter behavior of the search scope rules
- The rule type of the search scope rules
- The URL type of the search scope rules
- The Managed Property identifier
- The search scope value string
- The last search scope compilation time
- The last search scope compilation identifier

Compilation Search Scope Set

The compilation search scope set contains all the information about each search scope to be compiled as well as the rule information used by the search scope. The compilation search scope set has the following attributes:

- The search scope identifier
- The name of the search scope
- The state of the search scope compilation
- The filter behavior of the search scope rules
- The rule type of the search scope rules
- The URL type of the search scope rules
- The Managed Property identifier
- The search scope value
- The search scope compilation scheduling type
- The search scope compilation time
- The search scope compilation identifier

Deletion Search Scope Set

The deletion search scope set contains information about each search scope to be deleted, including:

- The search scope identifier
- Version corresponding to last deletion for the search scope, which is a LastScopeChangeID data type as specified in [\[MS-SQLPADM\]](#), section 2.2.1.14).

3.1.2 Timers

An execution timeout timer on the protocol server governs the execution time for any requests. The amount of time is specified by a timeout value that is configured on the protocol server for all connections.

3.1.3 Initialization

A connection that uses the underlying protocol layers that are specified in Relationship to Other Protocols (section [1.4](#)) MUST be established before using this protocol as specified in [\[MS-TDS\]](#).

Listening endpoints are set up on the back end database server to handle inbound TDS requests.

Authentication of the TDS connection to the back-end database server MUST occur before this protocol can be used.

The data structures, stored procedures, and actual data are persisted by the back-end database server within databases, so any operations to initialize the state of the database MUST occur before the back-end database server can use this protocol. The data for the search index server MUST already exist within the back-end database server in a valid state.

3.1.4 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 specified in [\[MS-TDS\]](#), section 2.2.6.9.9 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. If the stored procedures are not provided the expected calling parameters or return the expected result set values, the system behavior is indeterminate.

For the sake of 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, since the ordinal position of any column with no defined name is expected by the protocol client. Such names are designated in the text using curly braces in the form *{name}*.

This section describes the following stored procedures:

Procedure Name	Description
proc_MSS_AddAndReturnCrawledProperty	Adds the crawled property to the metadata schema if it doesn't exist and returns its parameters if it does.

Procedure Name	Description
proc_MSS_AddCrawledPropertyCategoryWithDefaults	Adds a crawled property category to the metadata schema .
proc_MSS_CommitAnchorTextCrawl	Performs cleanup at the end of the anchor text crawl .
proc_MSS_Crawl	Gets or changes a given crawl's state.
proc_MSS_FlushTemp0	Processes all the links for a given item within a given crawl from the MSSTranTempTable0 table (section 2.2.6.11) to the crawl queue and crawl URL history structures, as specified in Abstract Data Model (section 3.1.1).
proc_MSS_GetBigConfigurationProperty	Gets a value of the specified big configuration property .
proc_MSS_GetCompiledScopeRules	Gets all the search scope rules of search scopes that are compiled.
proc_MSS_GetCompiledScopes	Gets information about all search scopes that are compiled.
proc_MSS_GetCompletedScopesCompilationID	Gets the search scope compilation identifier for search scopes that have compiled most recently.
proc_MSS_GetConfigurationProperty	Gets the value of a property of the search shared application object or the configuration property structure.
proc_MSS_GetCrawledPropertyUpdates	Lists crawled properties which have been added or updated in the metadata Schema after the given time.
proc_MSS_GetCrawledPropMappingUpdates	Lists mappings for crawled properties which have been added or updated in the metadata schema after the given time.
proc_MSS_GetCrawls	Informs the server that the search application is initializing, and retrieves a list of crawls in progress.
proc_MSS_GetDeletedScopesForCompilation	Gets all search scopes to be deleted.
proc_MSS_GetDocStatus	Gets the crawl status of the specified set of items .
proc_MSS_GetManagedProperties	Lists managed properties from the metadata schema which were added or modified on or after the given time.
proc_MSS_GetNextCrawlBatch	Retrieves a list of items from the crawl queue , as specified in Abstract Data Model (section 3.1.1), for a given crawl .
proc_MSS_GetSampleExtremes	Lists the crawled properties whose number of taken samples are either above

Procedure Name	Description
	or below the given parameters.
proc_MSS_GetSchemaHighLevelInfo	Retrieves last modified and last deleted timestamps from the metadata schema .
proc_MSS_GetSchemaMappings	Retrieves the list of mappings between crawled properties and managed properties .
proc_MSS_GetSchemaParameters	Retrieves a list of schema parameters from the metadata schema .
proc_MSS_GetScopeRulesForCompilation	Gets the search scope rules for a search scope 's current search scope compilation .
proc_MSS_GetScopesForCompilation	Gets the search scopes that are involved for the current search scope compilation .
proc_MSS_GetSDID	Retrieves a search security descriptor from the metadata schema .
proc_MSS_InsertFromSession	Flushes session table data (see tables in section 2.2.5) to the metadata index .
proc_MSS_OnDocDelete	Deletes the metadata for an item from the metadata index .
proc_MSS_OnEndCrawl	Performs database-related maintenance at the end of a crawl .
proc_MSS_OnStartCrawl	Performs database related maintenance at the start of a crawl .
proc_MSS_PrepareAnchorTextCrawl	Prepares the anchor text crawl and gets the oldest portal content crawl identifier that has unprocessed anchor text information.
proc_MSS_ProcessCommitted	Sets each item's status as completed and removes it from the crawl queue , as specified in Abstract Data Model (section 3.1.1).
proc_MSS_PushSD	Stores a new search security descriptor .
proc_MSS_ReportScopesCompilationBegin	Notifies the search scopes system that search scope compilation has begun.
proc_MSS_ReportScopesCompilationEnd	Notifies the search scopes system that that search scopes compilation has completed.
proc_MSS_ResetCatalog	Clears all customer data from the metadata index .
proc_MSS_SetBigConfigurationProperty	Updates the value of the specified big configuration property or inserts it if it doesn't exist.

Procedure Name	Description
proc_MSS_SetConfigurationProperty	Sets the value of the specified configuration property or inserts it if it doesn't exist.
proc_MSS_SetCrawledPropertyIsSampleCacheFull	Updates the <i>IsSampleCacheFull</i> flag in the sample crawled properties set for the specified crawled property .
proc_MSS_ShareScopesCompilationInfo	Reports information about search scope compilation .
proc_MSS_TruncateCleanupTable	Clears structures used to adjust the <i>IsSampleCacheFull</i> property in the crawled property set (as specified in [MS-SQLPADM] , section 2.2.1.x) of the metadata index .

3.1.4.1 proc_MSS_AddAndReturnCrawledProperty

The **proc_MSS_AddAndReturnCrawledProperty** stored procedure is called to add the **crawled property** to the **metadata schema** if it doesn't exist and to return its parameters if it does. If the *FullTextQueryable* or *Retrievable* flags are set to 1, then a **managed property** MUST also be created if it doesn't already exist in the **managed property set** as defined in [\[MS-SQLPADM\]](#), section 2.2.1.x, and a mapping between the **crawled property** and **managed property** MUST be created if it doesn't already exist in the **mapping set** as defined in section [\[MS-SQLPADM\]](#), 2.2.1.x.

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

```

PROCEDURE proc_MSS_AddAndReturnCrawledProperty(
    @Propset                uniqueidentifier,
    @PropertyName            nvarchar(440),
    @PropertyNameIsEnum      bit,
    @VariantType             int,
    @FriendlyName            nvarchar(64),
    @ManagedType            int,
    @crawledPropId           int OUTPUT,
    @pid                    int OUTPUT,
    @crawledPropExists       bit OUTPUT,
    @managedPropExists       bit OUTPUT,
    @IsSampleCacheFull       bit OUTPUT,
    @IsMappedToContent       bit OUTPUT,
    @FullTextQueryable       bit OUTPUT,
    @Retrievable             bit OUTPUT,
    @HasMultipleValues       bit OUTPUT,
    @MappingOrder            int OUTPUT
);

```

@Propset: A **GUID** which identifies a crawled property set identifier in the list of crawled property categories.

@PropertyName: The name of the crawled property.

@PropertyNameIsEnum: If set to 1, an enumeration which is a number that was converted to a string Otherwise, it MUST be 0.

@VariantType: The **variant type** for the crawled property.

@FriendlyName: A string that uniquely identifies the managed property.

@ManagedType: The type of the managed property as defined in section [2.2.2.14](#).

@CrawledPropId: Upon return from this stored procedure, this parameter MUST be set to a unique identifier for the crawled property.

@pid: Upon return from this stored procedure, this parameter MUST be set to a 32-bit integer that uniquely identifies a managed property if the *FullTextQueryable* or *Retrievable* flags are set to 1, OTHERWISE this parameter MUST be set to -1.

@crawledPropExists: Upon return from this stored procedure, this parameter MUST be set to 1 if the crawled property already exists. It MUST be set to 0 if the crawled property has been added.

@managedPropExists: Upon return from this stored procedure, this parameter MUST be set to 1 if the managed property already exists. It MUST be set to 0 if the managed property has been added.

@IsSampleCacheFull: Upon return from this stored procedure, this parameter MUST be set to 1 if the sample crawled properties set as defined in [MS-SQLPADM], section 2.2.1.x is complete. Otherwise, it MUST be set to 0.

@IsMappedToContent: Upon return from this stored procedure, this parameter MUST be set to 1 if the variant type is a string, and data from this crawled property is put in the **full-text index catalog**. Otherwise, it MUST be set to 0.

@FullTextQueryable: Upon return from this stored procedure, this parameter MUST be set to 1 if the data for the managed property is kept in the full-text index catalog. Otherwise, it MUST be set to 0.

@Retrievable: Upon return from this stored procedure, this parameter MUST be set to 1 if the data for the **managed property** is kept in the **metadata index**. Otherwise, it MUST be set to 0.

@HasMultipleValues: Upon return from this stored procedure, this parameter MUST be set to 1 if the value of the managed property can contain multiple values. Otherwise, it MUST be set to 0.

@MappingOrder: Upon return from this stored procedure, this parameter MUST be set to an integer representing the relative priority of the crawled property mappings to a managed property.

Return Code Values: An integer which MUST be 0.

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

3.1.4.2 **proc_MSS_AddCrawledPropertyCategoryWithDefaults**

The **proc_MSS_AddCrawledPropertyCategoryWithDefaults** stored procedure is called to add a **crawled property category** to the **metadata schema**. This procedure updates the global last modified timestamp in the **crawled property set identifier** with the local time of the server.

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

```
PROCEDURE proc_MSS_AddCrawledPropertyCategoryWithDefaults (  
    @CategoryName          nvarchar(64) OUTPUT,  
    @DiscoverNewProperties  bit OUTPUT,  
    @MapToContents         bit OUTPUT,
```

```

        @FullTextQueryable          bit OUTPUT,
        @Retrievable                bit OUTPUT,
        @Propset                    uniqueidentifier
    );

```

@CategoryName: The name of the crawled property category.

@DiscoverNewProperties: Upon return from this stored procedure, this parameter **MUST be set** to 1 if the crawled properties within this crawled property category are added to the crawled properties set automatically. Otherwise, it **MUST** be set to 0.

@MapToContents: Upon return from this stored procedure, this parameter **MUST be set** to 1 if string data from a newly discovered crawled property within this crawled property category is put in the full-text index catalog. Otherwise, it **MUST** be set to 0.

@FullTextQueryable: Upon return from this stored procedure, this parameter **MUST be set to 1 if string** data from newly discovered Crawled Properties within this crawled property category is mapped to a new managed property which will be put in the full-text index catalog. Otherwise, it **MUST** be set to 0.

@Retrievable: Upon return from this stored procedure, this parameter **MUST be set to 1 if string** data from newly discovered Crawled Properties within this crawled property category is mapped to a new managed property which will be put in the metadata index. Otherwise, it **MUST** be set to 0.

@Propset: A GUID which identifies the crawled **property set** identifier of the **crawled property category**.

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

Result Sets: SHOULD NOT [<3>](#) return any result set. The protocol client **MUST** ignore any result sets returned by this stored procedure.

3.1.4.3 proc_MSS_CommitAnchorTextCrawl

The **proc_MSS_CommitAnchorTextCrawl** stored procedure is called by the **index server** after an **anchor text crawl** is completed to perform cleanup at the end of the **crawl**.

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

```

PROCEDURE proc_MSS_CommitAnchorTextCrawl (
    @CrawlID                int
);

```

@CrawlID: The identifier of the **portal content crawl** that generated the changes in the **anchor text** information.

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

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

3.1.4.4 proc_MSS_Crawl

The **proc_MSS_Crawl** stored procedure is called by the **index server** to retrieve or change a given **crawl's** state.

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

```
PROCEDURE proc_MSS_Crawl (
    @ProjectID          int,
    @CrawlStage         int,
    @CrawlType          int,
    @CrawlID            int,
    @ContentSourceID    int,
    @StartAddressID     int,
    @AcquiredCrawlID    int OUTPUT,
    @CrawlStatus        int OUTPUT
);
```

@ProjectID: A project identifier as specified in section [2.2.2.1](#).

@CrawlStage: The action the stored procedure executes. Its value MUST be in the following table.

Value	Description
1	Acquire a new crawl identifier. This is the first step in the Crawl Data Flow Diagram (Figure 1).
2	Add a start address to the list of start addresses to be crawled. This is the second step in the Crawl Data Flow Diagram (figure 1).
3	This is the Attempt Start Crawl step in the Crawl Data Flow Diagram (figure 1). A crawl can start if none of the content sources are involved in another crawl. If this condition is met then the stored procedure returns <i>@CrawlStatus</i> =4 which indicates that the crawl started; otherwise it returns <i>@CrawlStatus</i> =5 which indicates that the crawl is aborted.
6	The index server calls the stored procedure with this parameter to abort a crawl in lieu of making the call in the Attempt Start Crawl step of the Crawl Data Flow diagrams (figures 1-5).
7	Checks if the crawl is done. If the crawl is done the stored procedure returns <i>@CrawlStatus</i> =11; otherwise it returns <i>@CrawlStatus</i> =4. This is the Get Crawl Status/Finish crawl action in the Crawl Data Flow Diagram (figure 1).
9	Pauses the crawl . The server MUST set the crawl status , as specified in Abstract Data Model (section 3.1.1), to 9 in the crawl URL history , as specified in Abstract Data Model (section 3.1.1), which indicates that the crawl is paused.
10	Resumes the crawl . The server MUST set the crawl status , as specified in Abstract Data Model (section 3.1.1), to 4 in the crawl URL history , as specified in Abstract Data Model (section 3.1.1), which indicates that the crawl is in progress.
12	Stops the crawl . The server MUST set the crawl status , as specified in Abstract Data Model (section 3.1.1) to 12 in the crawl URL history , as specified in Abstract Data Model (section 3.1.1), which indicates that the crawl is stopped.
14	Cleans up crawl data when a search application is reset.
15	Starts a delete crawl . This is the second step in the Delete Crawl Data Flow Diagram (figure 3).
16	A content source has been deleted. A subsequent call will be made to initiate a delete crawl.
17	A start address is deleted. A subsequent call will be made to initiate a delete crawl.

@CrawlType: Specifies the type of the **crawl**. For details see **crawl type** in **Abstract Data Model** (section [3.1.1](#)).

@CrawlID: When *@CrawlStage* equals 1, 14, 16 or 17, **this parameter MUST be ignored by the server. Otherwise, it MUST be set to a** unique identifier for the crawl.

@ContentSourceID: When *@CrawlStage* equals 2 or 16, this parameter is a unique identifier for the content source. Otherwise **this parameter MUST be ignored by the server.**

@StartAddressID: When *@CrawlStage* equals 2 or 17, this parameter is a unique identifier for the start address. Otherwise, **this parameter MUST be ignored by the server.**

@AcquiredCrawlID: **Upon return from this stored procedure, this parameter MUST be set** to a unique identifier for the new crawl when *@CrawlStage* equals 1. Otherwise **this parameter MUST be ignored by the client.**

@CrawlStatus: **Upon return from this stored procedure, this parameter MUST be set** to the crawl status, as specified in **Abstract Data Model** (section [3.1.1](#)) when *@CrawlStage* equals 3 or 7. Otherwise **this parameter MUST be ignored by the client.**

Return Code Values: An integer which MUST be listed in the following table.

Value	Description
0	The stored procedure failed. No changes were persisted.
1	Successful execution.

Result Sets: MUST NOT return any result set.

3.1.4.5 `proc_MSS_FlushTemp0`

The **`proc_MSS_FlushTemp0`** stored procedure is called to process all the links for a given item within a given **crawl** from the **`MSSTranTempTable0`** table (section [2.2.6.11](#)) to the **crawl queue** and **crawl URL history** structures, as specified in **Abstract Data Model** (section [3.1.1](#)). This stored procedure implements the Process Links step in the Crawl Data Flow Diagram (figure 1).

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

```
PROCEDURE proc_MSS_FlushTemp0 (
    @FullIncrementalInterval    int,
    @DeleteOnErrorInterval     int,
    @DocID                     int,
    @MaxDocId                  int,
    @CrawlID                   int,
    @AnchorsLimit              int,
    @LicenseExceeded           bit OUTPUT
);
```

@FullIncrementalInterval: See **Full Incremental Interval** as specified in section [2.2.2.7](#).

@DeleteOnErrorInterval: See **Delete On Error Interval** as specified in section [2.2.2.8](#).

@DocID: A unique identifier of the **item**.

@MaxDocID: An integer which MUST be -1 if there is no limit of the number of items in the search application; otherwise, if there is a limit, the total number of items MUST be restricted to be less than this limit.[<4>](#)

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

@AnchorsLimit: Maximum number of anchor text links persisted for this **item**.

@LicenseExceeded: Upon return from this stored procedure, this parameter MUST be set to **the value of 1** if the number of items exceeds the limit specified by @MaxDocID, after the links are processed; otherwise 0. [<5>](#)

Return Code Values: An integer which MUST be listed in the following table.

Value	Description
0	The stored procedure failed. No changes were persisted.
1	Successful execution.

Result Sets: MUST NOT return any result set.

3.1.4.6 proc_MSS_GetBigConfigurationProperty

The **proc_MSS_GetBigConfigurationProperty** stored procedure is called to retrieve a value of the specified **big configuration property**.

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

```
PROCEDURE proc_MSS_GetBigConfigurationProperty(  
    @Name          nvarchar(64)  
) ;
```

@Name: The name of the configuration property.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.6.1 Big Configuration Property Result Set

The Big Configuration Property result set returns value of the property. The result set MUST contain one row if a property with the specified name is found; otherwise the stored procedure MUST return zero rows.

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

```
{PropertySize}      int,  
BigValue            image;
```

{PropertySize}: The size of the property value in bytes.

BigValue: The value of the property.

3.1.4.7 proc_MSS_GetCompiledScopeRules

The **proc_MSS_GetCompiledScopeRules** stored procedure is called to retrieve all the **search scope rules** of **search scopes** that are compiled.

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

```
PROCEDURE proc_MSS_GetCompiledScopeRules();
```

Return Code Values: An number which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.7.1 Compiled Scope Rules Result Set

The Compiled Scope Rules result set returns a list of **search scope rules**. The result set MUST contain zero or more rows, each corresponding to a single compiled search scope rule. The result set is ordered by **ScopeID**.

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

ScopeID	int,
FilterBehavior	smallint,
RuleType	smallint,
UrlRuleType	smallint,
PropertyID	int,
UserValueString	nvarchar(2048);

ScopeID: The unique identifier for the search scope rule.

FilterBehavior: The filter behavior of the **search scope rule**. The value MUST be a **FilterBehavior** Data Type, as specified in section [2.2.2.10](#).

RuleType: The rule type of the **search scope rule**. The value MUST be a **RuleType** Data Type as specified in section [2.2.2.11](#).

UrlRuleType: The URL type of the **search scope rule**. The value MUST be an **UrlRuleType** Data Type as specified in section [2.2.2.12](#).

PropertyID: The unique identifier for the **managed property** used by the **search scope rule**.

UserValueString: The value used for filtering by the **search scope rule**.

3.1.4.8 proc_MSS_GetCompiledScopes

The **proc_MSS_GetCompiledScopes** stored procedure is called to retrieve the information about all **search scopes** that have undergone search scope compilation.

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

```
PROCEDURE proc_MSS_GetCompiledScopes();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.8.1 Scopes Result Set

See **Scopes Result Set** as specified in section [2.2.5.1](#).

3.1.4.9 proc_MSS_GetCompletedScopesCompilationID

The **proc_MSS_GetCompletedScopesCompilationID** stored procedure is called to retrieve the **search scope compilation identifier** for **search scopes** that have compiled most recently.

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

```
PROCEDURE proc_MSS_GetCompletedScopesCompilationID(  
    @CompilationID          int OUTPUT  
);
```

CompilationID: Upon return from this stored procedure, this parameter MUST be set to the latest **search scope compilation identifier**.

Return Code Values: An integer which MUST be 0.

Result Set: MUST NOT return any result set.

3.1.4.10 proc_MSS_GetConfigurationProperty

The **proc_MSS_GetConfigurationProperty** stored procedure is called to retrieve the value of a property of the **search shared application object** or the **configuration property** structure. If the property is found on the **search shared application object** this value MUST be returned; if the property is found in the **configuration property** structure, this value MUST be returned. Otherwise **NULL MUST be returned**.

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

```
PROCEDURE proc_MSS_GetConfigurationProperty(  
    @Name          nvarchar(64),  
    @Value          sql_variant OUTPUT  
);
```

@Name: The name of the property.

@Value: Upon return from this stored procedure, this parameter MUST be set to **the value of the property**. **Return Code Values:** An integer which MUST be 0.

Result Sets: SHOULD NOT [<6>](#) return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.11 proc_MSS_GetCrawledPropertyUpdates

The **proc_MSS_GetCrawledPropertyUpdates** stored procedure is called to list **crawled properties** which have been added or updated in the **metadata schema** after the *@UpDate* time.

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

```
PROCEDURE proc_MSS_GetCrawledPropertyUpdates(  
    @UpDate          datetime  
);
```

@UpDate: Specifies the earliest per-item "last modified" 8 byte **datetime** for including a **crawled property** in the result set.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.11.1 GetCrawledPropertyUpdates Result Set

The GetCrawledPropertyUpdates result set returns the list of **crawled properties** added or updated in the **metadata schema** after the *@UpDate* time. The result set MUST contain zero or more rows.

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

Propset	uniqueidentifier,
PropertyName	nvarchar(440),
PropertyNameIsEnum	bit,
IsMappedToContent	bit,
IsSampleCacheFull	bit,
VariantType	int,
CrawledPropertyId	int,
URI	nvarchar(2048),
LastModified	datetime;

Propset: A GUID which identifies a crawled property set identifier in the list of crawled property categories.

PropertyName: A string that identifies the crawled property.

PropertyNameIsEnum: 1 if the string name was converted from an integer. Otherwise, it MUST be 0.

IsMappedToContent: If set to 1 and the *VariantType* is a string, data from this crawled property is put in the full-text index catalog. Otherwise, it MUST be 0.

IsSampleCacheFull: 1 if the sample crawled properties set is complete. Otherwise, it MUST be 0.

VariantType: A 32-bit integer that holds the **variant** type for the Crawled Property.

CrawledPropertyId: The unique identifier for the crawled property.

URI: A URI (Uniform Resource Identifier) associated with the **crawled property**.

LastModified: This parameter MUST be ignored by the protocol client.

3.1.4.12 proc_MSS_GetCrawledPropMappingUpdates

The **proc_MSS_GetCrawledPropMappingUpdates** stored procedure is called to list mappings for a **crawled property** which have been added or updated in the **metadata schema** after the *@UpDate* time.

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

```
PROCEDURE proc_MSS_GetCrawledPropMappingUpdates (
    @UpDate          datetime
);
```

@UpDate: The earliest per-item "last modified" 8-byte **datetime** for including a **crawled property** mapping in the result set.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.12.1 GetCrawledPropMappingUpdates Result Set

The GetCrawledPropMappingUpdates result set returns the list of **crawled property** mappings which have been added or updated in the **metadata schema** after the *@UpDate* time. The result set MUST contain zero or more rows.

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

```
pid                int,  
mappingorder       int,  
crawledPropertyid int;
```

pid: The unique identifier for the **managed property**.

mappingorder: An integer representing the relative priority of the crawled property mappings to a managed property.

crawledPropertyid: The unique identifier for the crawled property.

3.1.4.13 proc_MSS_GetCrawls

The **proc_MSS_GetCrawls** stored procedure is called when the **search application** is initialized to inform the server that the **search application** is initializing, and to retrieve a list of **crawls** in progress.

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

```
PROCEDURE proc_MSS_GetCrawls(  
    @CatalogID      int  
) ;
```

@CatalogID: The unique project identifier. For more information see **project identifier** as specified in section [2.2.2.1](#).

Return Code Values: An integer which MUST be listed in the following table.

Value	Description
0	The stored procedure failed. No changes were persisted.
1	Successful execution.

Result Sets: MUST return the following result set:

3.1.4.13.1 GetCrawls Result Set

The GetCrawls result set returns a list of **crawls** that are in progress. There is a row for each **crawl** per **start address** and there is no guaranteed order. The result set MUST contain zero or more rows.

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

CrawlID	int,
CrawlType	int,
Status	int,
ContentSourceID	int,
StartAddressID	int

CrawlID: The unique identifier for the **crawl**.

CrawlType: See crawl type in **Abstract Data Model** (section [3.1.1](#)).

Status: MUST be 4 when the crawl is in progress. Otherwise, it MUST be 9, indicating that the crawl is paused.

ContentSourceID: The unique identifier for the **content source** being crawled.

StartAddressID: The unique identifier for the **start address**.

3.1.4.14 proc_MSS_GetDeletedScopesForCompilation

The **proc_MSS_GetDeletedScopesForCompilation** stored procedure is called to retrieve all **search scopes** to be deleted.

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

```
PROCEDURE proc_MSS_GetDeletedScopesForCompilation();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.14.1 Deleted Search Scopes Result Set

The Deleted Search Scopes result set returns a list of all **search scopes** to be deleted. The result set MUST contain zero or more rows, each corresponding to a single search scope.

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

ScopeID	int;
---------	------

ScopeID: The unique identifier for the search scope to be deleted.

3.1.4.15 proc_MSS_GetDocStatus

The **proc_MSS_GetDocStatus** stored procedure is called to retrieve crawl status of the specified set of items.

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

```
PROCEDURE proc_MSS_GetDocStatus (
    @DisplayHashes      nvarchar(2048)
);
```

@ DisplayHashes: A comma-separated identifier list of the set of items for which status will be returned.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.15.1 Document Status Result Set

The Document Status result set returns the list of **authority pages**. The result set MUST contain zero or more rows, each corresponding to an item. The result is ordered (descending) by the **DisplayURL** index type and *DocID*, as specified in section [2.2.2.9](#),

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

```
DocId          int,
ErrorId        int,
DisplayURL     nvarchar(1500);
```

DocId: The unique identifier for the item.

ErrorId: A unique identifier for the error; or 0 if the item was crawled successfully.

DisplayURL: The display URL of the item.

3.1.4.16 proc_MSS_GetManagedProperties

The **proc_MSS_GetManagedProperties** stored procedure is called to list **managed properties** from the **metadata schema which were** added or modified on or after the *@Idate* time.

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

```
PROCEDURE proc_MSS_GetManagedProperties (
    @Idate        datetime
);
```

@Idate: The earliest per-item "last modified" 8-byte **datetime** for including a **managed property** in the result set.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.16.1 GetManagedProperties Result Set

The GetManagedProperties result set returns the list of **managed properties** added or updated on or after *@Idate*. The result set MUST contain zero or more rows.

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

PID	int,
FriendlyName	nvarchar(64),
ManagedType	int,
FullTextQueriable	bit,
Retrievable	bit,
Scoped	bit,
RespectPriority	bit,
RemoveDuplicates	bit,
NoDelete	bit,
NoMap	bit,
Hidden	bit,
HasMultipleValues	bit,
NoWordBreaker	bit,
NameNormalized	bit,
IncludeInMD5	bit,
openbit1	bit,
openbit2	bit,
Mapped	bit,
QueryIndependentRank	bit,
WordBreakerOverride	int,
Weight	float,
LengthNormalization	float,
LastModified	datetime;

PID: The unique identifier for the managed property.

FriendlyName: A string that uniquely identifies the managed property.

ManagedType: The type of the managed property is specified in section [2.2.2.14](#).

FullTextQueriable: A bit which MUST be 1 if the data for the managed property is kept in the full-text index catalog. Otherwise, it MUST be 0.

Retrievable: A bit which MUST be 1 if the data for the managed property is kept in the metadata index. Otherwise, it MUST be 0.

Scoped: A bit which MUST be 1 if the data for the managed property is kept in the search scope index. Otherwise, it MUST be 0.

RespectPriority: A bit which MUST be 1 if only data with highest priority (based on **mapping order**) from the crawled properties mapped to this managed property is used. It MUST be 0 if values from all crawled properties mapped to this managed property are used.

RemoveDuplicates: This parameter MUST be ignored by the client.

NoDelete: This parameter MUST be ignored by the client.

NoMap: This parameter MUST be ignored by the client.

Hidden: This parameter MUST be ignored by the client.

HasMultipleValues: A bit which MUST be 1 if the value of the managed property can contain multiple values. Otherwise, it MUST be 0.

NoWordBreaker: This parameter MUST be ignored by the client.

NameNormalized: A bit which MUST be 1 if the values of this managed property are to be normalized by the index server. Otherwise, it MUST be 0.

IncludeInMD5: A bit which MUST be 1 if values mapped to this managed property are used to determine if the item has changed. Otherwise, it MUST be 0.

openbit1: This parameter MUST be ignored by the client.

openbit2: This parameter MUST be ignored by the client.

Mapped: A bit which MUST be 1 when the property is a URL that is manipulated by **alternate access mappings**. Otherwise, it MUST be 0.

QueryIndependentRank: A bit which MUST be 1 when the property participates in **query independent rank**. Otherwise, it MUST be 0.

WordBreakerOverride: This parameter MUST be ignored by the client.

Weight: A decimal value used to adjust **property oriented rank**.

LengthNormalization: A decimal value used to adjust property oriented rank.

LastModified: This parameter MUST be ignored by the client.

3.1.4.17 **proc_MSS_GetNextCrawlBatch**

The **proc_MSS_GetNextCrawlBatch** stored procedure is called to retrieve a list of items from the **crawl queue**, as specified in **Abstract Data Model** (section [3.1.1](#)), for a given **crawl**.

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

```
PROCEDURE proc_MSS_GetNextCrawlBatch(  
    @ProjectID                int,  
    @CrawlID                  int,  
    @BatchSize                 int,  
    @FolderHighPriorityTransactions int  
);
```

@ProjectID: See **project identifier** as specified in section [2.2.2.1](#).

@CrawlID: A unique identifier for the **crawl**.

@BatchSize: The maximum number of rows that can be returned in the result set.

@FolderHighPriorityTransactions: The maximum number of **high priority folders** that can be returned.

Return Code Values: An integer which MUST be listed in the following table.

Value	Description
0	The stored procedure failed. No changes were persisted.
1	Successful execution.

Result Sets: MUST return the following result set:

3.1.4.17.1 GetNextCrawlBatch Result Set

The GetNextCrawlBatch result set returns a list of items to be crawled. The result set **MUST** contain zero or more rows and **MUST** be equal to or less than *@BatchSize*. The result set is sorted; the items that have higher priority to be indexed are returned first.

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

CrawlID	int,
SourceDocID	int,
DocID	int,
DisplayURL	nvarchar(1500),
AccessURL	nvarchar(1500),
CompactURL	nvarchar(40),
EndPathFlag	int,
StartAddressID	int,
HostDepth	int,
EnumerationDepth	int,
TransactionFlags	int,
MD5	int,
PropMD5	int,
UseChangeLog	int,
IndexType	int,
LastModifiedTime	bigint,
FolderDelCount	int,
Reserved1	bigint,
Reserved2	bigint,
Reserved3	int,
Reserved4	int,
TransactionType	int,
LCID	int,
ItemType	int,
FolderHighPriority	int,
ItemHighPriority	int,
SeqID	bigint,
ChangeLogCookie	int,
ChangeLogBatchID	int,
Scope	int,
DocPropsMD5	bigint,
Retry	int;

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

SourceDocID: The identifier of the parent item.

DocID: The unique identifier of the item.

DisplayURL: The **display URL** of the item.

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

CompactURL: The unique string ID that identifies the item in the list of URLs crawled by the **index server**. It is an abbreviated form of the **access URL**.

EndPathFlag: This value is a bit mask of flags. The flags are documented in the table below:

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

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

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.

TransactionFlags: The **transaction** flags. See **transaction flags** as specified in section [2.2.2.6](#).

MD5: The identifier of the item content.

PropMD5: The identifier of the item metadata. In the **incremental crawl**, if the value of the parameter is different from the existing value, the item and any children will be re-crawled.

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

IndexType: Specifies whether the item can be returned in search results. Its value MUST be in the following table.

Value	Description
0	The item cannot be returned in search results
1	The item can be returned in search results

LastModifiedTime: The UTC **FILETIME** that indicates when the item was modified.

FolderDelCount: This parameter MUST be ignored by the client.

Reserved1: This parameter MUST be ignored by the client.

Reserved2: This parameter MUST be ignored by the client.

Reserved3: This parameter MUST be ignored by the client.

Reserved4: This parameter MUST be ignored by the client.

TransactionType: See **transaction type** as specified in section [2.2.2.4](#).

LCID: The language code identifier (LCID).

ItemType: This parameter MUST be ignored by the client.

FolderHighPriority: An integer which MUST be 1 when the item is a container of high priority items. Otherwise, it MUST be 0. See the following explanation of *@ItemHighPriority*.

ItemHighPriority: An integer which MUST be 1 when the item will be processed before other items. Otherwise, it MUST be 0.

SeqID: The identifier of the item in the current **crawl**.

ChangeLogCookie: A token that represents the last change that was retrieved from the **change log**.

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

Scope: The scope of the **transaction**. See **transaction scope** as specified in section [2.2.2.5](#).

DocPropsMD5: The identifier of the item metadata.

Retry: This parameter **MUST** be ignored by the client.

3.1.4.18 **proc_MSS_GetSampleExtremes**

The **proc_MSS_GetSampleExtremes** stored procedure is called to list the **crawled properties** whose number of samples that have been taken are either above the *@cHighLimit* or below the *@cLowLimit* parameter.

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

```
PROCEDURE proc_MSS_GetSampleExtremes(  
    @cHighLimit      int,  
    @cLowLimit       int  
);
```

@cHighLimit: The upper limit of crawled properties to be returned.

@cLowLimit: The lower limit of crawled properties to be returned.

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

Result Sets: **MUST** return the following result set:

3.1.4.18.1 **GetSampleExtremes Result Set**

The **GetSampleExtremes** result set returns the list of **crawled property** identifiers and sample counts for which the **crawled property** sample count is either above the *@cHighLimit* or below the *@cLowLimit* parameter. The result set **MUST** contain zero or more rows.

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

```
CrawledPropertyId      int,  
CPCount                int;
```

CrawledPropertyId: A unique identifier of the crawled property.

CPCount: The number of crawled property samples for the **CrawledPropertyId**.

3.1.4.19 **proc_MSS_GetSchemaHighLevelInfo**

The **proc_MSS_GetSchemaHighLevelInfo** stored procedure is called to retrieve last modified and last deleted timestamps from the **metadata schema**.

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

```
PROCEDURE proc_MSS_GetSchemaHighLevelInfo();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.19.1 GetSchemaHighLevelInfo Result Set

The GetSchemaHighLevelInfo result set returns global timestamps reflecting changes made to the **metadata schema**. The result set MUST contain zero rows or one row.

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

LastCatChange	datetime,
LastCPDelete	datetime,
LastCPAddsBenignModified	datetime,
LastURIAdds	datetime,
LastURIModifiedDeleted	datetime,
LastManagedProp	datetime,
LastGlobalProps	datetime,
LastManagedPropDeleted	datetime,
LastSmpDelete	datetime,
LastAliasAdd	datetime,
LastAliasOther	datetime;

LastCatChange: A timestamp that is updated with the local time of the server whenever a crawled property category is added, modified, or deleted.

LastCPDelete: A timestamp that is updated with the local time of the server whenever a crawled property is deleted.

LastCPAddsBenignModified: A timestamp that is updated with the local time of the server whenever a crawled property is added or modified or when a mapping from a crawled property to a managed property is added, changed or deleted.

LastURIAdds: This parameter MUST be ignored by the client.

LastURIModifiedDeleted: This parameter MUST be ignored by the client.

LastManagedProp: A timestamp that is updated with the local time of the server whenever a managed property is added or modified.

LastGlobalProps: A timestamp that is updated with the local time of the server whenever a schema parameter is added or modified.

LastManagedPropDeleted: A timestamp that is updated with the local time of the server whenever a crawled property or managed property is deleted.

LastSmpDelete: This parameter MUST be ignored by the client.

LastAliasAdd: This parameter MUST be ignored by the client.

LastAliasOther: This parameter MUST be ignored by the client.

3.1.4.20 **proc_MSS_GetSchemaMappings**

The **proc_MSS_GetSchemaMappings** stored procedure is called to retrieve the list of mappings between **crawled properties** and **managed properties**.

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

```
PROCEDURE proc_MSS_GetSchemaMappings();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.20.1 **Schema Mappings Result Set**

The Schema Mappings result set contains unsorted list of mappings between **crawled properties** and **managed properties**. The result set MUST contain zero or more rows, each corresponding to a single mapping.

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

```
pid                int,  
MappingOrder       int,  
CrawledPropertyId  uniqueidentifier;
```

pid: A 32-bit integer that uniquely identifies the **managed property**.

MappingOrder: An integer representing the relative priority of the crawled property mappings to a managed property.

CrawledPropertyId: A **GUID** that uniquely identifies the **crawled property**.

3.1.4.21 **proc_MSS_GetSchemaParameters**

The **proc_MSS_GetSchemaParameters** stored procedure is called to retrieve a list of **schema** parameters from the **metadata schema**.

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

```
PROCEDURE proc_MSS_GetSchemaParameters();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.21.1 **Schema Parameters Result Set**

The Schema Parameters result set returns the list of **schema** parameters from the **metadata schema**. The result set MUST contain zero or more rows, each corresponding to a single parameter.

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

```
ParamName          nvarchar(40),
```


IsString	bit,
strValue	nvarchar(256),
fltValue	float;

ParamName: The name of the **schema** parameter.

IsString: If set to 1, the *strValue* field MUST be set to the value of the **schema** parameter. Otherwise, it MUST be set to 0, and the value of the schema parameter MUST be returned in the *fltValue* field.

strValue: The string value of the parameter. This field MUST be ignored when *IsString* is set to 0.

fltValue: The floating-point value of the parameter. This field MUST be ignored when *IsString* is set to 1.

3.1.4.22 proc_MSS_GetScopeRulesForCompilation

The **proc_MSS_GetScopeRulesForCompilation** stored procedure is called to retrieve the **search scope rules** for a **search scope's** current **search scope compilation**.

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

```
PROCEDURE proc_MSS_GetScopeRulesForCompilation();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.22.1 Pending Scope Rules Result Set

The Pending Scope Rules result set returns a list of all **search scope rules** for current compilation. The result set MUST contain zero or more rows, each corresponding to a single search scope rule.

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

ScopeID	int,
FilterBehavior	smallint,
RuleType	smallint,
UrlRuleType	smallint,
PropertyID	int,
UserValueString	nvarchar(2048);

ScopeID: An integer that uniquely identifies the search scope to which the search scope rule belongs.

FilterBehavior: The behavior of the search scope rule. The value MUST be a **FilterBehavior** Data Type, as specified in section [2.2.2.10](#).

RuleType: The type of the search scope rule. The value MUST be a **RuleType** Data Type, as specified in section [2.2.2.11](#).

UrlRuleType: The URL type of the search scope rule. The value MUST be a **UrlRuleType** Data Type, as specified in section [2.2.2.11](#).

PropertyID: An integer that uniquely identifies the **managed property** to use by the **search scope rule**.

UserValueString: The **value** used for filtering by the **search scope rule**.

3.1.4.23 **proc_MSS_GetScopesForCompilation**

The **proc_MSS_GetScopesForCompilation** stored procedure is called to retrieve the **search scopes** that are involved for the current **search scope compilation**.

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

```
PROCEDURE proc_MSS_GetScopesForCompilation(  
    @TakeSnapshot          bit,  
    @CompilationID         int OUTPUT,  
    @PreviousCompilationID int OUTPUT  
);
```

@TakeSnapShot: If set to 1 the Compilation Search Scope Set (See **Search Scopes Functionality** as specified in section [3.1.1.2](#)) MUST be replaced with all **search scopes** that have been modified since the last **search scope compilation**. Otherwise, it MUST be 0.

@CompilationID: Upon return from this stored procedure, this parameter MUST be set **to** the current **search scope compilation identifier** for the **search scopes system**.

@PreviousCompilationID: Upon return from this stored procedure, this parameter MUST be set **to** the last **search scope compilation identifier** for the search scope system.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the search scopes result set as defined in Scopes Result Set (Section 2.2.4.1).

3.1.4.24 **proc_MSS_GetSDID**

The **proc_MSS_GetSDID** stored procedure is called to retrieve a **search security descriptor** from the **metadata schema**.

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

```
PROCEDURE proc_MSS_GetSDID(  
    @sdChecksum          int  
);
```

@sdChecksum: An identifier of a search security descriptor.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.24.1 **GetSDID Result Set**

The GetSDID result set returns the list of **search security descriptors** that are associated with **@sdChecksum**. The result set MUST contain zero or more rows which are not ordered.

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

```
SDID          int,  
SD            image;
```

SDID: A unique identifier of the search security descriptor.

SD: The search security descriptor.

3.1.4.25 **proc_MSS_InsertFromSession**

The **proc_MSS_InsertFromSession** stored procedure is called to flush session table data (see tables as specified in Common Result Sets (section [2.2.5](#))) to the **metadata index**.

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

```
PROCEDURE proc_MSS_InsertFromSession(  
    @nCatalogId      smallint,  
    @nIsAlt          int  
);
```

@nCatalogId: An integer which MUST be 1.

@nIsAlt: An integer which MUST be 1 to flush session tables named with the "Alt" suffix. Otherwise, the "non-Alt" versions of the tables will be flushed.

Return Code Values: An integer which MUST be 1.

Result Sets: MUST NOT return any result set.

3.1.4.26 **proc_MSS_OnDocDelete**

The **proc_MSS_OnDocDelete** stored procedure is called to delete the metadata for an item from the **metadata index**.

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

```
PROCEDURE proc_MSS_OnDocDelete(  
    @nCatalogId      smallint,  
    @docid           int  
);
```

@ nCatalogId: A number which MUST be 1.

@docid: A unique identifier of an item.

Return Code Values: An integer which MUST be 1.

Result Sets: MUST NOT return any result set.

3.1.4.27 **proc_MSS_OnEndCrawl**

The **proc_MSS_OnEndCrawl** stored procedure is called to do database-related maintenance at the end of a **crawl**.

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

```
PROCEDURE proc_MSS_OnEndCrawl (
    @nCatalogId          SMALLINT,
    @nCrawlNumber        INT,
    @nCrawlType           INT,
    @nStopped             INT,
    @nSuccessfulTransactions INT,
    @nErrorTransactions   INT,
    @nExcludedTransactions INT,
    @nUnvisitedItems      INT
);
```

@nCatalogId: An integer which **MUST** be 1.

@nCrawlNumber: An identifier of a crawl.

@nCrawlType: The crawl type of the crawl.

@nStopped: A number which **MUST** be 1 if the crawl was stopped by user action. Otherwise, it **MUST** be 0.

@nSuccessfulTransactions: The number of successful transactions completed in the crawl.

@nErrorTransactions: The number of transactions that ended in error during the crawl.

@nExcludedTransactions: The number of transactions that were excluded during the crawl.

@nUnvisitedItems: The number of items that the full crawl for this content source found but that were not found in the current crawl.

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

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

3.1.4.28 proc_MSS_OnStartCrawl

The **proc_MSS_OnStartCrawl** stored procedure is called to do database-related maintenance at the start of a **crawl**.

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

```
PROCEDURE proc_MSS_OnStartCrawl (
    @CatID          int
    @bFullCrawl      smallint
);
```

@CatID: A number which **MUST** be 1.

@bFullCrawl: A number which **MUST** be 1 if the crawl type is a full crawl. Otherwise, it **MUST** be 0.

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

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

3.1.4.29 proc_MSS_PrepareAnchorTextCrawl

The **proc_MSS_PrepareAnchorTextCrawl** stored procedure is called by the **index server** to prepare the **anchor text crawl** and get the oldest **portal content crawl** identifier that has unprocessed anchor text information.

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

```
PROCEDURE proc_MSS_PrepareAnchorTextCrawl(  
    @CrawlID          int OUTPUT  
);
```

@CrawlID: Upon return from this stored procedure, this parameter **MUST** be set to the identifier of the oldest **crawl** that generated the changes in the **anchor text information** structure, as specified in **Abstract Data Model** (section [3.1.1](#)), if changes exist. Otherwise, it **MUST** be set to -1.

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

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

3.1.4.30 proc_MSS_ProcessCommitted

The **proc_MSS_ProcessCommitted** stored procedure is called by the **index server** for every **item** to set the item status as completed and to remove the item from the **crawl queue**, as specified in **Abstract Data Model** (section [3.1.1](#)). In the case of the anchor text crawl, if the item generated links, the index server makes the call to process the links and then sets the item status as completed. If the item doesn't produce a link, the index server inserts the item into the **anchor text completed documents** structure, as specified in **Abstract Data Model** (section [3.1.1](#)).

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

```
PROCEDURE proc_MSS_ProcessCommitted(  
    @FullIncrementalInterval    int,  
    @DeleteOnErrorInterval     int,  
    @DocID                     int,  
    @StartAddressID            int,  
    @ContentSourceID           int,  
    @ProjectID                 int,  
    @CrawlID                   int,  
    @SourceDocID               int,  
    @ItemType                  int,  
    @TransactionType           int,  
    @Scope                     int,  
    @TransactionFlags          int,  
    @AccessURL                 nvarchar(2048),  
    @AccessHash                int,  
    @CompactURL                varchar(40),  
    @CompactHash               int,  
    @ParentCompactURL          nvarchar(40),  
    @ParentCompactHash         int,  
    @DisplayURL                nvarchar(2048),  
    @DisplayHash               int,  
    @LastModifiedTime          bigint,  
    @EndPathFlag               int,  
    @PropMD5                   int,  
    @MD5                       int,
```

```

        @FolderDelCount                int,
        @Host                          nvarchar(300),
        @HostDepth                      int,
        @EnumerationDepth              int,
        @Retry                          int,
        @IndexType                     int,
        @SeqID                          bigint,
        @LCID                           int,
        @UseChangeLog                  int,
        @ChangeLogBatchID               int,
        @ChangeLogCookie                nvarchar(200),
        @ErrorDesc                      nvarchar(1024),
        @hrResult                       int,
        @DocPropsMD5                    bigint,
        @MaxDocId                       int,
        @FolderHighPriority              int,
        @ItemHighPriority                int,
        @ForceRecrawlInterval           int,
        @AccessDeniedCountAllowed        int
    );

```

@FullIncrementalInterval: See **Full Incremental Interval** in section [2.2.2.7](#).

@DeleteOnErrorInterval: See **Delete On Error Interval** in section [2.2.2.8](#).

@DocID: The unique identifier of the **item**.

@StartAddressID: The unique identifier of the **start address**.

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

@ProjectID: See **Project Identifier** in section [2.2.2.1](#).

@CrawlID: The unique identifier of the **crawl**.

@SourceDocID: The identifier of the parent item.

@ItemType: This parameter **MUST be ignored by the server**.

@TransactionType: See **Transaction Type** in section [2.2.2.4](#).

@Scope: The scope of the **transaction**. See **Transaction Scope** in section [2.2.2.5](#).

@TransactionFlags: The transaction flags. See **Transaction Flags** in section [2.2.2.6](#).

@AccessURL: The item access URL.

@AccessHash: The identifier of the *@AccessURL* string.

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

@CompactHash: The identifier of the *@CompactURL* string.

@ParentCompactURL: The compact URL for the parent item.

@ParentCompactHash: The identifier of the *@ParentCompactURL* string.

@DisplayURL: The **display URL** of the item.

@DisplayHash: The identifier of the *@DisplayURL* string.

@LastModifiedTime: The UTC timestamp that indicates when the **item** was modified.

@EndPathFlag: A bit mask of flags. Its value **MUST** be in the following table.

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

@PropMD5: The identifier of the item metadata. In the **incremental crawl**, if the value of the parameter is different than the existing value, the item and any children will be re-crawled.

@MD5: The identifier of the **item** content.

@FolderDelCount: An integer which **MUST** be 0.

@Host: The **host name** for the current item.

@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**.

@Retry: Not used. **MUST** be ignored.

@IndexType: Specifies whether the **item** can be returned in search results. Its value **MUST** be in the following table.

Value	Description
0	The item MUST NOT be returned in search results.
1	The item can be returned in search results.

@SeqID: The identifier of the item in the current **crawl**.

@LCID: The language code identifier (LCID).

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

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

@ChangeLogCookie: A token that represents the last change that was retrieved from the **change log**.

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

@hrResult: A **crawl** error represented as an **HRESULT**.

@DocPropsMD5: The identifier of the **item** metadata.

@MaxDocId: An integer which must be -1 if there is no limit of the number of items in the search application. If there is a limit, the total number of items must be restricted to be less than this limit. [<7>](#)

@FolderHighPriority: An integer which MUST be 1 when the document is a container of high priority items. Otherwise, it MUST be 0. See the following explanation of *@ItemHighPriority*.

@ItemHighPriority: An integer which MUST be 1 when the item will be crawled before other items in the **crawl queue**, as specified in **Abstract Data Model** (section 3.1.1). Otherwise, it MUST be 0.

@ ForceRecrawlInterval: An integer that represents the number of times the server unsuccessfully attempts to retrieve the changes before switching to a **full crawl**.<8>

@AccessDeniedCountAllowed: An integer that represents the number of times a item can fail with an access denied error before the item is removed from the crawl URL history.<9>

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	The stored procedure failed. No changes were persisted.
1	Successful execution.

Result Sets: SHOULD NOT<10> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.31 proc_MSS_PushSD

The **proc_MSS_PushSD** stored procedure is called to store a new **search security descriptor**.

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

```
PROCEDURE proc_MSS_PushSD(  
    @sd                image,  
    @sdChecksum        int,  
    @type              int,  
    @sdid              int OUTPUT  
);
```

@sd: The search security descriptor.

@sdChecksum: An identifier of a search security descriptor.

@type: An integer which MUST be 1 when the search security descriptor is in the format defined in [MS-DTYP], section 2.4.6. Otherwise, it MUST be 0.

@sdid: Upon return from this stored procedure, this parameter MUST be set to a unique identifier of the new Search Security Descriptor.

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.

3.1.4.32 proc_MSS_ReportScopesCompilationBegin

The **proc_MSS_ReportScopesCompilationBegin** stored procedure is called to notify the **search scopes system** that **search scope compilation** has begun.

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

```
PROCEDURE proc_MSS_ReportScopesCompilationBegin();
```

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.

3.1.4.33 proc_MSS_ReportScopesCompilationEnd

The **proc_MSS_ReportScopesCompilationEnd** stored procedure is called to notify the **search scopes system** that **search scope compilation** has completed.

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

```
PROCEDURE proc_MSS_ReportScopesCompilationEnd(  
    @Aborted          bit  
) ;
```

@Aborted: A bit which MUST be 1 if the current compilation is treated by the **search scopes system** as aborted. Otherwise, it MUST be 0.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT [<13>](#) return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.34 proc_MSS_ResetCatalog

The **proc_MSS_ResetCatalog** stored procedure is called to clear all customer data from the **metadata index**.

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

```
PROCEDURE proc_MSS_ResetCatalog(  
    @nCatalogId      smallint  
) ;
```

@ nCatalogId: An integer which MUST be 1.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.35 proc_MSS_SetBigConfigurationProperty

The **proc_MSS_SetBigConfigurationProperty** stored procedure is called to update the value of the specified **big configuration property** or to insert it if it doesn't exist.

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

```
PROCEDURE proc_MSS_SetBigConfigurationProperty(  

```

```

        @Name          nvarchar(64),
        @Value          image,
        @Version        int
    );

```

@Name: The name of the configuration property to be set.

@Value: The new value of the configuration property.

@Version: The new version number of the property. If the value of this parameter is not equal to the current version (or version plus one) of the property, then the property value MUST NOT be changed.

Return Code Values: An integer which MUST be listed in the following table.

Value	Description
0	Successful execution.
100	The value of the version parameter is not equal to the current version of the property or current version plus one.

Result Sets: SHOULD NOT [<14>](#) return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.36 **proc_MSS_SetConfigurationProperty**

The **proc_MSS_SetConfigurationProperty** stored procedure is called to set the value of the specified **configuration property** or to insert it if it doesn't exist.

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

```

PROCEDURE proc_MSS_SetConfigurationProperty(
    @Name          nvarchar(64),
    @Value          sql_variant);

```

@Name: The name of the configuration property.

@Value: The value of the configuration property.

Return Code Values: 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.

3.1.4.37 **proc_MSS_SetCrawledPropertyIsSampleCacheFull**

The **proc_MSS_SetCrawledPropertyIsSampleCacheFull** stored procedure is called to update the *IsSampleCacheFull* flag in the **sample crawled properties set** for the specified **crawled property**.

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

```

PROCEDURE proc_MSS_SetCrawledPropertyIsSampleCacheFull (

```

```

        @CrawledPropId          int,
        @IsSampleCacheFull      bit,
        @UseDateTrigger         bit
    );

```

@CrawledPropId: A unique identifier of a crawled property.

@IsSampleCacheFull: **A bit which MUST be 1 if the sample crawled properties set is complete. Otherwise, it MUST be 0.**

@UseDateTrigger: **A bit which MUST be 0.**

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT [<16>](#) return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.38 **proc_MSS_ShareScopesCompilationInfo**

The **proc_MSS_ShareScopesCompilationInfo** stored procedure is called every 30 seconds by the **search application** to report information about **search scope compilation**.

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

```

PROCEDURE proc_MSS_ShareScopesCompilationInfo (
    @CompilationPercentComplete    smallint,
    @QueryServers                  int,
    @ShouldBeCompiling             bit OUTPUT
);

```

@CompilationPercentComplete: The percentage of search scopes compilation completed.

@QueryServers: The number of **query servers** in the **search application**.

@ShouldBeCompiling: Upon return from this stored procedure, this parameter MUST be set to 1 if a **search scope compilation** is in progress. Otherwise, it MUST be set 0.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT [<17>](#) return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.39 **proc_MSS_TruncateCleanupTable**

The **proc_MSS_TruncateCleanupTable** stored procedure is called to delete structures used to adjust the *IsSampleCacheFull* property in the **crawled property set** (as defined in [\[MS-SQLPADM\]](#), section 2.2.1.x) of the **metadata index**.

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

```

PROCEDURE proc_MSS_TruncateCleanupTable();

```

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT [<18>](#) return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.40 **proc_MSS_Recompile**

The proc_MSS_Recompile stored procedure is called periodically to recompile some of the search stored procedures.

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

```
PROCEDURE proc_MSS_Recompile();
```

Return Code Values: An integer which MUST be 0.

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 **Timer Events**

None.

3.1.6 **Other Local Events**

None.

3.2 **WSS Server Details**

This section describes the Windows® SharePoint® Services Server Role.

3.2.1 **Abstract Data Model**

See **Abstract Data Model** (section [3.1.1](#)).

3.2.2 **Timers**

See **Timers** (section [3.1.2](#)).

3.2.3 **Initialization**

See **Initialization** (section [3.1.3](#)).

3.2.4 **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 specified in [\[MS-TDS\]](#), section 2.2.6.9.9 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. If the stored procedures are not provided the expected calling parameters or return the expected result set values, the system behavior is indeterminate.

For the sake of 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, since the ordinal position of any column with no defined name is expected by the protocol client. Such names are designated in the text using curly braces in the form {*name*}.

This section describes the following stored procedures:

Procedure Name	Description
proc_MSS_AddAndReturnCrawledProperty (specified in section 3.1.4.1)	Adds the crawled property to the metadata schema if it doesn't exist and returns its parameters if it does.
proc_MSS_AddCrawledPropertyCategoryWithDefaults (specified in section 3.1.4.2)	Adds a crawled property category to the metadata schema .
proc_MSS_Crawl (specified in section 3.1.4.4)	Gets or changes a given crawl 's state.
proc_MSS_FlushTemp0 (specified in section 3.1.4.5)	Processes all the links for a given item within a given crawl from the MSSTranTempTable0 table (section 2.2.6.11) to the crawl queue and GADM: crawl URL history structures, as specified in Abstract Data Model (section 3.1.1).
proc_MSS_GetBigConfigurationProperty (specified in section 3.1.4.6)	Gets a value of the specified big configuration property .
proc_MSS_GetConfigurationProperty (specified in section 3.2.4.1)	Gets the value of a property of the configuration property structure.
proc_MSS_GetCrawledPropertyUpdates (specified in section 3.1.4.11)	Lists crawled properties which have been added or updated in the metadata schema after the given time.
proc_MSS_GetCrawledPropMappingUpdates (specified in section 3.1.4.12)	Lists mappings for crawled properties which have been added or updated in the metadata schema after the given time.
proc_MSS_GetCrawls (specified in section 3.1.4.13)	Informs the server that the search application is initializing, and retrieves a list of crawls in progress.
proc_MSS_GetDocStatus (specified in section 3.1.4.15)	Gets the crawl status of the specified set of items .
proc_MSS_GetManagedProperties (specified in section 3.1.4.16)	Lists managed properties from the metadata schema which were added or modified on or after the given time.
proc_MSS_GetNextCrawlBatch (specified in section 3.1.4.17)	Retrieves a list of items from the crawl queue , as specified in Abstract Data Model (section 3.1.1), for a given crawl .
proc_MSS_GetSampleExtremes (specified in section 3.1.4.18)	Lists the crawled propertiess whose number of taken samples are either above or below the given parameters.
proc_MSS_GetSchemaHighLevelInfo (specified in section 3.1.4.19)	Retrieves last modified and last deleted timestamps from the metadata

Procedure Name	Description
3.1.4.19)	schema.
proc_MSS_GetSchemaMappings (specified in section 3.1.4.20)	Retrieves the list of mappings between crawled properties and managed properties .
proc_MSS_GetSchemaParameters (specified in section 3.1.4.21)	Retrieves a list of schema parameters from the metadata schema .
proc_MSS_GetSDID (specified in section 3.1.4.24)	Retrieves a search security descriptor from the metadata schema .
proc_MSS_InsertFromSession (specified in section 3.1.4.25)	Flushes session table data (see tables in section 2.2.5) to the metadata index .
proc_MSS_OnDocDelete (specified in section 3.1.4.26)	Deletes the metadata for an item from the metadata index .
proc_MSS_OnEndCrawl (specified in section 3.1.4.27)	Performs database-related maintenance at the end of a crawl .
proc_MSS_OnStartCrawl (specified in section 3.1.4.28)	Performs database related maintenance at the start of a crawl .
proc_MSS_ProcessCommitted (specified in section 3.1.4.30)	Sets each item's status as completed and removes it from the crawl queue , as specified in Abstract Data Model (section 3.1.1).
proc_MSS_PushSD (specified in section 3.1.4.31)	Stores a new search security descriptor .
proc_MSS_ResetCatalog (specified in section 3.1.4.34)	Clears all customer data from the metadata index .
proc_MSS_SetBigConfigurationProperty (specified in section 3.1.4.35)	Updates the value of the specified big configuration property or inserts it if it doesn't exist.
proc_MSS_SetConfigurationProperty (specified in section 3.1.4.36)	Sets the value of the specified configuration property or inserts it if it doesn't exist.
proc_MSS_SetCrawledPropertyIsSampleCacheFull (specified in section 3.1.4.37)	Updates the <i>IsSampleCacheFull</i> flag in the sample Crawled properties set for the specified crawled property .
proc_MSS_TruncateCleanupTable (specified in section 3.1.4.39)	Clears structures used to adjust the <i>IsSampleCacheFull</i> property in the crawled property set (as defined in [MS-SQLPADM] , section 2.2.1) of the metadata index

3.2.4.1 proc_MSS_GetConfigurationProperty

The **proc_MSS_GetConfigurationProperty** stored procedure is called to retrieve the value of a property of the **configuration property** structure. If the property is found in the **configuration property** structure, this value **MUST** be returned. Otherwise **NULL MUST be returned**.

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

```
PROCEDURE proc_MSS_GetConfigurationProperty(  
    @Name          nvarchar(64),  
    @Value         sql_variant OUTPUT  
);
```

@Name: Name of the property.

@Value: Upon return from this stored procedure, this parameter MUST be set **to the value of the property**.

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.

3.2.5 Timer Events

None.

3.2.6 Other Local Events

None.

4 Protocol Examples

This section provides specific example scenarios for end-to-end index server tasks. These examples describe in detail the process of communication between the various server components involved in the index server processes.

4.1 Full Crawl

This example describes the crawl-specific requests made and responses returned when a full crawl of a content source is requested.

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 Tabular Data Stream [\[MS-TDS\]](#) protocol.

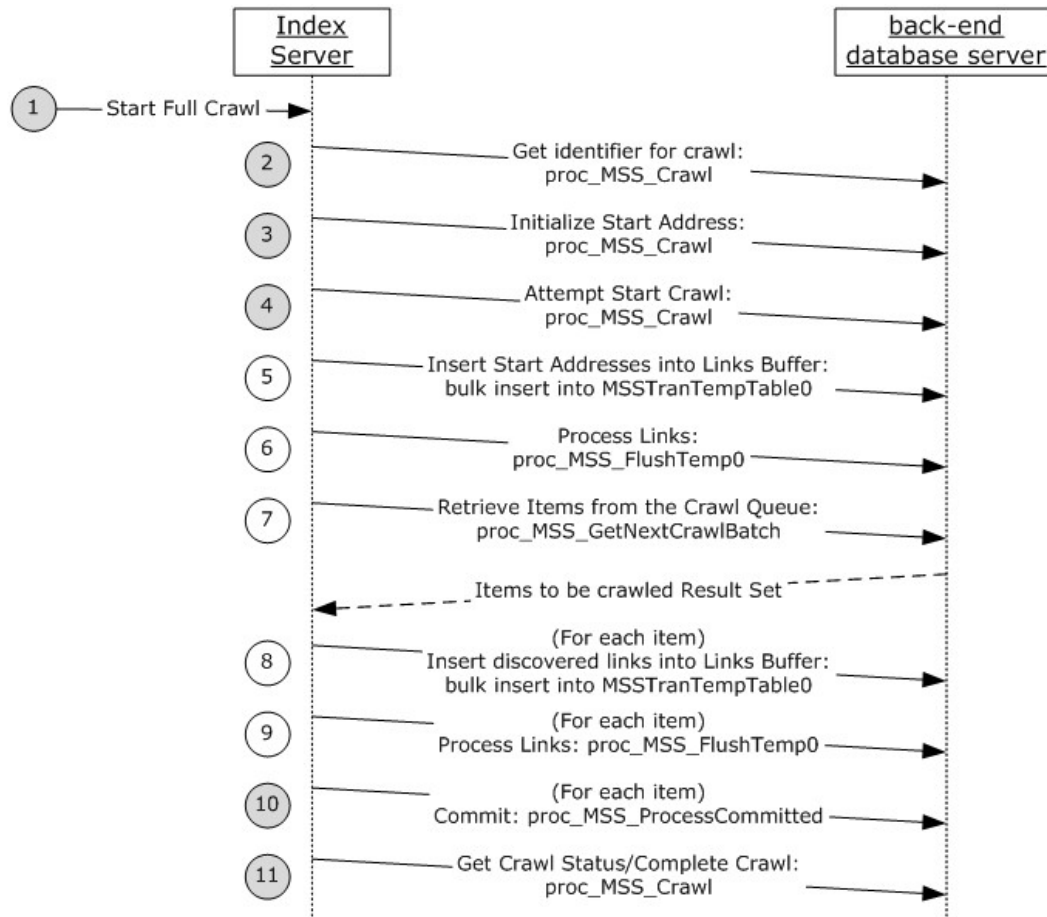


Figure 8: Example of a full crawl operation

The steps in the preceding diagram are explained in the following table:

1. The command that starts the crawl comes to the index server from the object model.
2. The **index server** gets a crawl identifier from the database backend server by invoking **proc_MSS_Crawl** with **@CrawlStage = 1**.

3. The index server initializes the list of start addresses for the current crawl in the backend server. The gather calls **proc_MSS_Crawl** with *@CrawlStage* = 2 for each start address.
4. The index server determines if the crawl can start by calling **proc_MSS_Crawl** with *@CrawlStage* = 3. A crawl can start if none of the **content sources** are involved in another crawl. If this condition is met then the stored procedure returns *@CrawlStatus*=4 which indicates that the crawl started; otherwise it returns *@CrawlStatus*=5 which indicates that the crawl is aborted.
5. If the crawl can start, the index server bulk inserts the start addresses in the links buffer, as specified in **Abstract Data Model** (section [3.1.1](#)).
6. The index server makes the call to process the links.
7. The index server periodically calls the backend to retrieve new items from the **crawl queue**, as specified in **Abstract Data Model** (section [3.1.1](#)).
8. For each filtered item, the links discovered by the index server are bulk inserted into the **links buffer**, as specified in **Abstract Data Model** (section [3.1.1](#)).
9. For each item, the index server makes the call to process the links. On the back-end database server, these actions are taken:
10. If the links are not in the crawl URL history then they are added to it.
11. If the links have not been crawled yet, they are inserted into the crawl queue.
12. After processing the links, the index server calls the backend and sets the item status as completed.
13. The index server periodically calls the backend to retrieve the crawl status by invoking **proc_MSS_Crawl** with *@CrawlStage* = 7; if the **crawl queue** is empty, the backend completes the crawl and returns to the index server a crawl complete status. Otherwise the status is not changed.

4.2 Incremental Crawl

This example describes the crawl-specific requests made and responses returned when an incremental crawl of a content source is requested.

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 Tabular Data Stream [\[MS-TDS\]](#) protocol.

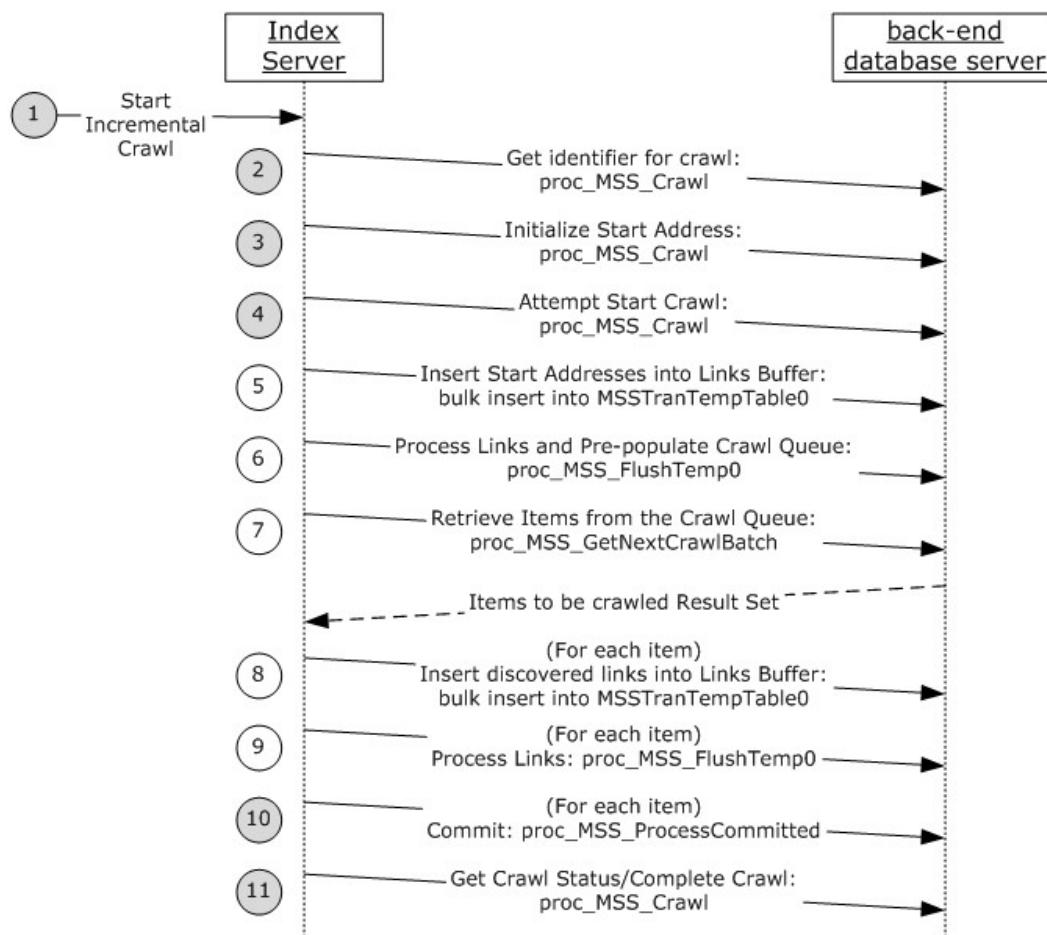


Figure 9: Example of an incremental crawl operation

The steps in the incremental crawl are identical to the steps in the full crawl described in section 4.1, with only one exception in step 6. The difference is that after the start addresses are inserted into the **links buffer**, as specified in **Abstract Data Model** (section 3.1.1), the process links action can pre-populate the **crawl queue** based on the **crawl url history**, as specified in **Abstract Data Model** (section 3.1.1).

4.3 Delete Crawl

This example describes the crawl-specific requests made and responses returned when a delete Crawl of a content source is requested.

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 Tabular Data Stream [\[MS-TDS\]](#) protocol.

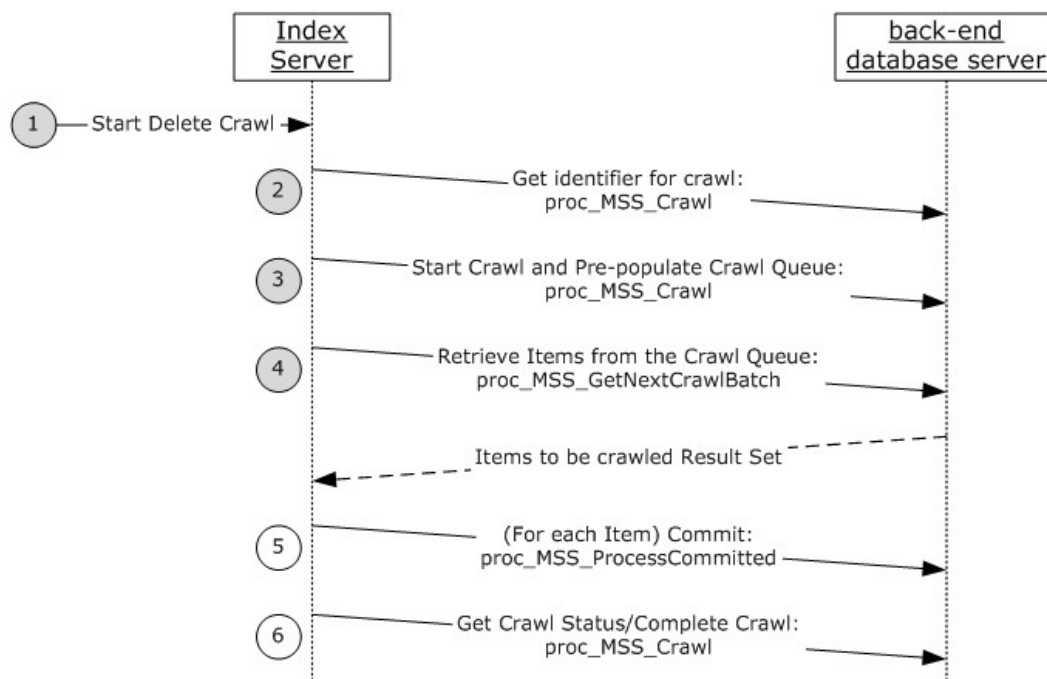


Figure 10: Example of a delete crawl operation

The steps in the preceding diagram are explained in the following table.

The **Delete crawl** removes items, which are associated with a deleted content source or start address, from the full-text and metadata indices.

1. The index server starts a delete crawl automatically after a content source or a start address is deleted by the administrator.
2. The **index server** gets a crawl identifier from the database backend server by invoking **proc_MSS_Crawl** with `@CrawlStage = 1`.
3. The index server makes a call to start the crawl and pre-populate the **crawl queue** to delete the items associated with a given content source or start address
4. The index server periodically calls the server to retrieve items to crawl from the **crawl queue**.
5. The index server calls the server to set the item status as completed and to remove the item from the **crawl queue**
6. The index server periodically calls the server to retrieve the **crawl status**. If the crawl queue is empty, the server completes the crawl and returns a crawl complete status.

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 Tabular Data Stream [\[MS-TDS\]](#) protocol.

This protocol requires that the database access account used by the index server have access to the appropriate search database on the back-end database server. If the account does not have the correct access rights, access will be denied when attempting to set up the [MS-TDS] connection to the search database, or when calling the stored procedures.

Interactions with SQL are susceptible to tampering and other forms of security risks. Implementers are advised to sanitize input parameters for stored procedures prior to invoking the stored procedure

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® Office SharePoint® Server 2007
- Microsoft® SQL Server® 2005
- Microsoft® SQL Server® 2008
- Microsoft® SQL Server® 2008 R2
- Windows® SharePoint® Services 3.0

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 2.2.6.2:](#) This functionality was added as part of the Office SharePoint Server 2007 Service Pack 2 (SP2).

[<2> Section 3.1.4.1:](#) 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 3.1.4.2:](#) 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.

[<4> Section 3.1.4.5:](#) This functionality was removed as part of the Office SharePoint Server 2007 Infrastructure Update.

[<5> Section 3.1.4.5:](#) This functionality was removed as part of the Office SharePoint Server 2007 Infrastructure Update.

[<6> Section 3.1.4.10:](#) 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 3.1.4.30:](#) This functionality was removed as part of the Office SharePoint Server 2007 Infrastructure Update.

[<8> Section 3.1.4.30:](#) This functionality was added as part of the Office SharePoint Server 2007 Infrastructure Update.

[<9> Section 3.1.4.30:](#) This functionality was added as part of the Office SharePoint Server 2007 Service Pack 2 (SP2).

[<10> Section 3.1.4.30:](#) 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 3.1.4.31:](#) 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.

[<12> Section 3.1.4.32:](#) 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.

[<13> Section 3.1.4.33:](#) 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.

[<14> Section 3.1.4.35:](#) 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 3.1.4.36:](#) 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 3.1.4.37:](#) 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 3.1.4.38:](#) 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 3.1.4.39:](#) 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 3.1.4.40:](#) 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 3.2.4.1:](#) 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

A

Abstract data model
server ([section 3.1.1](#) 31, [section 3.2.1](#) 68)
[Applicability](#) 17

B

[Binary structures - overview](#) 22
[Bit fields - overview](#) 22

C

[Capability negotiation](#) 17
[Change tracking](#) 79
Common data types
 [overview](#) 19
[Compilation state type simple type](#) 22
[Crawl status simple type](#) 19
[Crawl type simple type](#) 19

D

Data model - abstract
 server ([section 3.1.1](#) 31, [section 3.2.1](#) 68)
Data types
 [common](#) 19
 [compilation state type simple type](#) 22
 [crawl status simple type](#) 19
 [crawl type simple type](#) 19
 [delete on error interval simple type](#) 21
 [filter behavior simple type](#) 21
 [full incremental interval simple type](#) 21
 [index type simple type](#) 21
 [managed type simple type](#) 22
 [project identifier simple type](#) 19
 [rule type simple type](#) 21
 [transaction flags simple type](#) 20
 [transaction scope simple type](#) 20
 [transaction type simple type](#) 20
 [UrlRule type simple type](#) 21
Data types - simple
 [compilation state type](#) 22
 [crawl status](#) 19
 [crawl type](#) 19
 [delete on error interval](#) 21
 [filter behavior](#) 21
 [full incremental interval](#) 21
 [index type](#) 21
 [managed type](#) 22
 [project identifier](#) 19
 [rule type](#) 21
 [transaction flags](#) 20
 [transaction scope](#) 20
 [transaction type](#) 20
 [UrlRule type](#) 21
[Delete crawl example](#) 74
[Delete on error interval simple type](#) 21

E

Events
 local - server ([section 3.1.6](#) 68, [section 3.2.6](#) 71)
 timer - server ([section 3.1.5](#) 68, [section 3.2.5](#) 71)
Examples
 [delete crawl](#) 74
 [full crawl](#) 72
 [incremental crawl](#) 73
 [overview](#) 72

F

[Fields - vendor-extensible](#) 18
[Filter behavior simple type](#) 21
[Flag structures - overview](#) 22
[Full crawl example](#) 72
[Full incremental interval simple type](#) 21

G

[Glossary](#) 6

I

[Implementer - security considerations](#) 76
[Incremental crawl example](#) 73
[Index of security parameters](#) 76
[Index type simple type](#) 21
[Informative references](#) 8
Initialization
 server ([section 3.1.3](#) 35, [section 3.2.3](#) 68)
Interfaces - server
 [MOSS server interface](#) 31
 [WSS server interface](#) 68
[Introduction](#) 6

L

Local events
 server ([section 3.1.6](#) 68, [section 3.2.6](#) 71)

M

[Managed type simple type](#) 22
Message processing
 server ([section 3.1.4](#) 35, [section 3.2.4](#) 68)
Messages
 [binary structures](#) 22
 [bit fields](#) 22
 [common data types](#) 19
 [flag structures](#) 22
 [MSSAnchorChangeLot table structure](#) 23
 [MSSAnchorText table structure](#) 23
 [MSSAnchorTransactions table structure](#) 24
 [MSSCrawledPropSamples table structure](#) 24

MSSSessionDefinitions/MSSSessionDefinitionsAlt table structure	25	proc_MSS_TruncateCleanupTable	67
MSSSessionDocProps/MSSSessionDocPropsAlt table structure	25	MOSS server interface	31
MSSSessionDocSdIds/MSSSessionDocSdIdsAlt table structure	26	MSSAnchorChangeLot table structure	23
MSSSessionDocSignatures/MSSSessionDocSignaturesAlt table structure	26	MSSAnchorText table structure	23
MSSSessionDuplicateHashes/MSSSessionDuplicateHashesAlt table structure	27	MSSAnchorTransactions table structure	24
MSSSessionExistingDocs/MSSSessionExistingDocsAlt table structure	27	MSSCrawledPropSamples table structure	24
MSSTranTempTable0 table structure	27	MSSSessionDefinitions/MSSSessionDefinitionsAlt table structure	25
Scopes result set	23	MSSSessionDocProps/MSSSessionDocPropsAlt table structure	25
table structures	23	MSSSessionDocSdIds/MSSSessionDocSdIdsAlt table structure	26
transport	19	MSSSessionDocSignatures/MSSSessionDocSignaturesAlt table structure	26
view structures	23	MSSSessionDuplicateHashes/MSSSessionDuplicateHashesAlt table structure	27
XML structures	30	MSSSessionExistingDocs/MSSSessionExistingDocsAlt table structure	27
Methods		MSSTranTempTable0 table structure	27
proc_MSS_AddAndReturnCrawledProperty	38		
proc_MSS_AddCrawledPropertyCategoryWithDefaults	39	N	
proc_MSS_CommitAnchorTextCrawl	40	Normative references	7
proc_MSS_Crawl	40	O	
proc_MSS_FlushTemp0	42	Overview (synopsis)	8
proc_MSS_GetBigConfigurationProperty	43	P	
proc_MSS_GetCompiledScopeRules	43	Parameters - security index	76
proc_MSS_GetCompiledScopes	44	Preconditions	17
proc_MSS_GetCompletedScopesCompilationID	45	Prerequisites	17
proc_MSS_GetConfigurationProperty (section 3.1.4.10)	45, section 3.2.4.1 70)	proc_MSS_AddAndReturnCrawledProperty method	38
proc_MSS_GetCrawledPropertyUpdates	45	proc_MSS_AddCrawledPropertyCategoryWithDefault s method	39
proc_MSS_GetCrawledPropMappingUpdates	46	proc_MSS_CommitAnchorTextCrawl method	40
proc_MSS_GetCrawls	47	proc_MSS_Crawl method	40
proc_MSS_GetDeletedScopesForCompilation	48	proc_MSS_FlushTemp0 method	42
proc_MSS_GetDocStatus	48	proc_MSS_GetBigConfigurationProperty method	43
proc_MSS_GetManagedProperties	49	proc_MSS_GetCompiledScopeRules method	43
proc_MSS_GetNextCrawlBatch	51	proc_MSS_GetCompiledScopes method	44
proc_MSS_GetSampleExtremes	54	proc_MSS_GetCompletedScopesCompilationID method	45
proc_MSS_GetSchemaHighLevelInfo	54	proc_MSS_GetConfigurationProperty method (section 3.1.4.10)	45, section 3.2.4.1 70)
proc_MSS_GetSchemaMappings	56	proc_MSS_GetCrawledPropertyUpdates method	45
proc_MSS_GetSchemaParameters	56	proc_MSS_GetCrawledPropMappingUpdates method	46
proc_MSS_GetScopeRulesForCompilation	57	proc_MSS_GetCrawls method	47
proc_MSS_GetScopesForCompilation	58	proc_MSS_GetDeletedScopesForCompilation method	48
proc_MSS_GetSDID	58	proc_MSS_GetDocStatus method	48
proc_MSS_InsertFromSession	59	proc_MSS_GetManagedProperties method	49
proc_MSS_OnDocDelete	59	proc_MSS_GetNextCrawlBatch method	51
proc_MSS_OnEndCrawl	59	proc_MSS_GetSampleExtremes method	54
proc_MSS_OnStartCrawl	60	proc_MSS_GetSchemaHighLevelInfo method	54
proc_MSS_PrepareAnchorTextCrawl	61	proc_MSS_GetSchemaMappings method	56
proc_MSS_ProcessCommitted	61	proc_MSS_GetSchemaParameters method	56
proc_MSS_PushSD	64	proc_MSS_GetScopeRulesForCompilation method	57
proc_MSS_Recompile	68		
proc_MSS_ReportScopesCompilationBegin	64		
proc_MSS_ReportScopesCompilationEnd	65		
proc_MSS_ResetCatalog	65		
proc_MSS_SetBigConfigurationProperty	65		
proc_MSS_SetConfigurationProperty	66		
proc_MSS_SetCrawledPropertyIsSampleCacheFull	66		
proc_MSS_ShareScopesCompilationInfo	67		

[proc MSS GetScopesForCompilation method](#) 58
[proc MSS GetSDID method](#) 58
[proc MSS InsertFromSession method](#) 59
[proc MSS OnDocDelete method](#) 59
[proc MSS OnEndCrawl method](#) 59
[proc MSS OnStartCrawl method](#) 60
[proc MSS PrepareAnchorTextCrawl method](#) 61
[proc MSS ProcessCommitted method](#) 61
[proc MSS PushSD method](#) 64
[proc MSS Recompile method](#) 68
[proc MSS ReportScopesCompilationBegin method](#) 64
[proc MSS ReportScopesCompilationEnd method](#) 65
[proc MSS ResetCatalog method](#) 65
[proc MSS SetBigConfigurationProperty method](#) 65
[proc MSS SetConfigurationProperty method](#) 66
[proc MSS SetCrawledPropertyIsSampleCacheFull method](#) 66
[proc MSS ShareScopesCompilationInfo method](#) 67
[proc MSS TruncateCleanupTable method](#) 67
[Product behavior](#) 77
[Project identifier simple type](#) 19

R

[References](#)
[informative](#) 8
[normative](#) 7
[Relationship to other protocols](#) 16
[Result sets - messages](#)
[Scopes](#) 23
[Rule type simple type](#) 21

S

[Scopes result set](#) 23
[Security](#)
[implementer considerations](#) 76
[parameter index](#) 76
[Sequencing rules](#)
[server](#) ([section 3.1.4](#) 35, [section 3.2.4](#) 68)
[Server](#)
[abstract data model](#) ([section 3.1.1](#) 31, [section 3.2.1](#) 68)
[initialization](#) ([section 3.1.3](#) 35, [section 3.2.3](#) 68)
[local events](#) ([section 3.1.6](#) 68, [section 3.2.6](#) 71)
[message processing](#) ([section 3.1.4](#) 35, [section 3.2.4](#) 68)
[MOSS server interface](#) 31
[overview](#) ([section 3.1](#) 31, [section 3.2](#) 68)
[proc MSS AddAndReturnCrawledProperty method](#) 38
[proc MSS AddCrawledPropertyCategoryWithDefaults method](#) 39
[proc MSS CommitAnchorTextCrawl method](#) 40
[proc MSS Crawl method](#) 40
[proc MSS FlushTemp0 method](#) 42
[proc MSS GetBigConfigurationProperty method](#) 43
[proc MSS GetCompiledScopeRules method](#) 43
[proc MSS GetCompiledScopes method](#) 44

[proc MSS GetCompletedScopesCompilationID method](#) 45
[proc MSS GetConfigurationProperty method](#) ([section 3.1.4.10](#) 45, [section 3.2.4.1](#) 70)
[proc MSS GetCrawledPropertyUpdates method](#) 45
[proc MSS GetCrawledPropMappingUpdates method](#) 46
[proc MSS GetCrawls method](#) 47
[proc MSS GetDeletedScopesForCompilation method](#) 48
[proc MSS GetDocStatus method](#) 48
[proc MSS GetManagedProperties method](#) 49
[proc MSS GetNextCrawlBatch method](#) 51
[proc MSS GetSampleExtremes method](#) 54
[proc MSS GetSchemaHighLevelInfo method](#) 54
[proc MSS GetSchemaMappings method](#) 56
[proc MSS GetSchemaParameters method](#) 56
[proc MSS GetScopeRulesForCompilation method](#) 57
[proc MSS GetScopesForCompilation method](#) 58
[proc MSS GetSDID method](#) 58
[proc MSS InsertFromSession method](#) 59
[proc MSS OnDocDelete method](#) 59
[proc MSS OnEndCrawl method](#) 59
[proc MSS OnStartCrawl method](#) 60
[proc MSS PrepareAnchorTextCrawl method](#) 61
[proc MSS ProcessCommitted method](#) 61
[proc MSS PushSD method](#) 64
[proc MSS Recompile method](#) 68
[proc MSS ReportScopesCompilationBegin method](#) 64
[proc MSS ReportScopesCompilationEnd method](#) 65
[proc MSS ResetCatalog method](#) 65
[proc MSS SetBigConfigurationProperty method](#) 65
[proc MSS SetConfigurationProperty method](#) 66
[proc MSS SetCrawledPropertyIsSampleCacheFull method](#) 66
[proc MSS ShareScopesCompilationInfo method](#) 67
[proc MSS TruncateCleanupTable method](#) 67
[sequencing rules](#) ([section 3.1.4](#) 35, [section 3.2.4](#) 68)
[timer events](#) ([section 3.1.5](#) 68, [section 3.2.5](#) 71)
[timers](#) ([section 3.1.2](#) 35, [section 3.2.2](#) 68)
[WSS server interface](#) 68
[Simple data types](#)
[compilation state type](#) 22
[crawl status](#) 19
[crawl type](#) 19
[delete on error interval](#) 21
[filter behavior](#) 21
[full incremental interval](#) 21
[index type](#) 21
[managed type](#) 22
[project identifier](#) 19
[rule type](#) 21
[transaction flags](#) 20
[transaction scope](#) 20

[transaction type](#) 20
[UrlRule type](#) 21
[Standards assignments](#) 18
Structures
 [binary](#) 22
 [table and view](#) 23
 [XML](#) 30

T

Table structures
 [MSSAnchorChangeLog](#) 23
 [MSSAnchorText](#) 23
 [MSSAnchorTransactions](#) 24
 [MSSCrawledPropSamples](#) 24
 [MSSSessionDefinitions/MSSSessionDefinitionsAlt](#) 25
 [MSSSessionDocProps/MSSSessionDocPropsAlt](#) 25
 [MSSSessionDocSdIds/MSSSessionDocSdIdsAlt](#) 26
 [MSSSessionDocSignatures/MSSSessionDocSignaturesAlt](#) 26
 [MSSSessionDuplicateHashes/MSSSessionDuplicateHashesAlt](#) 27
 [MSSSessionExistingDocs/MSSSessionExistingDocsAlt](#) 27
 [MSSTranTempTable0](#) 27
[Table structures - overview](#) 23
Timer events
 server ([section 3.1.5](#) 68, [section 3.2.5](#) 71)
Timers
 server ([section 3.1.2](#) 35, [section 3.2.2](#) 68)
[Tracking changes](#) 79
[Transaction flags simple type](#) 20
[Transaction scope simple type](#) 20
[Transaction type simple type](#) 20
[Transport](#) 19

U

[UrlRule type simple type](#) 21

V

[Vendor-extensible fields](#) 18
[Versioning](#) 17
[View structures - overview](#) 23

W

[WSS server interface](#) 68

X

[XML structures](#) 30