[MS-SQLPADM]:
SQL Administration 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.mspx). If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability
06/27/2008	1.0	Major	Revised and edited the technical content
12/12/2008	1.01	Editorial	Revised and edited the technical content
07/13/2009	1.02	Major	Changes made for template compliance
08/28/2009	1.03	Editorial	Revised and edited the technical content
11/06/2009	1.04	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

This document specifies the communication sequences that are used by the protocol client (Web and application servers) to perform data query and update commands on the protocol server in relation to search administration functions.

1.1 Glossary

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

HRESULT
language code identifier (LCID)

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

authority level
back-end database server
best bet
binary large object (BLOB)
content source
crawl
crawl log
crawl queue
crawl status
crawled property
crawled property category
crawler
datetime
delete crawl
display URL
federated location
federation
folder
front-end Web server
full crawl
full-text index catalog
host name
incremental crawl
index server
item
keyword
keyword consumer
keyword consumer group
keyword synonym
list item
managed property
managed property alias
mapping order
metadata index
metadata schema
property identifier
property oriented rank
proxy
query independent rank
query result

query server
query text
result set
return code
search application
search database
search query
search scope
search scope compilation
search scope consumer
search scope display group
search scope index
search scope rule
search scope rule value
search scopes system
site
site collection
site collection administrator
start address
stored procedure
subdomain
Transact-Structured Query Language (T-SQL)
Uniform Resource Identifier (URI)
Uniform Resource Locator (URL)
variant type
visualization

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.

child farm: A server farm that relies on an associated parent farm to provide crawl capabilities. Any search query that is issued on the child farm is redirected to the parent farm and search results are obtained from the parent farm for display to the user.

click distance: The minimum number of links that need to be followed to create a path between an authority page and an item.

federated location definition: The configuration settings that describe how to issue a query for a given federated location and display the search results.

parent farm: A server farm that crawls content from another server farm and also responds to query requests from that server farm.

ranking parameter: A value that is used to influence the algorithm that determines the rank of an item.

visible scope: A search scope that is displayed to site collection administrators and users.

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

[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

[MS-SQLPGAT] Microsoft Corporation, "SQL Gatherer 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

[MSDN-ESCRXT] Microsoft Corporation, "Enterprise Search Core Results XSLT Transformation", http://msdn.microsoft.com/en-us/library/ms584121.aspx

[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 **front-end Web server** and the **back-end database server** used to satisfy requests for common search administration tasks. This server-to-server protocol uses the Tabular Data Stream Protocol, as specified in [MS-TDS], as its transport between the front-end Web server and the back-end database server.

1.3.1 Metadata Schema

The protocol allows clients to add, change, retrieve, and delete **metadata schema** information from a store on the back-end database server.

The following diagram specifies the data flow between the protocol client and the protocol server with regards to administration of **crawled property categories** and **crawled properties**.

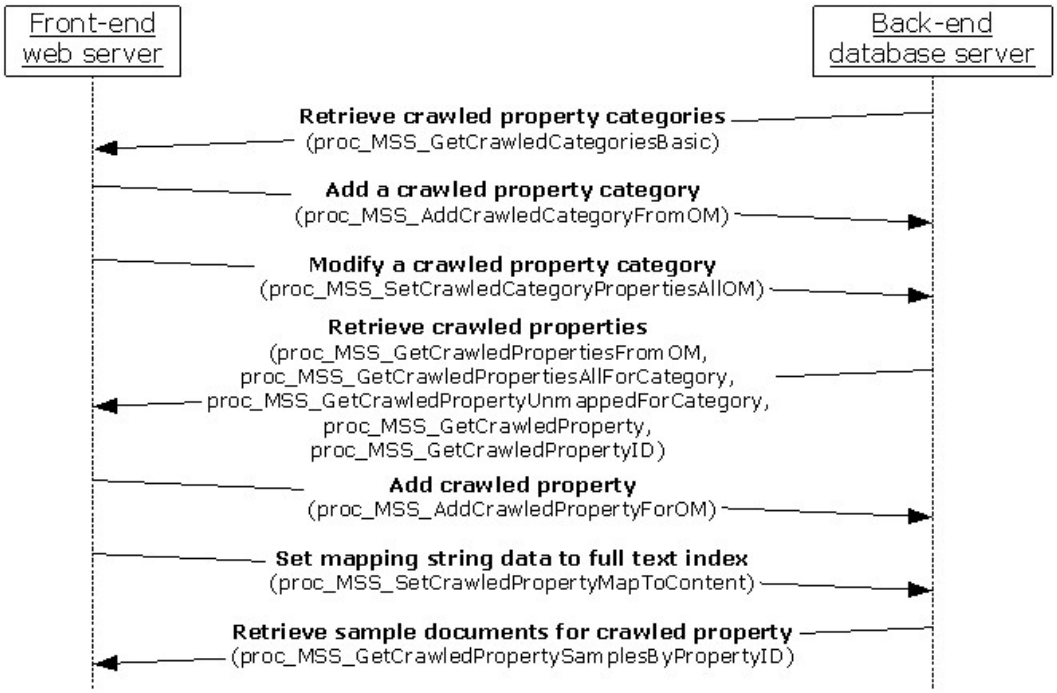

Figure 1: Crawled Property Administration Data Flow Diagram

The protocol allows clients to retrieve a list of all crawled property categories defined in the metadata schema. Clients can add a crawled property category and change its attributes.

All crawled properties can be queried using a filter either within a single crawled property category or over all crawled property categories. The protocol allows clients to add a crawled property to the crawled property category, configure it to map string data to the **full-text index catalog**, as well as retrieve a list of **items** that contain the crawled property.

The following diagram specifies the data flow between the protocol client and the protocol server with regards to administration of managed properties.

The protocol allows clients to retrieve a list of all managed properties defined in the metadata schema. Clients can add a managed property and change its attributes. Clients can also associate a **managed property alias** with a managed property, retrieve a list of all managed property aliases associated with a managed property, or delete the association.

The protocol allows clients to map a **crawled property** to a **managed property**. Clients can retrieve a list of mappings for either a **crawled property** or a **managed property.** When new mappings for a **managed property** are being added, a Pending Mappings Set, as specified in Section 3.1.1.1, is used to hold the uncommitted mappings while they are being assembled. First, pending mappings for the **managed property** are deleted. Then mappings are added one at a time. If the **managed property** is configured to respect the order priority of the **crawled properties**, then the order in which the mappings are added is preserved. The Commit Pending Mappings operation copies the entries in the Pending Mappings Set to the Mappings Set.

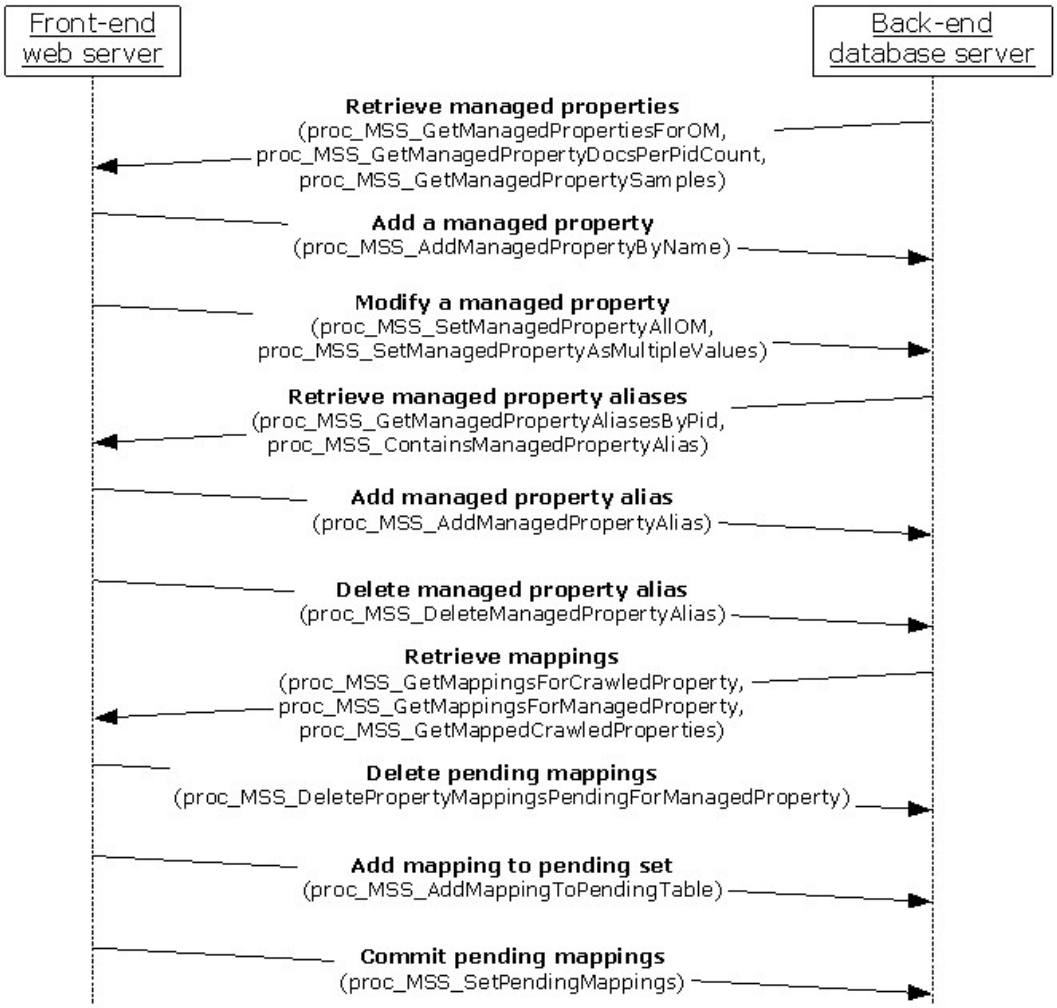

Figure 2: Managed Property Administration Data Flow Diagram

The following diagram specifies the data flow between the protocol client and the protocol server with regards to deletion of **metadata schema** information.

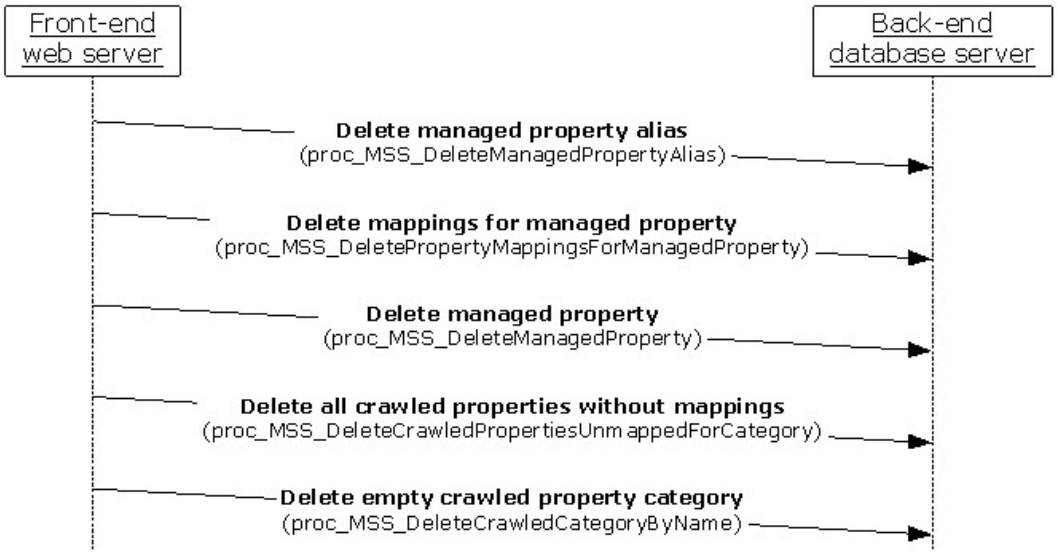

Figure 3: Metadata Schema Removal Data Flow Diagram

The protocol allows clients to delete a **managed property alias** associated with a **managed property.** All the mappings for a **managed property** can be deleted while the **managed properties** and **crawled properties** exist. The protocol allows clients to delete a **managed property** when the **managed property** has no mappings. The set of **crawled properties** within a **crawled property category** that have no mappings and are not mapped to the **full-text index catalog** can all be deleted in one call. Finally, when a **crawled property category** is empty, it can be deleted

1.3.2 Best Bets and Keywords

The following diagram specifies the data flow between the protocol client and the protocol server with regards to administration of **keywords**.

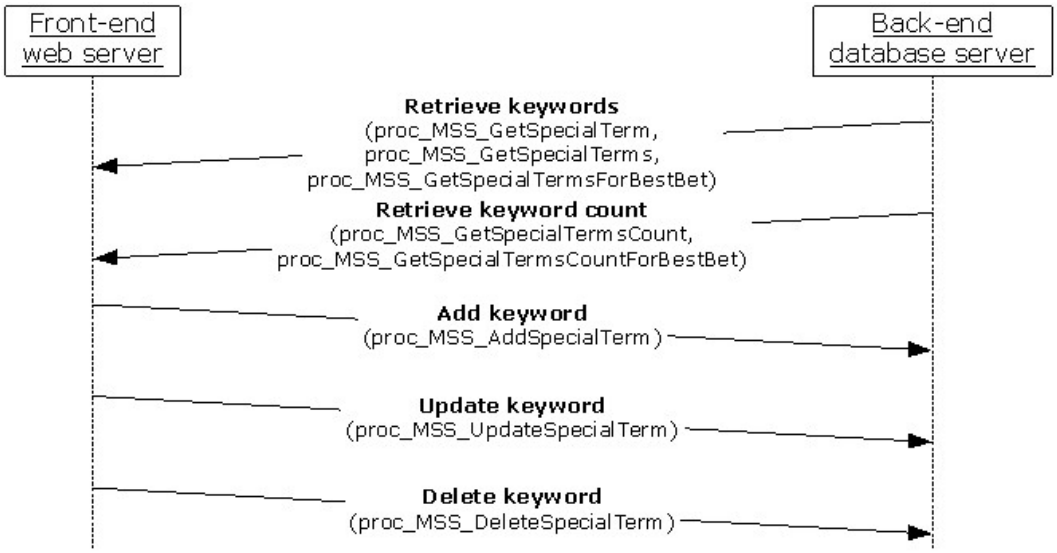

Figure 4: Keywords Administration Data Flow Diagram

The protocol allows clients to add, modify, and delete keywords from a store on the back-end database server. The clients can retrieve the list of keywords and their totals.

The following diagram specifies the data flow between the protocol client and the protocol server with regards to administration of **best bets** .

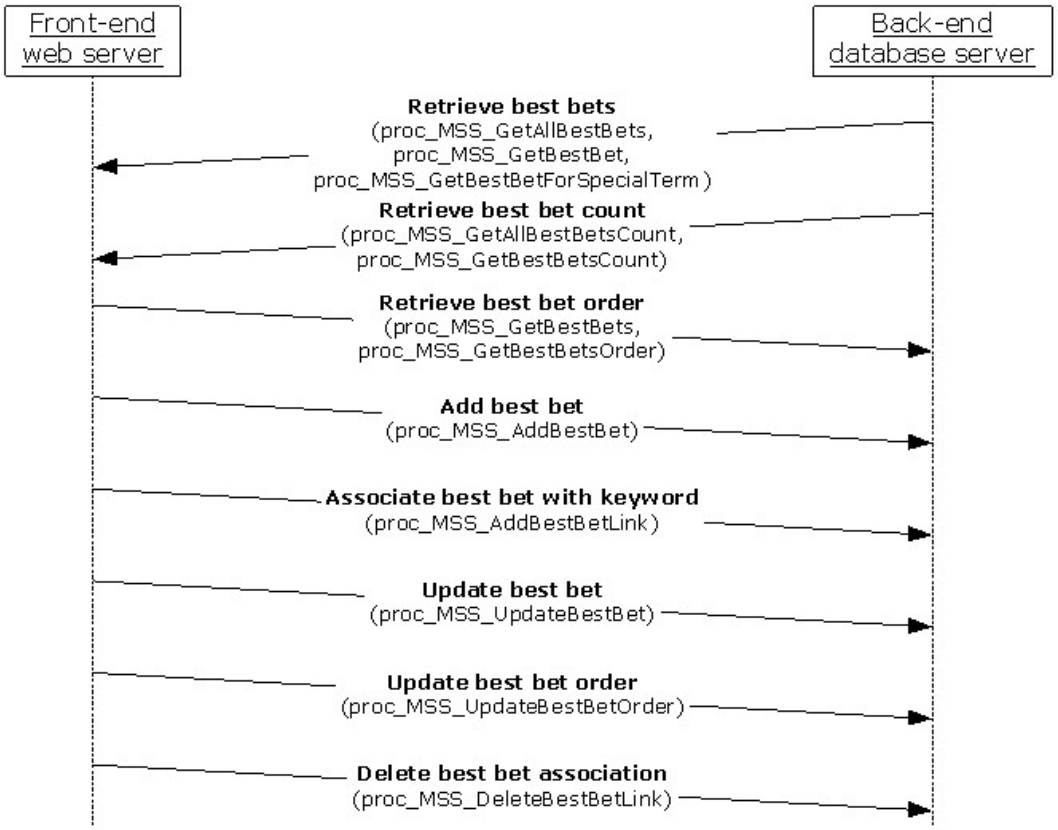

Figure 5: Best Bets Administration Data Flow Diagram

The protocol allows clients to add, change, and delete best bets information from a store on the back-end database server. The protocol allows a keyword to have more than one best bet associated with it, and a best bet can be associated with multiple keywords. The order of this association can be retrieved and updated. Clients configure the best bets list separately from the keywords list. After a best bet is added to this list, it can be associated with the appropriate entries in the keywords list.

The protocol allows clients to retrieve the list of best bets**,** and their totals.

The following diagram specifies the data flow between the protocol client and the protocol server with regards to administration of **keyword synonyms.**

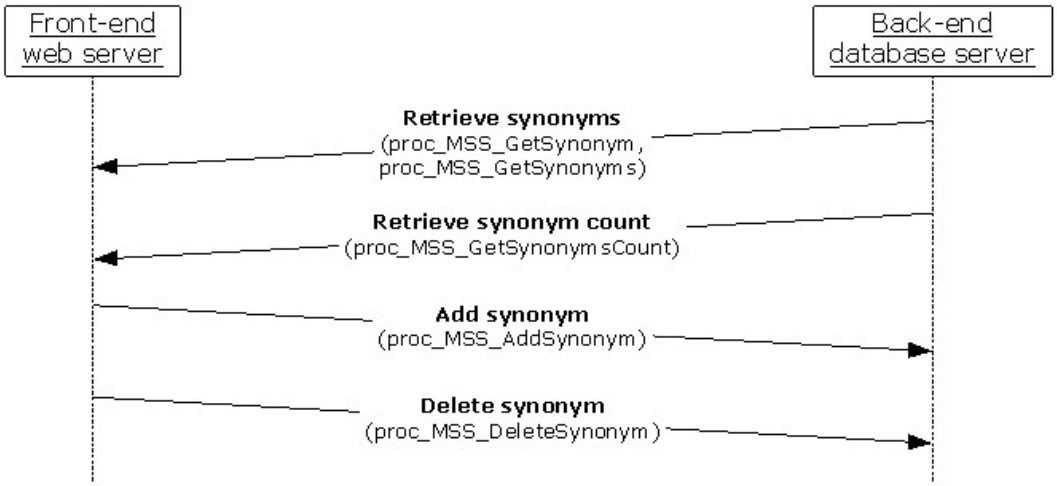

Figure 6: Keyword Synonyms Administration Data Flow Diagram

The protocol allows clients to add, and delete keyword synonyms information from a store on the back-end database server. The protocol allows clients to retrieve the list keyword synonyms and their totals.

1.3.3 Scopes

The protocol allows clients to add, change, retrieve, and delete **search scope** information from a store on the back-end database server. Data flows for the three general types of scope-related tasks are given in this section:

- Consumer-related scope data flow, which is used to manage the relationship between search scopes and **search scope consumers**.
- Scope and rule data flow, which is used to manage the makeup of scopes.
- Display group data flow, which is used to manage how scopes are displayed in the user interface.

The following diagram specifies the consumer-related data flow between the protocol client and the protocol server with regards to administration of search scope consumers.

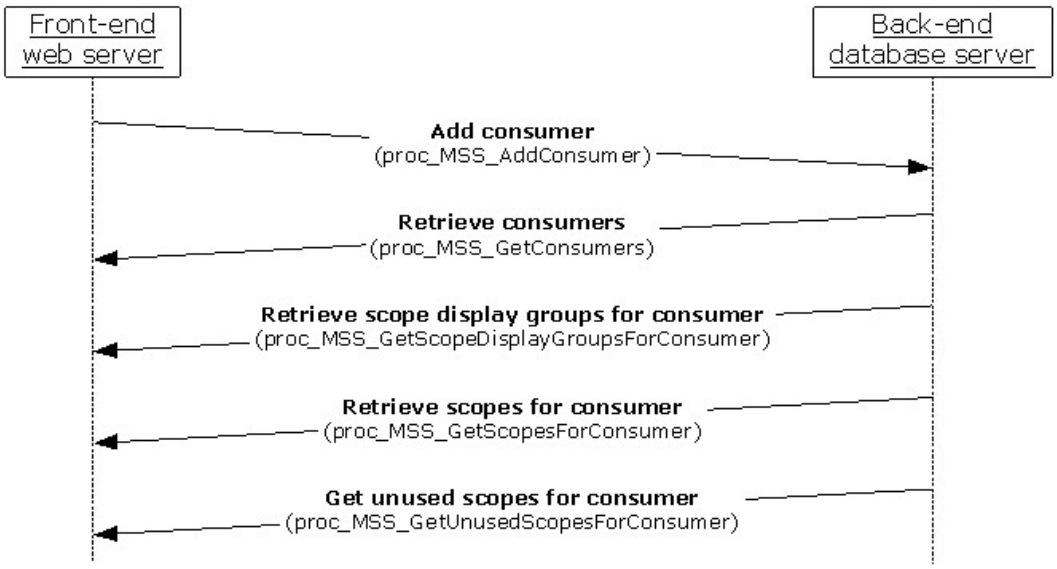

Figure 7: Search Scope Consumer Data Flow Diagram

The protocol allows the client to add a search scope consumer. The client can retrieve the names of all search scope consumers. The **search scope display groups**, used and unused search scopes owned by a search scope consumer, can be listed. The list of search scope consumers who own search scope display groups, search scopes or **search scope rules** of search scopes that have changed since a specific version can be retrieved. Search scopes, search scope display groups, and ranks of compiled search scopes within search scope display groups owned by a specific search scope consumer can be listed.

The following diagram specifies the search scope and rule data flow between the protocol client and the protocol server with regards to administration of search scopes and search scope rules.

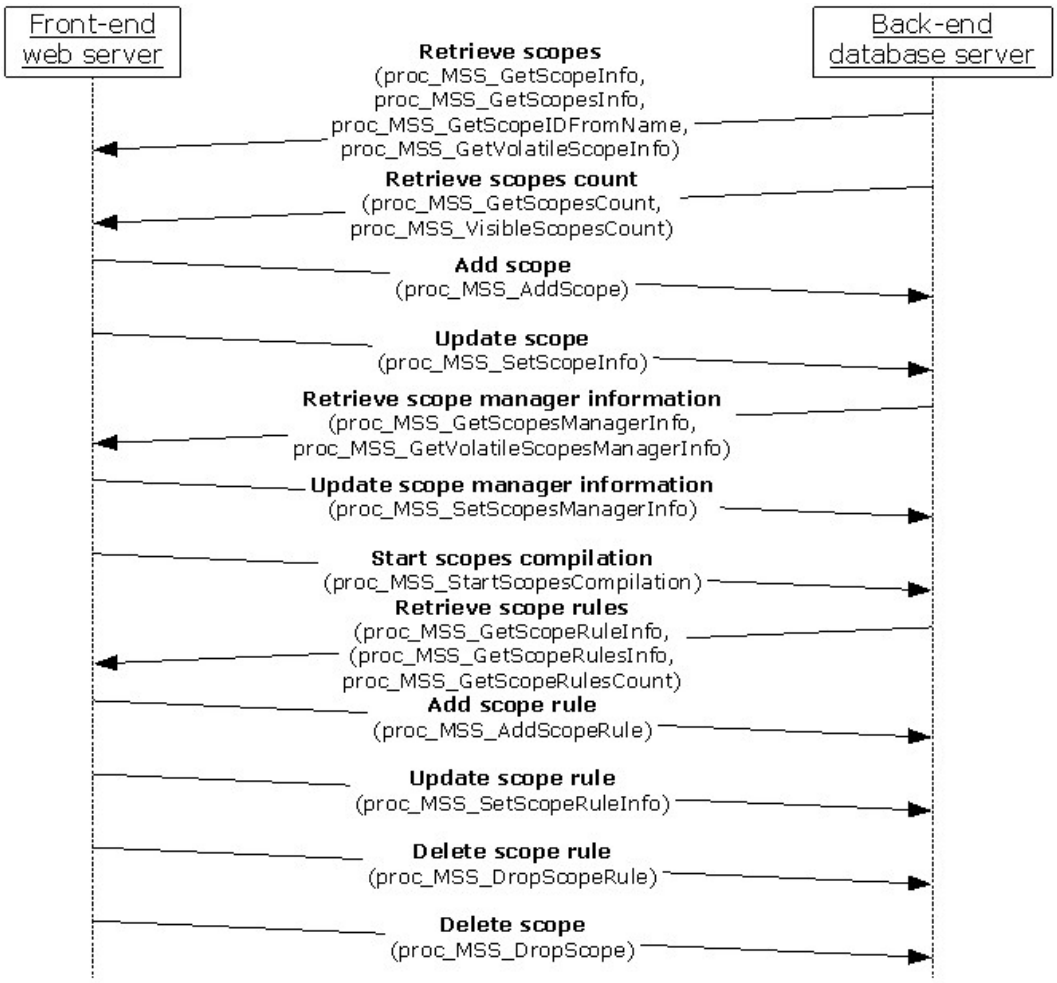

Figure 8: Search Scope and Rule Data Flow Diagram

The protocol allows clients to add, change and delete search scope information from a store on the back-end database server. The protocol clients can retrieve search scopes and their totals. The client can retrieve the status of the search scope, such as the state of the compilation, the time the compilation started, and the percentage of compilation completed.

The protocol allows clients to retrieve and change **search scopes system** information which governs the **search scope compilation** schedule. The type of search scope compilation schedule can be set to either automatic or on-demand update. The compilation of search scopes can be started on-demand even if the search scope compilation schedule is set to automatic, a compilation is currently in progress or the compilation is being stopped.

The protocol allows clients to add, change and delete search scope rule information for the specified search scope. The protocol clients can retrieve search scope rules and their totals.

A search scope rule or a search scope can be deleted.

The following diagram specifies the data flow between the protocol client and the protocol server with regards to administration of search scope display groups.

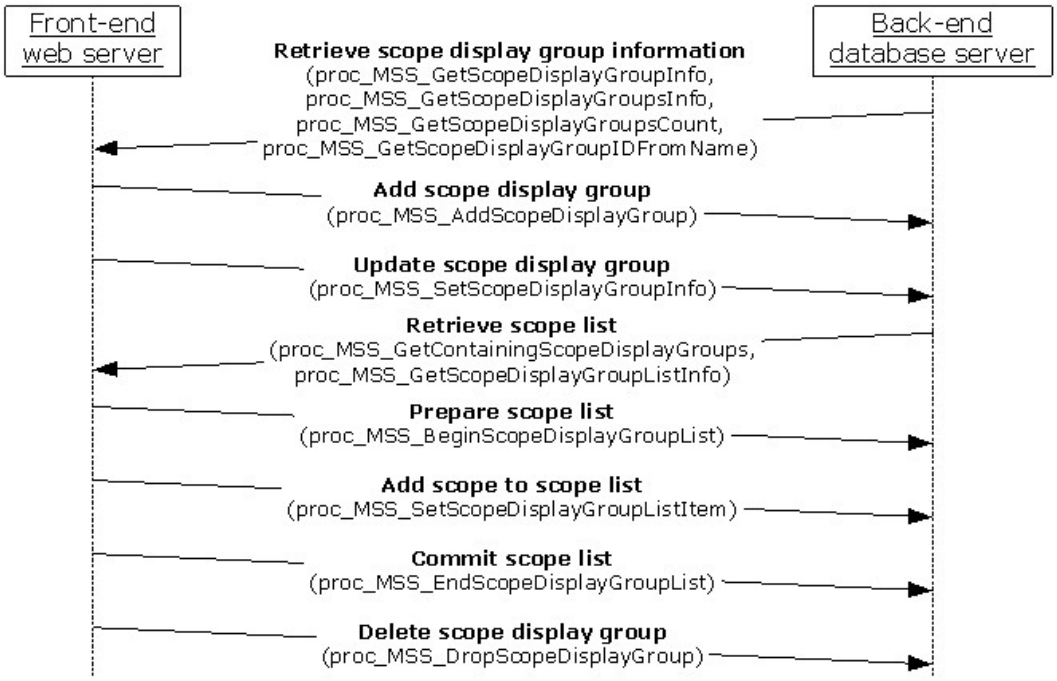

Figure 9: Search Scope Display Group Data Flow Diagram

The protocol allows clients to add, change and delete search scope display group information from a store on the back-end database server. The protocol clients can retrieve search scope display groups information and their totals. The search scopes are grouped together in an ordered set within a search scope display group. A search scopes is added to a search scope display group with the specified order. To do this, the protocol client first prepares the list of search scopes for the search scope display group, then adds each search scope to the list, and finally commits the list. The list is prepared by deleting any search scope for the specified search scope display group that has a negative order value. When the search scope is added to the list, the order value is negative until the search scope list is committed at which point the order value is changed to positive.

1.3.4 Crawl Log

The following diagram specifies the data flow between the protocol client and the protocol server with regards to administration of **the crawl log.**

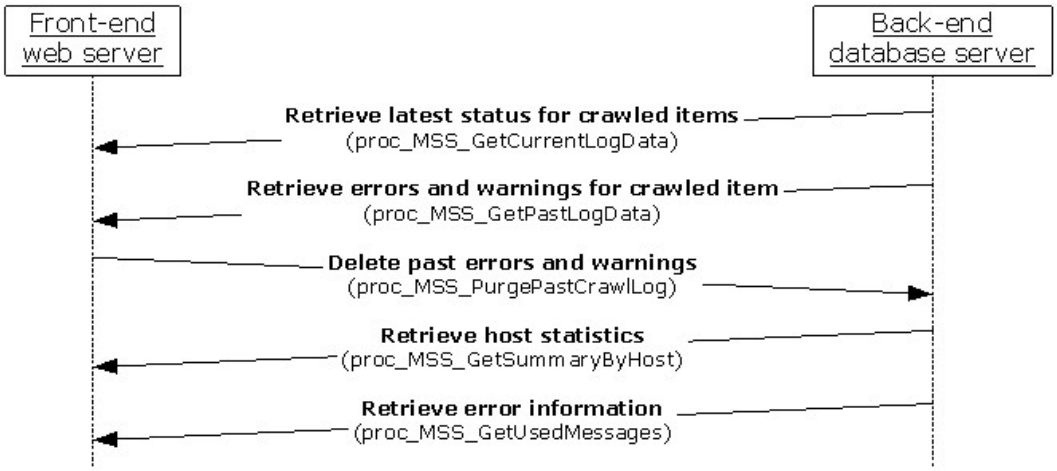

Figure 10: Crawl Log Administration Data Flow Diagram

The protocol allows clients to retrieve successes, errors and warnings for items as of the last time they were **crawl**ed by the **index server** which can be filtered by **display URL**, **content source**, **host name**, error type or date range. The protocol client can also retrieve errors and warnings for the specified item for past crawls. This information about past crawls can be deleted.

The protocol allows clients to retrieve the count of successes, warnings and errors and the total count of items processed by the index server for each host. The client can retrieve the details about errors and warnings.

1.3.5 Child farm Content

The following diagram specifies the data flow between the protocol client and the protocol server with regards to synchronizing the configuration for content sources on the **child farm**.

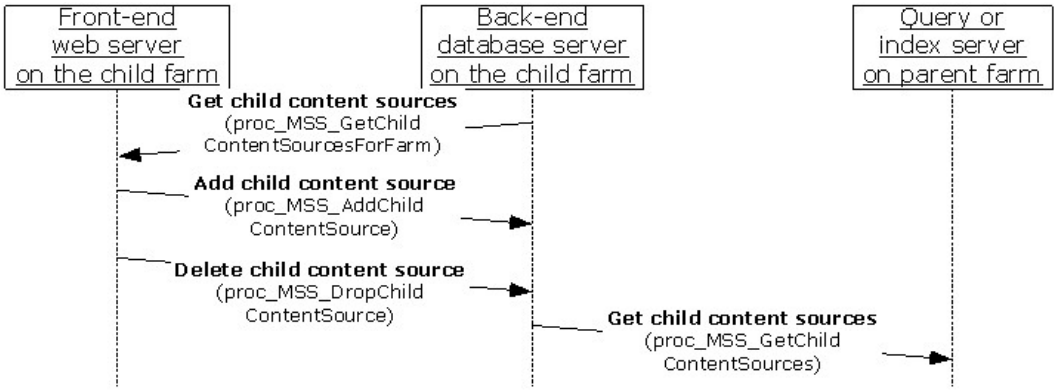

Figure 11: Child farm Content Data Flow Diagram

The protocol allows a front-end Web server on the child farm to retrieve a list of the child farm's **start addresses** from the **parent farm**'s Child Farm Start Addresses Set as specified in Section 3.1.1.4. The front-end Web server compares the child farms current configuration to this list. If a start address for the child farm is not found in the list, it is added to the Child farm Start Addresses Set or if a start address no longer exists, it is removed from the Child Farm Start Addresses Set.

The protocol allows the Child Farm Start Addresses Set to be retrieved from the back-end database server on the parent farm by the index server or **query server**, which uses this list to update the Index server's content sources.

1.3.6 Ranking Parameters

The following diagram specifies the data flow between the protocol client and the protocol server with regards to administration of **ranking parameters**.

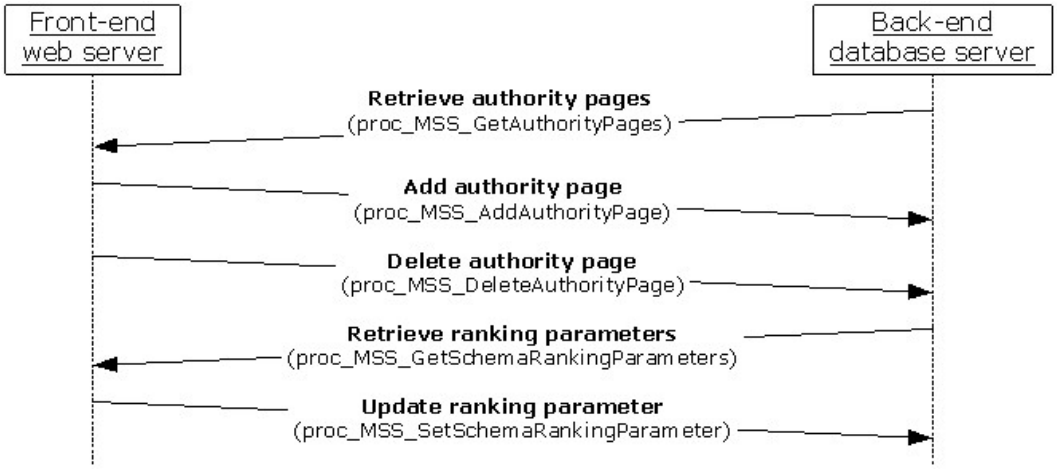

Figure 12: Ranking Parameter Administration Data Flow Diagram

The protocol allows the client to retrieve a list of **authority pages**, to add an authority page, to delete an authority page, to retrieve the set of ranking parameters, and to change their values.

1.3.7 Federated Search

The protocol allows clients to add, modify, retrieve, and delete **federated location** and visualization information from a store on the back-end database server.

The following diagram specifies the data flow between the protocol client and the protocol server with regards to administration of **Federation**.

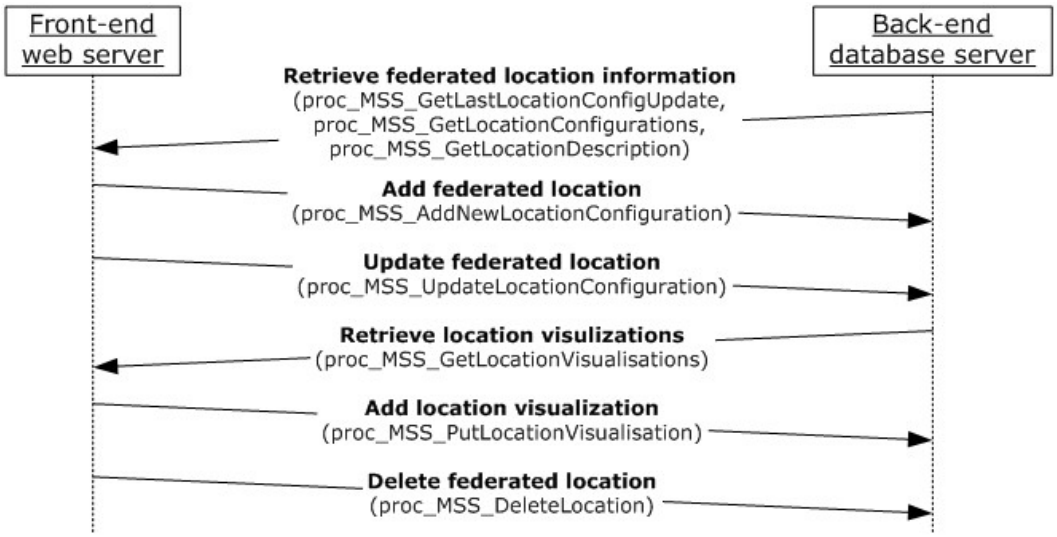

Figure 13: Federated Search Data Flow Diagram

The protocol allows the client to add, change and delete a federated location. The client can add a visualization for the federated location. The protocol allows the client to retrieve the federated location configuration and **visualizations** associated with it. The client can delete the specified federated location which also deletes visualizations associated with it.

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**.

This relationship is illustrated in the following diagram:

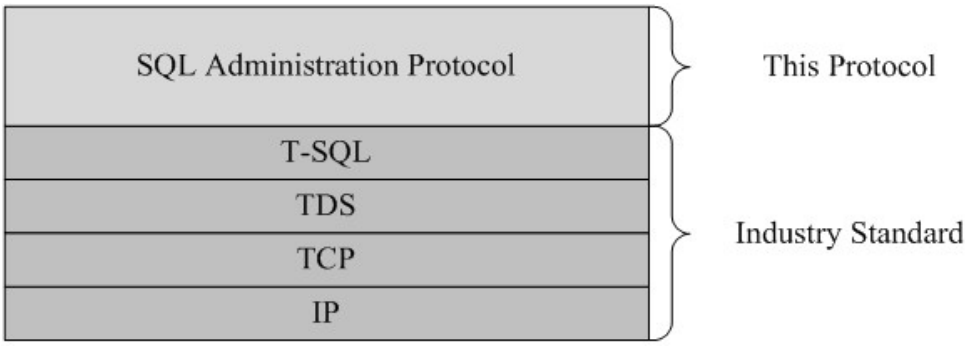

Figure 14: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

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.

1.6 Applicability Statement

This protocol is only applicable to front-end Web server servers when communicating with the back-end database server to satisfy requests for common search crawl tasks as part of Microsoft Office SharePoint Server (MOSS) search administration.

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 front-end Web server 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 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 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, query SQL Views or SQL Tables and return return codes and result sets.

2.2 Common Data Types

This section contains common definitions used by this protocol.

2.2.1 Simple Data Types and Enumerations

2.2.2 Simple Data Types

2.2.2.1 Authentication Type

AuthenticationType specifies the different types of authentication supported for federation. The value MUST be a 8-bit integer listed in the following table.

Value	Description
0	Anonymous
1	Local NT Authentication
2	Single Account Basic Authentication
3	Single Account Form Based Authentication
4	Single Account Cookie Based Authentication
5	Application Pool Identity Based Authentication
7	Kerberos Authentication
8	Per User Form Based Authentication
9	Per User Cookie Based Authentication
10	Single Sign On Authentication
11	Single Account NTLM Based Authentication
12	Per User NTLM Based Authentication
13	Single User Digest Based Authentication
14	Per User Digest Based Authentication
15	Per User Basic Authentication
16	Custom Authentication
17	Per User Custom Authentication

2.2.2.2 Best Bets Filter Type

Best Bets Filter Type specifies which attributes of the best bet are used for filtering the list of best bets**.** The attributes are specified as part of Best Bet Set structure as specified in Section 3.1.1.2. The value MUST be set to an integer listed in the following table.

Value	Description
0	Filter based on Title
1	Filter based on URL
2	Filter based on both Title and URL

2.2.2.3 Compilation Schedule Type

The type of schedule for the search scope compilation. The value MUST be an integer listed in the following table.

Value	Description
0	Search scope compilation only occurs on demand.
1	Search scope compilation occurs automatically on a self-adjusting schedule
2	Search scope compilation occurs based on the schedule specified by the administrator.

2.2.2.4 Compilation State

The state of search scope compilation for a given search scope. The value MUST be an integer listed in the following table.

Value	Description
0	Empty: There are no search scope rules.
1	Invalid: The search scope rules have not included any attributes for compilation.
2	Query Expanded: There are not enough **search scope rules** (fewer than 25) and the compilation rules have been set to conditional compile, so that compilation will not happen.
3	Needs Compilation.
4	Compiled.
5	Needs to be recompiled.

2.2.2.5 Compilation Type

The type of search scope compilation. The value MUST be an integer listed in the following table.

Value	Description
0	Search scope will be compiled on demand.
1	Search scope will always be compiled.

2.2.2.6 DisplayInAdminUI

DisplayInAdminUI indicates whether the search scope display group is displayed in the Administration user interface. The value MUST be an integer listed in the following table.

Value	Description
0	Search scope display group is not displayed in the Administration UI.
1	Search scope display group is displayed in the Administration UI.

2.2.2.7 ErrorLevel

The **error level.** The value MUST be an integer listed in the following table.

Value	Description
0	Success
1	Warning
2	Error

2.2.2.8 Filter wildcard rules

Filter wildcard rules are used for pattern-matching on property names defined in the metadata schema according to the following rules:

- The "%" character is used as a wildcard that can both begin and end the substring.
- "%%" will match all crawled properties.

2.2.2.9 Keyword Type

The type of keywords that the stored procedure operates on or includes in the result set. The attributes are described as part of the Keyword Set definition in Section 3.1.1.2. The value MUST be an integer listed in the following table.

Value	Description
0	All. Include all keywords.
1	Expired. Include keywords with EndDate attribute value earlier than the current date and time.
2	Pending Review. Include keywords with ReviewDate attribute value earlier than the current date and time.

2.2.2.10 Keyword Filter Type

Keyword Filter Type specifies which attributes of the keyword are used for filtering the list of keywords**.** The attributes are described in Section 3.1.1.2. The value MUST be an integer listed in the following table.

Value	Description
0	Apply filter to values in Term attribute in Keywords Set.
1	Apply filter to values in Term attribute in Synonyms Set.
2	Apply filter to values in Title attribute in Best Bet Set for entries associated with the keyword.
3	Apply filter to values in URL attribute in Best Bet Set for entries associated with the keyword.
4	Apply filter to values in Contact attribute in Keywords Set.

2.2.2.11 LastChangeID

Represents the last change made to a search scope or its search scope rules. The search scope is said to have changed in the following scenarios:

- Add / update / delete of the search scope.
- Add / update / delete of a search scope rule for the search scope.

When the LastChangeID value for a search scope is greater than the LastCompilationID value, as specified in Section 2.2.2.12, it indicates that the search scope needs search scope compilation.

2.2.2.12 LastCompilationID

An integer value that is set to the value of the **LastScopeChangeID** data type that represents the **LastChangeID** data type of the latest search scope that will be compiled when search scopes that need search scope compilation are retrieved.

2.2.2.13 LastConsumerChangeID

Represents a version. When any change is made to the search scope display groups, search scopes or search scope rules of any search scope owned by any search scope consumer, this value is incremented by 1, indicating that a search scope consumer has changed. A search scope consumer is said to have changed in the following scenarios:

- Add / update / delete of a search scope that it owns.
- Add / update / delete of a search scope rule of a search scope that it owns.
- Add / update / delete of a search scope rule that it owns.
- Completion of search scope compilation of a search scope that it owns.

The **LastUpdate** data type is incremented every time a specific search scope consumer changes, while the **LastConsumerChangeID** data type is incremented every time a search scope consumer changes. Thus any search scope consumer whose **LastUpdate** data type value is equal to the **LastConsumerChangeID** Data type value is the latest search scope consumer to have changed.

2.2.2.14 LastScopeChangeID

Represents a version. When any change is made to any search scope or search scope rule of any search scope, this value is incremented by 1, indicating that a search scope has changed. A search scope is said to have changed in the following scenarios:

- Add / update / delete of the search scope.

- Add / update / delete of a search scope rule of the search scope.

The **LastChangeID** data type is incremented for a specific search scope every time the specific search scope changes, while the **LastScopeChangeID** data type is incremented every time any search scope changes. Thus any search scope whose **LastChangeID** data type value is equal to the **LastScopeChangeID** data type value is the latest search scope to have changed.

2.2.2.15 LastUpdate

An integer value that is incremented by 1, after which the **LastUpdate** data type of the specific search scope consumer is set to the value of the **LastConsumerChangeID** data type indicating that the specific search scope consumer has changed, when any change is made to the search scope display group, search scopes or search scope rules of the search scopes owned by the specific search scope consumer. The search scope consumer is said to have changed in the following scenarios:

- Add / update / delete of a search scope that it owns.
- Add / update / delete of a search scope rule of a search scope that it owns.
- Add / update / delete of a search scope rule that it owns.
- Completion of search scope compilation of a search scope that it owns.

2.2.2.16 Location Type

Location Type represents which protocol is used to connect to a federated location. The value is an 8-bit integer listed in the following table.

Value	Description
0	Local SharePoint Search
2	Open Search 1.0/1.1. See [OpenSearch1.1].

2.2.2.17 Managed Type

Managed Type identifies the data type of the managed property. The value is an integer listed 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 is TRUE and everything else is FALSE.
6	**Binary large object (BLOB)**.

2.2.2.18 Properties

The list of properties that is returned with each search result, if available, from the federated location.

2.2.2.19 SampleData

It contains the sample data that is used to provide a visual preview of how search results from federated location will look with the given XSL.

2.2.2.20 ScopeFilterBehavior

ScopeFilterBehavior identifies how a search scope rule will filter items. The value MUST be an integer listed 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.

2.2.2.21 ScopeRuleType

ScopeRuleType specifies the type of the search scope rule. The value MUST be an integer listed 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 RuleString attribute value as specified in Section 3.1.1.3.
2	Property Query: The search scope rule includes items whose managed property value matches the RuleString attribute value and whose managed property identifier matches the PropertyId value. RuleString and PropertyId attributes are specified in Section 3.1.1.3.

2.2.2.22 Undeletable

The Undeletable flag specifies if the search scope display group can be deleted. The value MUST be listed in the following table.

Value	Description
0	Search scope display group can be deleted.
1	Search scope display group cannot be deleted.

2.2.2.23 UrlRuleType

UrlRuleType specifies which part of the item **URL is matched against the** RuleString **value as specified in** Section 2.2.2.16. The value MUST be an integer listed in the following table.

Value	Description
0	folder
1	host name
2	subdomain

2.2.2.24 XSL

The XSL that transforms structured XML search results returned by the federated location into HTML and defines how the search results from the federated location will be displayed. For additional information about the the fields, see [MSDN-ESCRXT].

2.2.3 Bit Fields and Flag Structures

None.

2.2.4 Binary Structures

None.

2.2.5 Result Sets

2.2.5.1 Best Bet Result Set

The Best Bet result set contains information about best bets. The result set MUST contain zero or more rows, each corresponding to a single best bet.

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

```
BestBetID               int,
Title                   nvarchar(100),
Url                     nvarchar(2048),
Description             nvarchar(500);
```

BestBetID: The unique identifier of the best bet.

Title: The title for the best bet.

Url: The URL of the best bet

Description: The description of the best bet. This value can be NULL.

2.2.5.2 Crawled Properties Result Set

The Crawled Properties result set contains information about crawled properties associated with the specified crawled property category**.** The result set MUST contain zero or more rows, each corresponding to a single crawled property.

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

```
CategoryName            nvarchar(64),
Propset                 uniqueidentifier,
```

```
PropertyName            nvarchar(440),
PropertyName            IsEnum bit,
IsMappedToContent       bit,
IsSampleCacheFull       bit,
VariantType             int,
CrawledPropertyId       int;
```

CategoryName: The **name of the crawled property associated with the crawled property**. This value MUST NOT be NULL.

Propset: The crawled property set identifier for the **crawled property category**. This value MUST NOT be NULL.

PropertyName: The name of the **crawled property**. This value MUST NOT be NULL.

PropertyNameIsEnum: The value **MUST be set 1 if the** *@PropertyName* **string value was converted from an integer, otherwise 0**.

IsMappedToContent: The value **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.

IsSampleCacheFull: The value **MUST be set to 1 if the Sample Crawled Properties Set is complete. Otherwise, it MUST be set to 0**. For the **Sample Crawled Properties Set** specification see Section 3.1.1.1.

VariantType: The variant type for the crawled property. This value MUST NOT be NULL.

CrawledPropertyId: The unique identifier of **the crawled property.** This value MUST NOT be NULL.

2.2.5.3 Scope Display Groups Result Set

The Scope Display Groups result set contains information about search scope display groups. The result set MUST contain zero or more rows, each row corresponding to a single search scope display group.

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

```
DisplayGroupID          int,
Name                    nvarchar(60),
Description             nvarchar(300),
ConsumerName            nvarchar(60),
DisplayInAdminUI        bit,
Undeletable             bit,
DefaultScopeID          int,
LastModifiedTime        datetime,
LastModifiedBy          nvarchar(60);
```

DisplayGroupID: The unique identifier of the search scope display group. This value MUST NOT be NULL.

Name: The name of the search scope display group. This value MUST NOT be NULL.

Description: The description of the search scope display group. This value MUST NOT be NULL.

ConsumerName: The name of the search scope consumer who is the owner of the search scope display group. This value MUST NOT be NULL.

DisplayInAdminUI: The bit flag indicating whether the search scope display group is displayed in the Administration user interface. The value MUST be a DisplayInAdminUI data type, as specified in Section 2.2.2.5**.**

Undeletable: The bit flag indicating whether the search scope display group can be deleted. The value MUST be an Undeletable data type as specified in Section 2.2.2.22**.**

DefaultScopeID: The unique identifier of the default search scope for the search scope display group. This value MUST NOT be NULL.

LastModifiedTime: The date and time of the last change to the search scope display group. This value MUST NOT be NULL.

LastModifiedBy: The name of the user who last changed the search scope display group. This value MUST NOT be NULL.

2.2.5.4 Scopes Result Set

The Scopes result set contains information about the search scopes. The result set MUST contain zero or more rows, each row corresponding to a single search scope.

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

```
ScopeID                     int,
Name                        nvarchar(60),
Description                 nvarchar(300),
ConsumerName                nvarchar(60),
DisplayInAdminUI            bit,
AlternateResultsPageURL     nvarchar(2048),
CompilationType             smallint,
CompilationState            smallint,
LastCompilationTime         datetime,
LastModifiedTime            datetime,
LastModifiedBy              nvarchar(60);
```

ScopeID: The unique identifier of the search scope.

Name: The name of the search scope.

Description: The description of the search scope.

ConsumerName: The name of the search scope consumer who is the owner of the search scope.

DisplayInAdminUI: The bit flag indicating whether the search scope display group is displayed in the Administration user interface. The value MUST be a DisplayInAdminUI data type as specified in Section 2.2.2.5**.**

AlternateResultsPageUrl: The alternate results page URL of the search scope. This value can be NULL.

CompilationType: The compilation type of the search scope. The value MUST be a CompilationType data type, as specified in Section 2.2.2.5**.**

CompilationState: The search scope compilation state of the given search scope. The value MUST be a CompilationState data type, as specified in Section 2.2.2.4**.**

LastCompilationTime: The date and time when the search scope was last compiled. This value can be NULL.

LastModifiedTime: The date and time of the last change to the search scope.

LastModifiedBy: The name of the user who last changed the search scope.

2.2.5.5 Scope Rules Result Set

The Scope Rules contains information about the search scope rules. The result set MUST contain zero or more rows, each row corresponding to a single search scope rule.

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

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

RuleID: The unique identifier of the search scope rule.

FilterBehavior: The filter behavior of the search scope rule. The value MUST be a ScopeFilterBehavior data type, as specified in Section 2.2.2.20**.**

RuleType: The type of the search scope rule. The value MUST be a ScopeRuleType **data type, as specified in** Section 2.2.2.21**.**

UrlRuleType: The URI type of the search scope rule. The value MUST be an UrlRuleType **data type, as specified in** Section 2.2.2.23**.**

PropertyID: The unique identifier of the managed property used by the search scope rule. This value can be NULL.

UserValueString: The search scope rule value used by the search scope rule. This value can be NULL.

2.2.5.6 Special Term Result Set

The Special Term result set contains information about the keyword. The result set MUST contain zero or more rows, each corresponding to a single keyword.

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

```
SpecialTermId         int,
Term                  nvarchar(100),
Definition            nvarchar(500),
Contact               nvarchar(50),
StartDate             datetime,
EndDate               datetime,
ReviewDate            datetime;
```

SpecialTermId: The unique identifier of the keyword. This value MUST NOT be NULL.

Term: The term for the keyword. This value MUST NOT be NULL.

Definition: The description of the keyword. This value can be NULL.

Contact: The contact name for the keyword. This value can be NULL.

StartDate: The date and time when the keyword begins to appear in search result. This value MUST NOT be NULL.

EndDate: The date and time when the keyword stops appearing in the search result. This value can be NULL.

ReviewDate: The date and time when the keyword is expected to be reviewed. This value can be NULL.

2.2.5.7 Synonym Result Set

The Synonym result set contains information about the keyword synonyms associated with a specified keyword. The result set MUST contain zero or more rows, each corresponding to a single keyword synonym.

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

```
@Term                nvarchar(100);
```

@Term: The term for the keyword synonym. This value MUST NOT be NULL.

2.2.6 Tables and Views

None.

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.

3.1.1.1 Metadata Schema

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

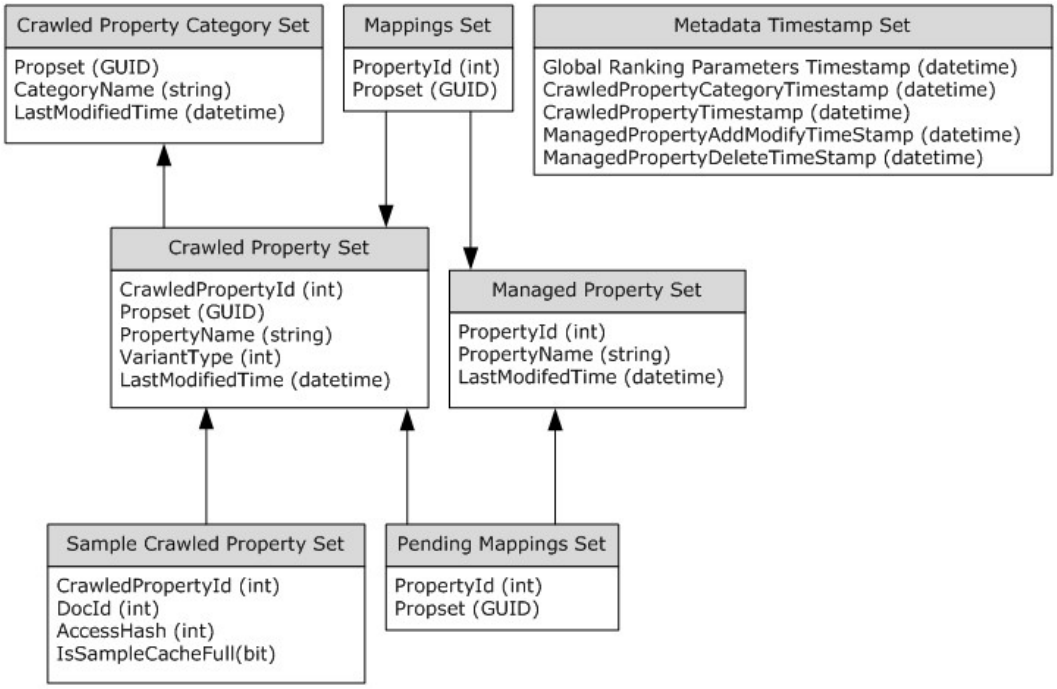

Figure 15: Metadata Schema Abstract Data Model

Crawled Property Category Set: A collection of entries corresponding to the crawled property categories defined in the metadata schema. Each entry MUST be uniquely identified by its Propset, and it MUST include the following elements:

- **Propset**: **The crawled property set identifier associated with the crawled property category**.
- **CategoryName**: The name for **the crawled property category**.

- **LastModifiedTime** : The date and time **the crawled property** was last changed. The value MUST be stored in the local time of the server.
- **Crawled Property Set:** A collection of entries corresponding to the crawled properties defined in the metadata schema. Each entry MUST be uniquely identified by its CrawledPropertyId, and it MUST include the following elements:
- **CrawledPropertyId**: The unique identifier of the **crawled property**.
- **Propset**: The reference to the corresponding crawled property category.
- **PropertyName**: The name for **the crawled property**.
- **VariantType**: T**he** variant type for the crawled property.
- **LastModifiedTime** : The date and time **the crawled property** was last changed. The value MUST be stored in the local time of the server.

Managed Property Set: A collection of entries corresponding to the managed properties defined in the metadata schema**.** Each entry MUST be uniquely identified by its PropertyId, and it MUST include the following elements:

- **PropertyId**: The unique identifier of the **managed property**.
- **PropertyName**: The name for **the managed property**.
- **LastModifiedTime**: The date and time **the managed property** was last changed. The value MUST be stored in the local time of the server.

Mappings Set: A collection of entries each describing the mapping of crawled property to managed property**.** Each entry MUST include the following elements:

- **PropertyId**: The reference to the **managed property**.
- **Propset**: The reference to the crawled property category.
- **Pending Mappings Set:** A collection of entries each describing the mapping of crawled property to managed property that is not yet persisted in the Mappings Set. Each entry MUST include the following elements:
- **PropertyId**: The reference to the **managed property**.
- **Propset**: The reference to the crawled property category.

Sample Crawled Property Set: A collection of relationships between crawled properties and sample items that contain them. Each entry MUST be uniquely identified by the combination of its CrawledPropertyId and DocId. Each entry MUST include the following elements:

- **CrawledPropertyId**: The reference to the **crawled property**.
- **DocId**: The unique identifier of the item.
- **AccessHash**: The identifier of the item access URL.
- **IsSampleCacheFull**: A bit flag indicating whether the Sample Crawled Properties Set is complete and no more entries needed for the crawled property.

Metadata Timestamp Set: Used to track date and time of the last modification to **the** metadata schema entities. Metadata Timestamp Set MUST include exactly one entry. The entry MUST include the following elements:

- **Global Ranking Parameters Timestamp**: The date and time of the last change to the Global Ranking Parameters Set as specified in Section 3.1.1.5.
- **CrawledPropertyCategoryTimestamp**: The date and time of the last change to the **Crawled Property Category Set**. The value MUST be stored in the local time of the server.
- **CrawledPropertyTimestamp**: The date and time of the last change to the **Crawled Property Set**. The value MUST be stored in the local time of the server.
- **ManagedPropertyAddModifyTimestamp**: The date and time when an entry in the **Managed Property Set was last added or changed**. The value MUST be stored in the local time of the server.
- **ManagedPropertyDeleteTimestamp**: The date and time when an entry in the **Managed Property Set was last deleted**. The value MUST be stored in the local time of the server.

3.1.1.2 Best Bets and Keywords

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

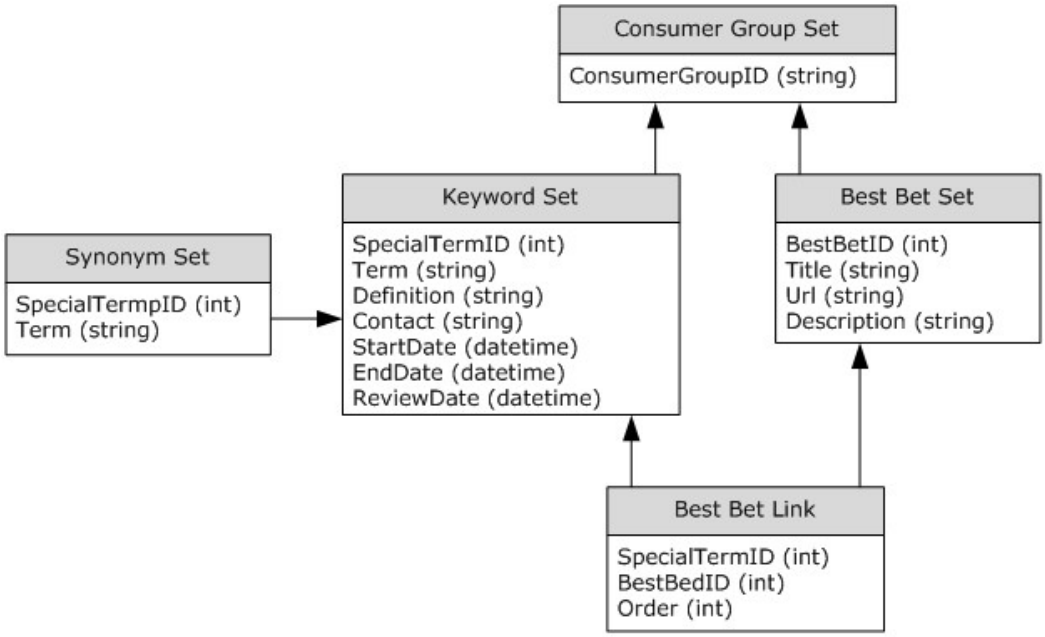

Figure 16: Best Bets and Keywords Abstract Data Model

The protocol server stores all implementation-specific information about best bets and keywords in the following sets of data:

Consumer Group Set: A collection of entries corresponding to **keyword consumers**. Each entry MUST be uniquely identified by its ConsumerGroupID and it MUST include the following elements:

- ConsumerGroupID: The unique identifier of the keyword consumer.

Keyword Set: A collection of entries representing keywords defined within a **site collection**. There is a many-to-many relationship between **keywords** and **best bets**. A **keyword** can have more than one **best bet** associated with it, and a **best bet** can be associated with multiple **keywords**. Each entry MUST be uniquely identified by its SpecialTermId, and it MUST include the following elements:

- **SpecialTermId**: The unique identifier of the **keyword.**
- **Term**: The term for the **keyword.**
- **Definition**: The definition of the **keyword.**
- **StartDate:** The date and time when the keyword begins to appear in search results.
- **Contact:** The contact name for the keyword.
- **EndDate:** The date and time when the keyword stops appearing in search result.
- **ReviewDate:** The date and time when the keyword is expected to be reviewed.

Best Bet Set: A collection of entries representing best bets defined within a site collection. Each entry MUST be uniquely identified by its BestBetId, and it MUST include the following elements:

- **BestBetId**: The unique identifier of the **best bet.**
- **Title**: The title of the **best bet.**
- **Url**: The URL for the **best bet.**
- **Description**: The description of the **best bet.**

Synonym Set: A collection of entries representing keyword synonyms associated with keywords. Each entry MUST include the following elements:

- **SpecialTermId**: The unique identifier of the **keyword** associated with the **keyword synonym.**
- **Term**: The term of the **keyword synonym.**

3.1.1.3 Scopes

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

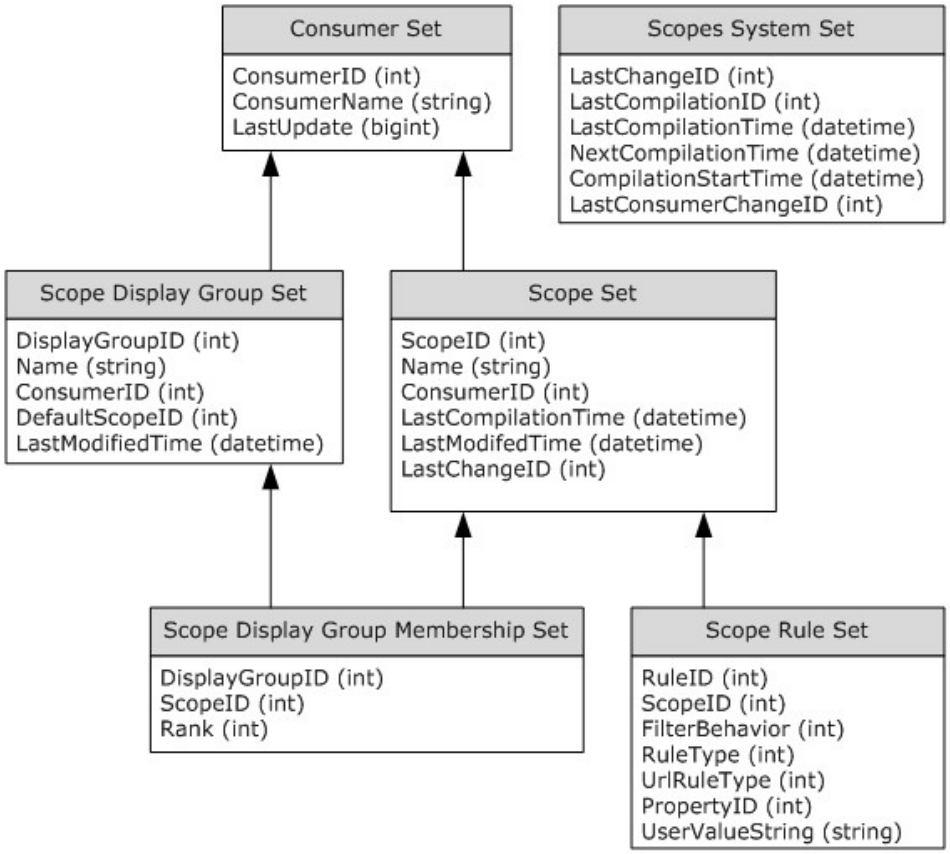

Figure 17: Search Scopes Abstract Data Model

Consumer Set: A collection of entries corresponding to search scope consumers. Each entry MUST be uniquely identified by its ConsumerID and it MUST include the following elements:

- **ConsumerID:** The unique identifier of the search scope consumer.
- **ConsumeName:** The name for the search scope consumer.
- **LastUpdate:** The version which MUST be a LastUpdate data type as specified in Section 2.2.2.15.

Scope Set: A collection of entries corresponding to search scopes. Each entry MUST be uniquely identified by its ScopeID and it MUST include the following elements:

- **ScopeID:** The unique identifier of the search scope.
- **Name:** The name for the search scope.
- **ConsumerID:** The reference to the corresponding search scope consumer.
- **LastCompilationTime:** The date and time when the search scope was last compiled. The value MUST be stored in the local time of the server.
- **LastModifiedTime:** The date and time when the search scope was last changed. The value MUST be stored in the local time of the server.

- **LastChangeID**: The version which MUST be a LastChangeID data type as specified in Section 2.2.2.11.

Scope Rule Set: A collection of entries corresponding to search scope rules Each entry MUST be uniquely identified by its RuleID and it MUST include the following elements:

- **RuleID:** The unique identifier of the search scope rule.
- **ScopeID:** The reference to the corresponding search scope.
- **FilterBehavior**: The value MUST be a ScopeFilterBehavior data type as specified in Section 2.2.2.20.
- **RuleType**: The type of search scope rule which MUST be a ScopeRuleType data as specified in Section 2.2.2.21.
- **UrlRuleType**: The type of URL rule which MUST be a UrlRuleType data as specified in Section 2.2.2.23.
- **PropertyID**: The property ID, which is a reference to a managed property.
- **UserValueString**: **The** search scope rule value to use in the search scope rule.

Scope Display Group Set: A collection of entries corresponding to search scope display groups. Each entry MUST be uniquely identified by its DisplayGroupID and it MUST include the following elements:

- **DisplayGroupID:** The unique identifier of the search scope display group.
- **Name:** The name for the search scope display group.
- **ConsumerID:** The reference to the corresponding search scope consumer.
- **DefaultScopeID:** The reference to the corresponding default search scope.
- **LastModifiedTime:** The date and time when the search scope display group was last changed. The value MUST be stored in the local time of the server.

Scope Display Group Membership Set: A collection of relationships between search scope display groups and the search scopes they contain. Each entry MUST be uniquely identified by the combination of its DisplayGroupID and ScopeID. Each entry MUST include the following elements:

- **DisplayGroupID:** The reference to the search scope display group.
- **ScopeID:** The reference to the search scope.
- **Rank:** The identifier of the search scope rank within the search scope display group.

Scopes System Set: Used to track date and time and version of the last change to search scope entities. Scopes System Set MUST include exactly one entry. The entry MUST include the following elements:

- **LastChangeID:** The version which MUST be a LastScopeChangeID data type as specified in Section 2.2.2.13.
- **LastCompilationID:** The value MUST be a LastCompilationID data type as specified in Section 2.2.2.11.

- **LastCompilationTime:** The date and time when search scopes in the Scope Set were last compiled. The value MUST be stored in the local time of the server.
- **NextCompilationTime:** The date and time when search scopes in the Scope Set will be compiled next. The value MUST be stored in the local time of the server.
- **CompilationStartTime: The date and time when the compilation of search scopes was started for the current compilation. The value MUST be stored in the local time of the server.**
- **LastConsumerChangeID:** The version which MUST be a LastConsumerChangeID data type as specified in Section 2.2.2.12.

3.1.1.4 Child farm Content

The Child farm Start addresses Set is a list of start addresses for the child farms which is persisted on the parent farm. Each entry MUST include the following elements:

- Name of the child farm to which this start address belongs.
- The unique identifier of the start address.
- The start address of content from the child farm.

3.1.1.5 Ranking Parameters

The Authority Pages Set is a list of authority pages which are used in page ranking. Each entry MUST include the following elements:

- **The valid URL of the** authority page.
- **The authority level** of the authority page.

The Global Ranking Parameters Set is a list of name/value pairs for variables used in ranking. Each entry MUST include the following elements:

- The name of the variable
- The value for the variable.

3.1.1.6 Federated Search

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

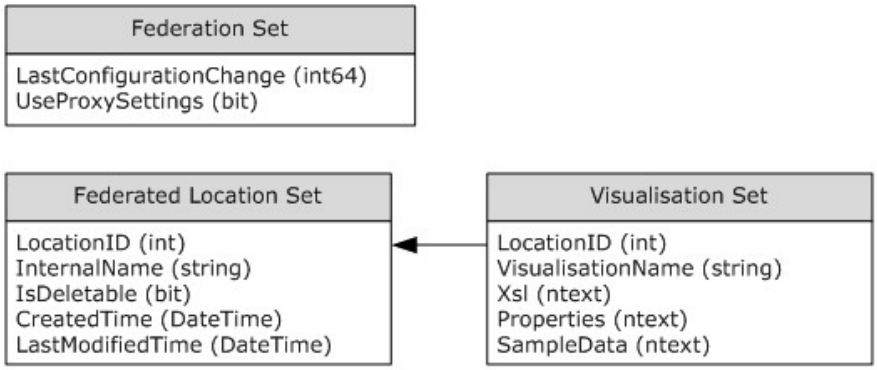

Figure 18: Federated Search Data Flow Diagram

Federation Set: Used to track date and time and version of the last change to federated location and visualization entities. The Federation Set MUST include exactly one entry. The entry MUST include the following elements:

- LastConfigurationChange: The version.
- UseProxySettings: A 1-bit number that indicates whether federated locations are configured to use a **proxy** when retrieving search results.

Federated Location Set: A collection of entries corresponding to Federated locations. Each entry MUST be uniquely identified by its LocationID and it MUST include the following elements:

- **LocationID:** The unique identifier of the federated location.
- **InternalName:** The unique internal name of the federated location.
- **IsDeletable: A** flag that indicates that the federated location and the visualizations associated with it can be deleted.
- **CreatedTime:** The UTC **datetime** when the federated location was created.
- **LastModifiedTime:** The UTC datetime when the federated location was last modified.

Federated Location Set: A collection of entries corresponding to Federated locations. Each entry MUST be uniquely identified by its LocationID and it MUST include the following elements:

- **LocationID:** The unique identifier of the federated location.
- **VisualisationIName:** The name of the visualization.
- **Xsl:** The Xsl for this visualization. This MUST be an **Xsl** Data Type as specified in Section 2.2.2.24.
- **Properties:** The properties for this visualization. This MUST be a Properties Data Type as specified in Section 2.2.2.18.
- **SampleData:** The sample data for this visualization. This MUST be a SampleData Data Type as specified in Section 2.2.2.19.

3.1.2 Timers

None.

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

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

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

Procedure Name	Description
proc_MSS_AddAuthorityPage	Creates a new authority page
proc_MSS_AddBestBet	Creates a new best bet and associates it with a given keyword.
proc_MSS_AddBestBetLink	Adds an association between a best bet and a keyword.
proc_MSS_AddChildContentSource	Adds a start address to the Child farm Start addresses Set as specified in Section 3.1.1.4.
proc_MSS_AddConsumer	Adds a search scope consumer to the list of consumers in order to view, add, modify or delete search scopes and search scope display groups within it.
proc_MSS_AddCrawledPropertyCategoryFromOM	Adds a crawled property to the

Procedure Name	Description
	Crawled Property Category Set, specified in Section 3.1.1.1.
proc_MSS_AddCrawledPropertyForOM	Adds a crawled property to the Crawled Property Set, specified in Section 3.1.1.1.
proc_MSS_AddManagedPropertyAlias	Adds another name which is associated with the managed property to the metadata schema.
proc_MSS_AddManagedPropertyByName	Adds a custom managed property to the Managed Property Set, specified in Section 3.1.1.1.
proc_MSS_AddMappingToPendingTable	Adds a mapping between a crawled property and a managed property into the Pending Mappings Set as the lowest priority mapping.
proc_MSS_AddNewLocationConfiguration<1>	Adds a new federated location.
proc_MSS_AddScope	Adds a new search scope to the **search application**
proc_MSS_AddScopeDisplayGroup	Adds a search scope display group for a search scope consumer.
proc_MSS_AddScopeRule	Adds a new search scope rule to an existing search scope in the search application**.**
proc_MSS_AddSpecialTerm	Adds a new keyword to the site collection.
proc_MSS_AddSynonym	Associates a synonym with a given keyword.
proc_MSS_BeginScopeDisplayGroupList	Deletes all search scopes with negative order from the specified search scope display group
proc_MSS_Cleanup	Clears out old data from the search database.
proc_MSS_CleanupPropertyTables	Fixes the **metadata index** when it has data that is out of sync and isn't being corrected by crawls.
proc_MSS_ContainsManagedPropertyAlias	Determines whether an managed property alias is defined in the metadata schema.

Procedure Name	Description
proc_MSS_DeleteAuthorityPage	Deletes an authority page.
proc_MSS_DeleteBestBetLink	Deletes an association between a keyword and a best bet.
proc_MSS_DeleteCrawledCategoryByName	Deletes a crawled property from the Crawled Property Category Set, specified in Section 3.1.1.1.
proc_MSS_DeleteCrawledPropertiesUnmappedForCategory	Removes the set of crawled properties from the Crawled Property Set , specified in Section 3.1.1.1, which conform to the following criteria: ▪ The crawled property belongs to the same crawled property set identifier as the named crawled property category ▪ The crawled properties data is not put in the full-text index catalog. ▪ The crawled property is not mapped to any managed property.
proc_MSS_DeleteLocation<2>	Deletes a federated location and the visualizations that are associated with it.
proc_MSS_DeleteManagedProperty	Deletes a managed property from the Managed Property Set, specified in Section 3.1.1.1.
proc_MSS_DeleteManagedPropertyAlias	Deletes a managed property alias from the metadata schema.
proc_MSS_DeletePropertyMappingsForManagedProperty	Deletes the crawled property mappings to this managed property from the Mappings Set, specified in Section 3.1.1.1.
proc_MSS_DeletePropertyMappingsPendingForManagedProperty	Deletes the crawled property mappings to this managed property from the Pending Mappings Set, specified in Section 3.1.1.1.
proc_MSS_DeleteSpecialTerm	Deletes a keyword from the site collection.

Procedure Name	Description
proc_MSS_DeleteSynonym	Deletes a synonym of a given keyword from the site collection.
proc_MSS_DropChildContentSource	Deletes a start address in the Child farm Start addresses Set, specified in Section 3.1.1.4.
proc_MSS_DropScope	Deletes an existing search scope from the search application.
proc_MSS_DropScopeDisplayGroup	Deletes a search scope display group
proc_MSS_DropScopeRule	Deletes an existing search scope rule from the search application.
proc_MSS_EndScopeDisplayGroupList	Removes all the search scopes from the specified search scope display group that have positive order and change the negative order of the search scopes contained in the specified search scope display group to its positive value - 1.
proc_MSS_GetAllBestBets	Retrieves all best bets that belong to the specified **keyword consumer group**.
proc_MSS_GetAllBestBetsCount	Calculates the number of best bets for the specified keyword consumer group that match the specified filtering criteria.
proc_MSS_GetAuthorityPages	Lists the authority pages with their authority level.
proc_MSS_GetBestBet	Retrieves a best bet given its URL for the specified keyword consumer group.
proc_MSS_GetBestBetForSpecialTerm	Retrieves the information about a best bet for the specified keyword consumer group, URL, and keyword.
proc_MSS_GetBestBets	Retrieves the list of best bets for the specified keyword.
proc_MSS_GetBestBetsCount	Calculates the number of best bets for a given keyword.
proc_MSS_GetBestBetsOrder	Retrieves a list of best bets associated with a given keyword and the information about their priorities within this

Procedure Name	Description
	list.
proc_MSS_GetChildContentSources	Lists all entries from the Child farm Start addresses Set, specified in Section 3.1.1.4.
proc_MSS_GetChildContentSourcesForFarm	Lists start addresses for a given child farm from the Child farm Start addresses Set, specified in Section 3.1.1.4
proc_MSS_GetConsumers	Retrieves names of all search scope consumers
proc_MSS_GetContainingScopeDisplayGroups	Retrieves a list of all search scope display groups that contain a specified search scope.
proc_MSS_GetCrawledProperty	Retrieves a crawled property from the metadata schema.
proc_MSS_GetCrawledPropertyID	Retrieves the identifier of the specified crawled property from the metadata schema.
proc_MSS_GetCrawledPropertiesAllForCategory	Retrieves a list of crawled properties associated with a crawled property from the metadata schema.
proc_MSS_GetCrawledPropertiesForOM	Retrieves a list of crawled properties associated with a crawled property from the metadata schema.
proc_MSS_GetCrawledPropertiesBasic	Retrieves a list of crawled properties from the metadata schema **which is ordered by** *@Propset* **and then by** *@PropertyName.*
proc_MSS_GetCrawledPropertyCategoriesBasic	Lists all **crawled property categories** from the **metadata schema.**
proc_MSS_GetCrawledPropertiesamplesByPropertyID	Retrieves the list of items associated with the specified of crawled property from the Sample Crawled Property Set, specified in Section 3.1.1.1.
proc_MSS_GetCrawledPropertiesUnmappedForCategory	Retrieves crawled properties which have no mappings to the full-text index catalog or the metadata index in the metadata schema.
proc_MSS_GetCrawlHistory<3>	Retrieves information about

Procedure Name	Description
	crawls.
proc_MSS_GetCurrentLogData	Retrieves the count and the list of successes, warnings and errors logged for display URLs when they were processed by the **crawler**.
proc_MSS_GetLastLocationConfigUpdate<4>	Retrieves the LastConfigurationChange of the Federation Set as specified in Section 3.1.6.
proc_MSS_GetLocationConfigurations<5>	Retrieves configuration information for all the federated locations.
proc_MSS_GetLocationDescription<6>	Retrieves the **federated location definition**.
proc_MSS_GetLocationVisualisations<7>	Retrieves the specified visualization or all visualizations of the specified federated location.
proc_MSS_GetManagedPropertiesForOM	Retrieves a list of managed properties from the **Managed Properties Set ,** specified in Section 3.1.1.1, **which were** added or modified on or after the date and time specified by *@ldate* parameter.
proc_MSS_GetManagedPropertyAliasesByPid	Lists the aliases for a managed property from the metadata schema.
proc_MSS_GetManagedPropertyDocsPerPidCount	Retrieves the count of items in the metadata index which have a specified managed property.
proc_MSS_GetManagedPropertySamples	Lists sample values for a managed property from the metadata schema.
proc_MSS_GetMappedCrawledProperties	Retrieves a list of crawled properties mapped to a managed property in the metadata schema **Mappings Set,** specified in Section 3.1.1.1.
proc_MSS_GetMappingsForCrawledProperty	Retrieves a list of managed properties mapped to a crawled property in the metadata schema **Mappings Set,** specified in Section 3.1.1.1.
proc_MSS_GetMappingsForMangedProperty	Retrieves a list of crawled

Procedure Name	Description
	properties mapped to a managed property in the metadata schema **Mappings Set,** specified in Section 3.1.1.1.
proc_MSS_GetNDayAvgCrawlHistoryStats<8>	Retrieves the statistics for completed crawls
proc_MSS_GetPastLogData	Retrieves the list of errors and warnings logged for a specific display URL when the display URL was processed by the crawler.
proc_MSS_GetSchemaRankingParameters	Retrieves a list of parameters used in ranking **query results**.
proc_MSS_GetScopeDisplayGroupIDFromName	Retrieves the identifier of a search scope display group with the name of the search scope display group and the search scope consumer name.
proc_MSS_GetScopeDisplayGroupInfo	Retrieves information about a search scope display group.
proc_MSS_GetScopeDisplayGroupListInfo	Retrieves a list of all search scopes within the specified search scope display group.
proc_MSS_GetScopeDisplayGroupsCount	Retrieves the total number of search scope display groups.
proc_MSS_GetScopeDisplayGroupsForConsumer	Retrieves a list of all search scope display groups owned by the specified search scope consumer.
proc_MSS_GetScopeDisplayGroupsInfo	Retrieves a list of all search scope display groups.
proc_MSS_GetScopesForConsumer	Retrieves the list of search scopes that are owned by the specified search scope consumer.
proc_MSS_GetScopeIDFromName	Retrieves the identifier of a search scope when given the name of the search scope and the name of the search scope consumer that owns the search scope.
proc_MSS_GetScopeInfo	Retrieves the information about a search scope given its identifier.
proc_MSS_GetScopeRuleInfo	Retrieves the information about a search scope rule given its

Procedure Name	Description
	identifier.
proc_MSS_GetScopeRulesCount	Retrieves the count of search scope rules defined for a search scope in the search application.
proc_MSS_GetScopeRulesInfo	Retrieves the details of all the search scope rules defined for a search scope in the search application.
proc_MSS_GetScopesCount	Retrieves the count of all search scopes defined in the search application.
proc_MSS_GetScopesInfo	Retrieves the details of all the search scopes defined in the search application.
proc_MSS_GetScopesManagerInfo	Retrieves the details of the search scopes system in the search application.
proc_MSS_GetSharepointLocationVisualisations<9>	Retrieves the preselected visualizations of a federated location.
proc_MSS_GetSummaryLogData<10>	Retrieves the count of successes, warnings and errors logged by the crawler.
proc_MSS_GetSynonym	Retrieves the information about the synonym associated with a given keyword.
proc_MSS_GetSynonyms	Retrieves the list of synonyms associated with a given keyword.
proc_MSS_GetSynonymsCount	Calculates the number of synonyms associated with a given keyword.
proc_MSS_GetUnusedScopesForConsumer	Retrieves the information of all the Search scopes owned by a search scope consumer that are NOT in a search scope display group.
proc_MSS_GetUsedMessages	Retrieve the description of unique successes, warnings and errors encountered by the crawler.
proc_MSS_GetVisibleScopesCount	Retrieves the count of all **visible scopes** defined in the search application.
proc_MSS_GetVolatileScopeInfo	Retrieves the details likely to change for a search scope with

Procedure Name	Description
	the specified identifier.
proc_MSS_GetVolatileScopesManagerInfo	Retrieves the details likely to change for a search scopes system in the search application.
proc_MSS_PurgePastCrawlLog	Deletes all the warnings and errors logged for all the display URLs when they were processed by the crawler.
proc_MSS_PutLocationVisualisation<11>	Adds a new visualization definition for a federated location.
proc_MSS_RemoveFilenameFromResults	Removes a file from the database so it will not appear in the query results.
proc_MSS_SetCrawledCategoryPropertiesAllOM	Changes data for the specified **crawled property** in the **metadata schema.**
proc_MSS_SetCrawledPropertyMapToContent	Changes IsMappedToContent flag for the specified crawled property.
proc_MSS_SetManagedPropertyAllOM	Sets all attributes for a managed property in the metadata schema.
proc_MSS_SetManagedPropertyHasMultipleValues	Sets the flag to indicate that the specified managed property can contain multiple values for a single item.
proc_MSS_SetPendingMappings	Copies mappings between the specified managed property and crawled properties from the Pending Mappings Set to the Mappings Set, as specified in Section 3.1.1.1.
proc_MSS_SetSchemaParameter	Changes a Ranking parameter in the Global Ranking Parameter Set, specified in Section 3.1.1.5.
proc_MSS_SetScopeDisplayGroupInfo	Sets the configuration information for a specific search scope display group.
proc_MSS_SetScopeDisplayGroupListItem	Adds a specific search scope to a specific search scope display group with the specified order.
proc_MSS_SetScopeInfo	Sets the details of an existing search scope in the search

Procedure Name	Description
	application.
proc_MSS_SetScopesManagerInfo	Sets the details of the search scopes system in the search application.
proc_MSS_SetScopeRuleInfo	Sets the details of an existing search scope rule in the search application.
proc_MSS_StartScopesCompilation	Sets an indicator to start search scope compilation as soon as possible.
proc_MSS_UpdateBestBet	Changes the information about the specified best bet.
proc_MSS_UpdateBestBetOrder	Updates the order of a best bet among all best bets of the keyword it is associated with. .
proc_MSS_UpdateLocationConfiguration<12>	Changes the attributes of the federated location.
proc_MSS_UpdateProxy<13>	Changes whether or not to use the proxy settings for federated locations.
proc_MSS_UpdateSpecialTerm	Changes the information about the specified keyword.

3.1.4.1 proc_MSS_AddAuthorityPage

The **proc_MSS_AddAuthorityPage** stored procedure is called to add an authority page.

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

```
PROCEDURE proc_MSS_AddAuthorityPage(
    @Url                nvarchar(2048),
    @Hash               int,
    @AuthorityLevel     int
);
```

@Url: The URL of the authority page.

@Hash: The identifier of the URL.

@AuthorityLevel: The authority level of the authority page. This parameter MUST be set to an integer listed in the following table.

Value	Description
0	First-level **authority page**.
33	Second-level **authority page**.

Value	Description
66	Third-level **authority page**.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<14> return any result sets. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.2 proc_MSS_AddBestBet

The **proc_MSS_AddBestBet** stored procedure is called to create a new best bet and associate it with the specified keyword.

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

```
PROCEDURE proc_MSS_AddBestBet(
    @Title              nvarchar(100),
    @Url                nvarchar(2048),
    @Description        nvarchar(500)
    @SpecialTermId      int,
    @ConsumerGpId       nvarchar(50),
    @Order              int,
    @BestBetId          int output
);
```

@Title: The title for the best bet.

@Url: The URL for the best bet.

@Description: The description for the best bet.

@SpecialTermId: The unique identifier of the keyword associated with the newly created best bet.

@ConsumerGpId: The unique identifier of the keyword consumer group for the best bet.

@Order: The order of the best bet among all best bets for a specified keyword.

@BestBetId: Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the newly added best bet.

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

Value	Description
0	Successful execution.
2627	The identifier of the best bet already exists
50000	The best bet with the given URL is already defined for the specified keyword consumer group.

Result Sets: SHOULD NOT<15> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.3 proc_MSS_AddBestBetLink

The **proc_MSS_AddBestBetLink** stored procedure is called to add an association between a best bet and a keyword.

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

```
PROCEDURE proc_MSS_AddBestBetLink(
    @SpecialTermId        int,
    @BestBetId            int,
    @Order                int
);
```

@SpecialTermId: The unique identifier of the keyword associated with the newly created best bet.

@BestBetId: The unique identifier of the best bet.

@Order: The order of the best bet among all best bets for specified keyword.

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

Value	Description
0	Successful execution.
2627	The association between the best bet and the keyword already exists

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

3.1.4.4 proc_MSS_AddChildContentSource

The **proc_MSS_AddChildContentSource** stored procedure is called to add a start address to the Child Farm Start Addresses Set specified in Section 3.1.1.4.

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

```
PROCEDURE proc_MSS_AddChildContentSource (
    @FarmName          nvarchar(64),
    @Name              nvarchar(64),
    @StartAddress      nvarchar(2048)
);
```

@FarmName: The name of the child farm**.**

@Name: A The unique identifier of the start address

@StartAddress: The start address of content from a child farm.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<17> return any result sets. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.5 proc_MSS_AddConsumer

The **proc_MSS_AddConsumer** Stored Procedure is called to add a search scope consumer to the list of search scope consumers in order to view, add, change or delete search scopes and search scope display groups within it.

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

```
PROCEDURE proc_MSS_AddConsumer (
    @ConsumerName       nvarchar(60)
);
```

@ConsumerName: The name that uniquely identifies the search scope consumer.

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

Value	Description
0	Successful execution.
1	The search scope consumer already exists.

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

3.1.4.6 proc_MSS_AddCrawledPropertyCategoryFromOM

The **proc_MSS_AddCrawledPropertyCategoryFromOM** stored procedure is called to add a crawled property category to the Crawled Property Category Set, specified in Section 3.1.1.1. This procedure also changes the CrawledPropertyCategoryTimestamp attribute in the Metadata Timestamp Set with the current date and time in local time of the server.

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

```
PROCEDURE proc_MSS_AddCrawledPropertyCategoryFromOM (
    @CategoryName       nvarchar(64),
    @Propset            uniqueidentifier
);
```

@CategoryName: The name of the crawled property category.

@Propset: The crawled property set identifier associated with the crawled property category.

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.

3.1.4.7 proc_MSS_AddCrawledPropertyForOM

The **proc_MSS_AddCrawledPropertyForOM** stored procedure is called to add a crawled property to the Crawled Property Set, specified in Section 3.1.1.1. This procedure also changes the

CrawledPropertyTimestamp attribute in the Metadata Timestamp Set to the current date and time in local time of the server.

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

```
PROCEDURE proc_MSS_AddCrawledPropertyForOM (
    @Propset                uniqueidentifier,
    @PropertyName           nvarchar(440),
    @PropertyNameIsEnum     bit,
    @VariantType            int,
    @CrawledPropertyId      int OUTPUT
);
```

@Propset: The crawled property set identifier associated with the crawled property category.

@PropertyName: The name of the crawled property.

@PropertyNameIsEnum: This parameter **MUST be set to 1 if the** *@PropertyName* **string value was converted from an integer. Otherwise, it MUST be set to 0**.

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

@CrawledPropertyId: Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the crawled 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.

3.1.4.8 proc_MSS_AddManagedPropertyAlias

The **proc_MSS_AddManagedPropertyAlias** stored procedure is called to add another name which is associated with the managed property to the metadata schema.

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

```
PROCEDURE proc_MSS_AddManagedPropertyAlias (
    @PID                    int,
    @Name                   nvarchar(2048),
    @UpdateManagedProperty  bit
);
```

@PID: The unique identifier of the managed property.

@Name: An alternate string which identifies the managed property**.**

@ UpdateManagedProperty: If this parameter is set to 1, the server MUST change the value of ManagedPropertyAddModifyTimestamp in the Metadata Timestamp Set and the value of LastModifiedTime attribute in the Managed Property Set for the corresponding managed property to the current date and time in local time of the server;. Otherwise, it MUST be set to 0. For the specification of Metadata Timestamp Set and Managed Property Set see Section 3.1.1.1.

Return Code Values: An integer which MUST be 0

Result Sets: SHOULD NOT<21> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.9 proc_MSS_AddManagedPropertyByName

The **proc_MSS_AddManagedPropertyByName** stored procedure is called to add a custom managed property to the Managed Property Set, specified in Section 3.1.1.1. All custom properties are created with the FullTextQueriable and Retrievable flags set to 1 which puts the data for the custom properties in both the full-text index catalog and the metadata index.

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

```
PROCEDURE proc_MSS_AddManagedPropertyByName (
    @Name                   nvarchar(64),
    @ManagedType            int,
    @PID                    int OUTPUT,
    @FullTextQueriable      bit OUTPUT,
    @Retrievable            bit OUTPUT
);
```

@Name: The name of the managed property.

@ManagedType: The type as specified in INTREFERNCE:[Section 2.1.1.17] of the managed property**.**

@PID: Upon return from this stored procedure, this parameter MUST be set to **t**he unique identifier of the managed property.

@FullTextQueriable: Upon return from this stored procedure, this parameter MUST be set to 1.

@Retrievable: Upon return from this stored procedure, this parameter MUST be set to 1.

Return Code Values: An integer which MUST be 0

Result Sets: SHOULD NOT<22> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.10 proc_MSS_AddMappingToPendingTable

The **proc_MSS_AddMappingToPendingTable** stored procedure is called to add a mapping between a crawled property and a managed property into the Pending Mappings Set, specified in Section 3.1.1.1, as the lowest priority mapping.

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

```
PROCEDURE proc_MSS_ AddMappingToPendingTable (
    @PID                    int,
    @CrawledPropset         uniqueidentifier,
    @CrawledPropertyName    nvarchar(440),
    @CrawledVariantType     int
);
```

@PID: The unique identifier of the managed property.

@CrawledPropset: The crawled property set identifier associated with the crawled property.

@CrawledPropertyName: The name of the crawled property.

@CrawledVariantType: The variant type for the crawled property.

Return Code Values: An integer which MUST be 0

Result Sets: SHOULD NOT<23> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.11 proc_MSS_AddNewLocationConfiguration

The **proc_MSS_AddNewLocationConfiguration** stored procedure is called to add a new Federated Location.

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

```
PROCEDURE proc_MSS_AddNewLocationConfiguration(
     @InternalName                 nvarchar(60),
     @DisplayName                  nvarchar(60),
     @AdminDescription             nvarchar(256),
     @LocationType                 tinyint,
     @Author                       nvarchar(60),
     @Version                      nvarchar(50),
     @IsDeletable                  bit,
     @IsPrefixPattern              bit,
     @QueryReformatPattern         ntext,
     @Propertieschema              ntext,
     @QueryRestriction             nvarchar(512),
     @KindsOfResults               ntext,
     @Languages                    ntext,
     @IsRestricted                 bit,
     @AllowedSiteCollectionGuids   ntext,
     @Type                         tinyint,
     @Data                         ntext,
     @ConnectionUrlTemplate        nvarchar(2048),
     @MoreUrlTemplate              nvarchar(2048),
     @CreationDate                 datetime OUTPUT,
     @LastModifiedDate             datetime OUTPUT,
     @LocationDescription          ntext,
     @LocationId                   int OUTPUT
);
```

@InternalName: The unique internal name of the federated location that will be added.

@DisplayName: The display name of the federated location.

@AdminDescription: The description of the federated location for the **site collection administrator**.

@LocationType: An 8 bit integer that specifies the protocol that MUST be used to connect to this federated location. The value MUST be a Location Type data type as specified in Location Type (section 2.2.2.16).

@Author: The author of the federated location.

@Version: An optional version number for this federated location. If a value is specified, it MUST contain at least one period (".").

@IsDeletable: MUST be 0 when the federated location is added prior to steady state. Otherwise it MUST be 1.

@IsPrefixPattern: MUST be 1 when the *@QueryReformatPattern* parameter is used as a prefix match. Otherwise, it MUST be 0, indicating that the *@QueryReformatPattern* parameter is used as a regular expression.

@QueryReformatPattern: The pattern for triggering the federated location during a search.

@Propertieschema: This parameter MUST be ignored by the server.

@QueryRestriction: The supplemental **query text** that MUST be appended to every query issued by the end user to this federated location.

@KindsOfResults: This parameter MUST be ignored by the server.

@Languages: The languages supported by the federated location.

@IsRestricted: MUST be 1 when this federated location is restricted to only the allowed **sites** specified by the *@AllowedSiteCollectionGuids* parameter. Otherwise, it MUST be 0.

@AllowedSiteCollectionGuids: A semi-colon delimited string of sites which are allowed to configure this federated location.

@Type: An 8-bit integer that specifies the type of authentication supported for federation. The value MUST be an Authentication Type data type, as specified in **Authentication Type** (section 2.2.2.1).

@Data: The authentication credential data.

@ConnectionUrlTemplate: The template for passing **search queries** to this federated location.

@MoreUrlTemplate: The template that specifies the URL of the HTML page that displays results for a search query.

@CreationDate: Upon return from this stored procedure, this parameter MUST be set to the UTC date and time when the federated location is created.

@LastModifiedDate: Upon return from this stored procedure, this parameter MUST be set to the UTC date and time when the federated location was last modified.

@LocationDescription: The XML document that describes the federated location.

@LocationId: Upon return from this stored procedure, this parameter MUST be set to the identifier of the newly added federated location.

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

Value	Description
0	Successful execution.
2627	The Federated Location with this @InternalName already exists.

Result Sets: MUST NOT return any result set.

3.1.4.12 proc_MSS_AddScope

The **proc_MSS_AddScope** stored procedure is called to add a new search scope to the search application**.** Upon successful execution the stored procedure increments the LastChangeId attribute in the Scope System Set and copies this value to the LastChangeId attribute in the Scope Set for the specified search scope. For the specification of ScopeSystem Set and Scope Set see Section 3.1.1.3 .

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

```
PROCEDURE proc_MSS_AddScope (
      @Name                         nvarchar(60),
      @Description                  nvarchar(300),
      @ConsumerName                 nvarchar(60),
      @DisplayInAdminUI             bit,
      @AlternateResultsPageUrl      nvarchar(2048),
      @CompilationType              smallint,
      @ModifierName                 nvarchar(60),
      @ScopeID                      int OUTPUT
);
```

@Name: The name of the search scope.

@Description: The description of the search scope.

@ConsumerName: The name of the search scope consumer which owns the search scope.

@DisplayInAdminUI: If this parameter is set to 1, the search scope is displayed in the Administration UI. Otherwise, it MUST be set to 0.

@ AlternateResultsPageUrl: The URL of an alternate web page to display the results of a search performed on this search scope.

@CompilationType: The compilation type of the search scope. The value MUST be a CompilationType data type as specified in Section 2.2.2.5.

@ModifierName: The name of the user who last changed the attributes of the search scope.

@ScopeID: Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the created search scope.

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

Value	Description
0	Successful execution. The search scope was successfully added to the search application.
1	Search scope consumer with the specified name could not be found. The search scope was not added to the search application.
2627	A search scope with the name specified already exists for the specified search scope consumer. The search scope was not added to the search application.

Result Sets: SHOULD NOT<24> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.13 proc_MSS_AddScopeDisplayGroup

The **proc_MSS_AddScopeDisplayGroup** stored procedure is called to add a search scope display group for a search scope consumer.

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

```
PROCEDURE proc_MSS_AddScopeDisplayGroup (
    @Name                   nvarchar(60),
    @Description            nvarchar(300),
    @ConsumerName           nvarchar(60),
    @DisplayInAdminUI       bit,
    @Undeletable            bit,
    @DefaultScopeID         int,
    @ModifierName           nvarchar(60),
    @DisplayGroupID         int OUTPUT
);
```

@Name: The name that uniquely identifies the search scope display group.

@Description: The description of the search scope display group.

@ConsumerName: The name of the search scope consumer who is the owner of the search scope display group.

@DisplayInAdminUI: A bit flag indicating if the search scope display group is displayed in the Administration user interface. The value MUST be a DisplayInAdminUI data type as specified in Section 2.2.2.6.

@Undeletable: A bit flag indicating if the search scope display group can be deleted. The value MUST be an Undeletable data type as specified in Section 2.2.2.22**.**

@DefaultScopeID: The unique identifier of the default search scope associated with the search scope display group.

@ModifierName: The name of the user who last changed attributed of the search scope display group.

@DisplayGroupID: Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the search scope display group.

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

Value	Description
0	Successful execution.
1	The search scope consumer does not exist.
2627	The search scope display group with the specified name already exists for the search scope consumer.

Result Sets: SHOULD NOT<25> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.14 proc_MSS_AddScopeRule

The **proc_MSS_AddScopeRule** stored procedure is called to add a new search scope rule to the specified search scope**.** Upon successful execution the stored procedure increments the LastChangeId attribute in the Scope System Set and copies this value to the LastChangeId attribute in the Scope Set for the specified search scope. For the specification of Scope System Set and Scope Set see Section 3.1.1.3.

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

```
PROCEDURE proc_MSS_AddScopeRule (
    @ScopeID                int,
    @FilterBehavior         smallint,
    @RuleType               smallint,
    @UrlRuleType            smallint,
    @PropertyID             int,
    @UserValueString        nvarchar(2048),
    @ModifierName           nvarchar(60),
    @RuleID                 int OUTPUT
);
```

@ScopeID: The unique identifier of the search scope.

@FilterBehavior: The filter behavior of the search scope rule. The value MUST be a valid ScopeFilterBehavior data type as specified in Section 2.2.2.19.

@RuleType: The type of the search scope rule. The value MUST be a ScopeRuleType data type as specified in Section 2.2.2.16.

@UrlRuleType: The type of the URL for the search scope rule. The value MUST be an UrlRuleType data type as specified in Section 2.2.2.23.

@ PropertyID: The unique identifier of the managed property to use in the search scope rule.

@UserValueString: The search scope rule value to use in the search scope rule.

@ModifierName: The name of the user adding the search scope rule.

@RuleID: Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the search scope rule.

Return Code Values: An integer which MUST be 0.

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

3.1.4.15 proc_MSS_AddSpecialTerm

The **proc_MSS_AddSpecialTerm** stored procedure is called to add a new keyword to the site collection**.**

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

```
PROCEDURE proc_MSS_AddSpecialTerm(
    @Term                   nvarchar(100),
    ConsumerGpId            nvarchar(50),
```

```
    @StartDate                datetime,
    @SpecialTermId            int output
);
```

@Term: The term for the keyword.

@ConsumerGpId: The unique identifier of the keyword consumer group associated with the keyword.

@StartDate: The date and time when the keyword begins to appear in search result.

@SpecialTermId: Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the newly added keyword.

Return Code Values: Must return the @SpecialTermId.

Result Sets: SHOULD NOT<26> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.16 proc_MSS_AddSynonym

The **proc_MSS_AddSynonym** stored procedure is called to associate a synonym with the specified keyword in the site collection.

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

```
PROCEDURE proc_MSS_AddSynonym(
    @SpecialTermId          int,
    @Term                   nvarchar(100)
);
```

@SpecialTermId: The unique identifier of the keyword associated with the keyword synonym.

@Term: The term for the keyword synonym.

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

Value	Description
0	Successful execution.
2627	The synonym with the specified *@Term* is already defined for the specified keyword.

Result Sets: SHOULD NOT<27> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.17 proc_MSS_BeginScopeDisplayGroupList

The **proc_MSS_BeginScopeDisplayGroupList** Stored Procedure is called to remove all search scopes with negative order from the specified search scope display group. The stored procedure MUST be called before the first call to proc_MSS_SetScopeDisplayGroupListItem stored procedure. See section 3.1.4.109 for the specification of proc_MSS_SetScopeDisplayGroupListItem sotred procedure .

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

```
PROCEDURE proc_MSS_BeginScopeDisplayGroupList(
    @DisplayGroupID           int
);
```

@DisplayGroupID: The unique identifier of the search scope display group.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<28> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.18 proc_MSS_Cleanup

The **proc_MSS_Cleanup** stored procedure is called to delete internal data from the search database.

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

```
PROCEDURE proc_MSS_Cleanup();
```

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<29> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.19 proc_MSS_CleanupPropertyTables

The **proc_MSS_CleanupPropertyTables** stored procedure is called to fix the metadata index when it has data that is out of sync and isn't being corrected by crawls.

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

```
PROCEDURE proc_MSS_CleanupPropertyTables();
```

Return Code Values: MUST be one more than the number of out of sync items removed from the metadata index.

Result Sets: SHOULD NOT<30> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.20 proc_MSS_ContainsManagedPropertyAlias

The **proc_MSS_ContainsManagedPropertyAlias** stored procedure is called to determine whether a managed property alias is defined in the metadata schema.

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

```
PROCEDURE proc_MSS_ContainsManagedPropertyAlias (
    @PID              int,
    @alias            nvarchar(2048),
    @found            bit OUTPUT
);
```

@PID: The unique identifier of the managed property.

@alias: An alternate string name which identifies a managed property.

@found: Upon return from this stored procedure, this parameter MUST be set to 1 if the managed property alias is defined in the metadata schema; otherwise the value MUST be set to 0.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<31> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.21 proc_MSS_DeleteAuthorityPage

The **proc_MSS_DeleteAuthorityPage** stored procedure is called to delete the **authority page** with the specified **URL** from the list of **authority pages**.

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

```
PROCEDURE proc_MSS_DeleteAuthorityPage (
    @Url                nvarchar(2048)
);
```

@Url: The URL of the authority page to be deleted.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<32> return any result sets. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.22 proc_MSS_DeleteBestBetLink

The **proc_MSS_DeleteBestBetLink** stored procedure is called to delete an association between the specified keyword and the best bet.

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

```
PROCEDURE proc_MSS_DeleteBestBetLink(
@SpecialTermId          int,
@BestBetId              int
);
```

@SpecialTermId: The unique identifier of the keyword.

@BestBetId: The unique identifier of the best bet.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<33> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.23 proc_MSS_DeleteCrawledCategoryByName

The **proc_MSS_DeleteCrawledCategoryByName** stored procedure is called to delete the specified crawled property from the Crawled Property Category Set. The crawled property MUST

have no crawled properties associated with it in the Crawled Property Set. This procedure also changes the CrawledPropertyCategoryTimestamp attribute in the Metadata Timestamp Set to the current date and time in local time of the server. For the specification of Crawled Property Category Set and Metadata Timestamp Set see Section 3.1.1.1.

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

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

@Name: The name of the crawled property category.

Return Code Values: An integer which MUST be 0

Result Sets: SHOULD NOT<34> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.24 proc_MSS_DeleteCrawledPropertiesUnmappedForCategory

The **proc_MSS_DeleteCrawledPropertiesUnmappedForCategory** stored procedure is called to remove the set of crawled properties from the Crawled Property Set which conform to the following criteria:

- The crawled property belongs to the same crawled property set identifier as the specified crawled property category
- The crawled property 's data is not put in the full-text index catalog.
- The crawled property is not mapped to any managed property.

For the specification of Crawled Property Set see Section 3.1.1.1.

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

```
PROCEDURE proc_MSS_DeleteCrawledPropertiesUnmappedForCategory (
    @CategoryName             nvarchar(64)
);
```

@CategoryName: The name of the crawled property category**.**

Return Code Values: An integer which MUST be 0

Result Sets: SHOULD NOT<35> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.25 proc_MSS_DeleteLocation

The **proc_MSS_DeleteLocation** stored procedure is called to delete a federated location and the visualizations that are associated with it.

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

```
PROCEDURE proc_MSS_DeleteLocation(
    @LocationId        int
```

```
);
```

@LocationId: The unique identifier of the federated location that will be deleted.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.26 proc_MSS_DeleteManagedProperty

The **proc_MSS_DeleteManagedProperty** stored procedure is called to remove a managed property from the Managed Property Set. The managed property MUST have no crawled properties associated with it in the Mappings Set. This procedure also changes the ManagedPropertyDeleteTimestamp attribute in the Metadata Timestamp Set to the current date and time in local time of the server. For the specification of Managed Property Set, Mappings Set, and Metadata Timestamp Set see Section 3.1.1.1.

```
PROCEDURE proc_MSS_DeleteManagedProperty (
    @PID                int
);
```

@PID: The unique identifier of the managed property**.**

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

Value	Description
0	Successful execution.
11	Failed to delete the managed property because there are crawled properties that are still mapped to it.

Result Sets: SHOULD NOT<36> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.27 proc_MSS_DeleteManagedPropertyAlias

The **proc_MSS_DeleteManagedPropertyAlias** stored procedure is called to delete a managed property alias from the metadata schema**.** This procedure also changes the ManagedPropertyAddModifyTimestamp attribute in the Metadata Timestamp Set and the LastModifiedTime attribute in the Managed Property Set for the corresponding managed property to the current date and time. For the specification of Managed Property Set and Metadata Timestamp Set see Section 3.1.1.1.

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

```
PROCEDURE proc_MSS_DeleteManagedPropertyAlias(
    @PID                int,
    @Alias              nvarchar(2048)
);
```

@PID: The unique identifier of the managed property.

@Alias: The managed property alias.

Return Code Values: An integer which MUST be 0

Result Sets: SHOULD NOT<37> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.28 proc_MSS_DeletePropertyMappingsForManagedProperty

The **proc_MSS_DeletePropertyMappingsForManagedProperty** stored procedure is called to delete the crawled property mappings to this managed property from the Mappings Set. This procedure changes the CrawledPropertyTimestamp attribute in the Metadata Timestamp Set to the current date and time in local time of the server**.** For the specification of Mappings Set and Metadata Timestamp Set see Section 3.1.1.1.

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

```
PROCEDURE proc_MSS_DeletePropertyMappingsForManagedProperty (
    @PID                int
);
```

@PID: The unique identifier of the managed property.

Return Code Values: An integer which MUST be 0

Result Sets: SHOULD NOT<38> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.29 proc_MSS_DeletePropertyMappingsPendingForManagedProperty

The **proc_MSS_DeletePropertyMappingsPendingForManagedProperty** stored procedure is called to delete the crawled property mappings to this managed property from the Pending Mappings Set **specified in** Section 3.1.1.1.

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

```
PROCEDURE proc_MSS_DeletePropertyMappingsPendingForManagedProperty (
    @PID                int
);
```

@PID: The unique identifier of the managed property.

Return Code Values: An integer which MUST be 0

Result Sets: SHOULD NOT<39> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.30 proc_MSS_DeleteSpecialTerm

The **proc_MSS_DeleteSpecialTerm** stored procedure is called to delete a keyword from the site collection.

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

```
PROCEDURE proc_MSS_DeleteSpecialTerm(
    @SpecialTermId      int
);
```

@SpecialTermId: The unique identifier of the keyword.

Return Code Values: An integer which MUST be 0

Result Sets: SHOULD NOT<40> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.31 proc_MSS_DeleteSynonym

The **proc_MSS_DeleteSynonym** stored procedure is called to delete a synonym of the specified keyword from the site collection.

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

```
PROCEDURE proc_MSS_DeleteSynonym(
    @SpecialTermId      int,
    @Term               nvarchar(100)
);
```

@SpecialTermId: The unique identifier of the keyword**.**

@Term: The keyword synonym to be deleted.

Return Code Values: An integer which MUST be 0

Result Sets: SHOULD NOT<41> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.32 proc_MSS_DropChildContentSource

The **proc_MSS_DropChildContentSource** stored procedure is called to delete a start address in the Child farm Start Addresses Set, specified in Section 3.1.1.4.

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

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

@Name: A The unique identifier of the start address.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<42> return any result sets. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.33 proc_MSS_DropScope

The **proc_MSS_DropScope** stored procedure is called to delete the specified search scope from the search application. Upon successful execution the stored procedure increments the LastChangeId

attribute in the Scope System Set and copies this value to the LastChangeId attribute in the Scope Set for the specified search scope. The stored procedure also sets the DefaultScopeID attribute in the Scope Display Group Set to -1 for the search scope display groups that have the specified search scope set as their the default search scope.

For the specification of Scope System Set, Scope Set, and Scope Display Group Set see Section 3.1.1.3.

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

```
PROCEDURE proc_MSS_DropScope(
    @ScopeID            int,
    @ModifierName       nvarchar(300)
);
```

@ScopeID: The unique identifier of the search scope to delete.

@ModifierName: The name of the user deleting the search scope

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<43> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.34 proc_MSS_DropScopeDisplayGroup

The **proc_MSS_DropScopeDisplayGroup** stored procedure is called to delete a search scope display group.

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

```
PROCEDURE proc_MSS_DropScopeDisplayGroup(
    @DisplayGroupID           int
);
```

@DisplayGroupID: The unique identifier of the search scope display group.

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

Value	Description
0	Successful execution.
1	The search scope display group does not exist.

Result Sets: SHOULD NOT<44> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.35 proc_MSS_DropScopeRule

The **proc_MSS_DropScopeRule** stored procedure is called to delete an existing search scope rule from the search application. Upon successful execution the stored procedure increments the LastChangeId attribute in the Scope System Set and copies this value to the LastChangeId attribute in the Scope Set for the search scope associated with the specified search scope rule. For the specification of Scope System Set and Scope Set see Section 3.1.1.3.

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

```
PROCEDURE proc_MSS_DropScopeRule(
      @RuleID          int,
      @ModifierName    nvarchar(300)
);
```

@RuleID: The unique identifier of the search scope rule to delete.

@ModifierName: The name of the user deleting the search scope.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<45> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.36 proc_MSS_EndScopeDisplayGroupList

The **proc_MSS_EndScopeDisplayGroupList** stored procedure is called to delete all the search scopes from the specified search scope display group that have positive order and change the negative order of the search scopes contained in the specified search scope display group to its positive value - 1. Upon successful execution the stored procedure changes LastModifiedTime attribute in the Scope Display Group Set to the current date and time in UTC time and increments the LastConsumerChangeID attribute in the Scope System Set. For the specification of Scope System Set and Scope Display Group Set see Section 3.1.1.3. The stored procedure MUST be called after the last call to **proc_MSS_SetScopeDisplayGroupListItem** stored procedure. See section 3.1.4.109 for the specification of **proc_MSS_SetScopeDisplayGroupListItem** stored procedure.

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

```
PROCEDURE proc_MSS_EndScopeDisplayGroupList(
    @DisplayGroupID           int
);
```

@DisplayGroupID: The unique identifier of the search scope display group.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<46> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.37 proc_MSS_GetAllBestBets

The **proc_MSS_GetAllBestBets** stored procedure is called to retrieve all best bets that belong to the specified keyword consumer group**.** Optionally, the list of best bets can be filtered based on *@Filter* value.

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

```
PROCEDURE proc_MSS_GetAllBestBets(
    @ConsumerGpId       nvarchar(50),
    @Filter             int,
    @Value              nvarchar(2048) = null
);
```

@ConsumerGpId: The unique identifier of the keyword consumer group to which the best bets belong.

@Filter: Specifies which attributes of the best bets are used to apply the *@Value filter*. The value MUST be of type Best Bets Filter Type as specified in Section 2.2.2.2.

@Value: A string that is compared against when performing the filtering. The value MUST be set to NULL to list all best bets for the specified keyword consumer group.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.37.1 Best Bets Result Set

The Best Bets Result Set contains information about all best bets for the specified keyword consumer group sorted by *Title* in ascending order. The result set MUST contain zero or more rows, each corresponding to a best bet. See Section 2.2.5.1 for the specification of the result set.

3.1.4.38 proc_MSS_GetAllBestBetsCount

The **proc_MSS_GetAllBestBetsCount** stored procedure is called to calculate the number of best bets for the specified keyword consumer group that match the specified filtering criteria.

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

```
PROCEDURE proc_MSS_GetAllBestBetsCount (
    @ConsumerGpId       NVARCHAR(50),
    @Filter             int,
    @Value              NVARCHAR(2048) = null,
    @Count              int output
);
```

@ConsumerGpId: The unique identifier of the keyword consumer group to which the best bets belong.

@Filter: Specifies which attributes of the best bets are used to apply the *@Value* filter. The value MUST be of type Best Bets Filter Type as specified in Section 2.2.2.2.

@Value: A string that is compared against when performing the filtering. The value MUST be set to NULL to calculate the number of all best bets for the specified keyword consumer group.

@Count: Upon return from this stored procedure, this parameter MUST be set to the number of best bets.

Return Code Values: An integer which MUST be 0

Result Sets: MUST NOT return any result set.

3.1.4.39 proc_MSS_GetAuthorityPages

The **proc_MSS_GetAuthorityPages** stored procedure is called to retrieve a list of authority pages.

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

```
PROCEDURE proc_MSS_GetAuthorityPages();
```

Return Code Values: An integer which MUST be 0

Result Sets: MUST return the following result set:

3.1.4.39.1 Authority Pages Result Set

The Authority Pages result set contains information about authority pages. The result set MUSTcontain zero or more rows, each corresponding to a single authority page.

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

```
DisplayUrl              nvarchar(2048),
DisplayHash             int,
AuthorityLevel          int;
```

DisplayUrl: URL of the authority page.

DisplayHash: The identifier of the authority page URL.

AuthorityLevel: The authority level of the new authority page.

3.1.4.40 proc_MSS_GetBestBet

The **proc_MSS_GetBestBet** stored procedure is called to retrieve a best bet for the specified URL and keyword consumer group.

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

```
PROCEDURE proc_MSS_GetBestBet(
    @ConsumerGpId         nvarchar(50),
    @Url                  nvarchar(2048)
);
```

@ ConsumerGpId: The unique identifier of the keyword consumer group.

@ Url: The URL of the best bet.

Return Code Values: An integer which MUST be 0

Result Sets: MUST return the result set as specified in Section 2.2.5.1.

3.1.4.41 proc_MSS_GetBestBetForSpecialTerm

The **proc_MSS_GetBestBetForSpecialTerm** stored procedure is called to retrieve the information about a best bet for the specified keyword consumer group**,** URL**,** and keyword

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

```
PROCEDURE proc_MSS_GetBestBetForSpecialTerm(
    @ConsumerGpId       nvarchar(50),
    @Url                nvarchar(2048),
    @SpecialTermId      int
```

```
);
```

@ ConsumerGpId: The unique identifier of the keyword consumer group.

@ Url: The URL of the best bet.

@SpecialTermId: The unique identifier of the keyword**.**

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the result set as specified in Section 2.2.5.1.

3.1.4.42 proc_MSS_GetBestBets

The **proc_MSS_GetBestBetForSpecialTerm** stored procedure is called to retrieve the list of best bets for the specified keyword.

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

```
PROCEDURE proc_MSS_GetBestBets(
    @SpecialTermId      int
);
```

@SpecialTermId: The unique identifier of the keyword**.**

Return Code Values: An integer which MUST be 0

Result Sets: MUST return the following result set:

3.1.4.42.1 Best Bets By Order Result Set

Best Bets contains information about the best bets for the specified keyword sorted by *Order* in ascending order. The result set MUST contain zero or more rows, each corresponding to a single best bet.

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

```
BestBetID           int,
Title               nvarchar(100),
Url                 nvarchar(2048),
Description         nvarchar(500),
Order               int;
```

BestBetID: The unique identifier of the best bet.

Title: The title for the best bet.

Url: The URL of the best bet. The value MUST NOT be NULL.

Description: The description of the best bet.

Order: This value MUST be ignored by the protocol client.

3.1.4.43 proc_MSS_GetBestBetsCount

The **proc_MSS_GetBestBetsCount** stored procedure is called to calculate the number of best bets for the specified keyword.

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

```
PROCEDURE proc_MSS_GetBestBetsCount(
    @SpecialTermId     int,
    @Count             int OUTPUT
);
```

@SpecialTermId: The unique identifier of the keyword for which the best bets count is retrieved.

@ Count: Upon return from this stored procedure, this parameter MUST be set to the number of best bets for the specified keyword.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.44 proc_MSS_GetBestBetsOrder

The **proc_MSS_GetBestBetsOrder** stored procedure is called to retrieve a list of best bets associated with the specified keyword and the information about their priorities within this list. The priority of the best bet is determined by the order in which the best bets appears in the search result**.**

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

```
PROCEDURE proc_MSS_GetBestBetsOrder(
    @SpecialTermId            int
);
```

@SpecialTermId: The unique identifier of the keyword**.**

Return Code Values: An integer which MUST be 0

Result Sets: MUST return the following result set:

3.1.4.44.1 Best Bets Priorities Result Set

The Best Bets Priorities result set contains information about the best bets associated with the specified keyword and the information about their priorities within this list. The Best Bets Priorities result set MUST contain zero or more rows. The rows in the result set MUST be sorted by *Order* in ascending order.

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

```
SpecialTermId            nvarchar(100),
BestBetId                int,
Order                    int;
```

@SpecialTermId: The unique identifier of the keyword The value MUST NOT be NULL.

@BestBetId: The unique identifier of the best bet

@Order: Specifies the priority of the best bet within the list of best bets for the specified keyword The value MUST NOT be NULL.

3.1.4.45 proc_MSS_GetChildContentSources

The **proc_MSS_GetChildContentSources** stored procedure is called to retrieve all entries from the Child Farm Start Addresses Set specified in Section 3.1.1.4.

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

```
PROCEDURE proc_MSS_GetChildContentSources();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set.

3.1.4.45.1 Child Content Sources Result Set

The Child Content Source result set contains information about all start addresses created by all child farms. The result set MUST contain zero or more rows, each corresponding to a single start address

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

```
@Name                   nvarchar(64),
@StartAddress           nvarchar(2048)
```

@Name: The unique identifier of the start address,

@StartAddress: start address of content from a child farm.

3.1.4.46 proc_MSS_GetChildContentSourcesForFarm

The **proc_MSS_GetChildContentSourcesForFarm** stored procedure is called to list start addresses for the specified child farm.

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

```
PROCEDURE proc_MSS_GetChildContentSourcesForFarm (
    @FarmName           nvarchar(64)
);
```

@FarmName: The name of the child farm.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.46.1 Child Content Sources Result Set

The Child Content Source result set contains information about all start addresses created by the specified child farm The result set MUST contain zero or more rows, each corresponding to a single start address

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

```
@Name                   nvarchar(64),
@StartAddress           nvarchar(2048)
```

@Name: The unique identifier of the start address,

@StartAddress: The start address of content from a child farm.

3.1.4.47 proc_MSS_GetConsumers

The **proc_MSS_GetConsumers** stored procedure is called to retrieve a list of names of all search scope consumers.

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

```
PROCEDURE proc_MSS_GetConsumers();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.47.1 Consumers Result Set

The Consumers result set contains information about search scope consumers. The result set MUST contain zero or more rows, each corresponding to a single search scope consumer.

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

```
ConsumerName            nvarchar(60);
```

ConsumerName: The name of the search scope consumer.

3.1.4.48 proc_MSS_GetContainingScopeDisplayGroups

The proc_MSS_GetContainingScopeDisplayGroups stored procedure is called to retrieve a list of all search scope display groups that contain the specified search scope.

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

```
PROCEDURE proc_MSS_GetContainingScopeDisplayGroups(
    @ScopeID                int
);
```

@ScopeID: The unique identifier of the search scope.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.48.1 Scope Display Groups Result Set

The Scope Display Groups result set contains information about the search scope display groups that contain the specified search scope. The result set MUST contain zero or more rows, each corresponding to a single search scope display group that contains the specified search scope.

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

```
DisplayGroupID          int;
```

DisplayGroupID: The unique identifier of the search scope display group.

3.1.4.49 proc_MSS_GetCrawledProperty

The **proc_MSS_GetCrawledProperty** stored procedure is called to retrieve a crawled property from the metadata schema.

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

```
PROCEDURE proc_MSS_GetCrawledProperty(
    @Propset            uniqueIdentifier,
    @PropertyName       nvarchar(440),
    @VariantType        int
);
```

@Propset: The crawled property set identifier associated with the crawled property category.

@PropertyName: The name of the crawled property.

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a single result set as specified in Section 2.2.5.2.

3.1.4.50 proc_MSS_GetCrawledPropertyID

The **proc_MSS_GetCrawledPropertyID** stored procedure is called to retrieve the identifier of the specified crawled property from the metadata schema.

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

```
PROCEDURE proc_MSS_GetCrawledPropertyID(
    @Propset            uniqueIdentifier,
    @PropertyName       nvarchar(440),
    @VariantType        int
);
```

@Propset: The **crawled property set identifier associated with the crawled property category.**

@PropertyName: The name of the crawled property.

@VariantType: The variant type for the crawled property.

Return Code Values: An integer which MUST be 0

Result Sets: MUST return the following result set:

3.1.4.50.1 Crawled Property ID Result Set

Crawled Property ID result set contains information about the specified crawled property. The result set MUST contain zero or one row.

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

```
CrawledPropertyId      int;
```

CrawledPropertyId: The unique identifier of the crawled property.The protocol client MUST ignore it.

3.1.4.51 proc_MSS_GetCrawledPropertiesAllForCategory

The **proc_MSS_GetCrawledPropertiesAllForCategory** stored procedure is called to retrieve a list of crawled properties associated with the specified crawled property category from the metadata schema.

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

```
PROCEDURE proc_MSS_GetCrawledPropertiesAllForCategory (
    @CategoryName      nvarchar(64)
);
```

@CategoryName: The name for the crawled property category.

Return Code Values: An integer which MUST be 0

Result Sets: MUST return a single result set as specified in Section 2.2.5.2.

3.1.4.52 proc_MSS_GetCrawledPropertiesForOM

The **proc_MSS_GetCrawledPropertiesForOM** stored procedure is called to retrieve a list of crawled properties associated with the specified crawled property category from the metadata schema.

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

```
PROCEDURE proc_MSS_GetCrawledPropertiesForOM(
    @CategoryName           nvarchar(64),
    @Filter                 nvarchar(440),
    @MaxProps               int,
    @LastPropset            uniqueIdentifier,
    @LastPropertyName       nvarchar(440),
    @ForwardDirection       bit
);
```

@CategoryName: If the value is set to "%%", the result set MUST include crawled properties from any crawled property category**;** otherwise the result set MUST include crawled properties from the crawled property category matching this parameter**.**

@Filter: The Filter crawled properties to which property names are matched, according to the Filter Wildcard Rules as defined in Section 2.2.2.8**.**

@Maxprops: The maximum number of rows to be included in the result set. If zero, MUST return all rows.

@LastPropset: The crawled property set identifier used to disambiguate when multiple *@LastPropertyName* values exist in the Crawled Properties Set, or null for the first call. See Section 3.1.1.1 for the specification of Crawled Properties Set.

@LastPropertyName: The property name used as the offset to determine the next property that will be returned in the result set, or NULL for the first call.

@ForwardDirection: If 1, result set MUST be ordered by ascending *PropertyName* and *Propertieset*; otherwise, if 0, MUST order the result set by descending *PropertyName* and *Propertieset*.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a single result set as specified in Section 2.2.5.2, subject to the conditions imposed by this stored procedures parameters.

3.1.4.53 proc_MSS_GetCrawledPropertiesBasic

The **proc_MSS_GetCrawledPropertiesBasic** stored procedure is called to retrieve a list of crawled properties from the metadata schema **which is ordered by** *@Propset* **and then by** *@PropertyName*.

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

```
PROCEDURE proc_MSS_GetCrawledPropertiesBasic (
    @Propset           uniqueidentifier,
    @PropertyName      nvarchar(440)
);
```

@Propset: The crawled property set identifier used as the offset to determine the next property that will be returned in the result set, or NULL for the first call.

@PropertyName: The property name used to disambiguate when multiple properties exist with the same crawled property set identifier in the Crawled Properties Set, or NULL for the first call.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.53.1 Crawled Properties Basic Result Set

The Crawled Properties Basic result set contains information about the crawled properties starting with the specified crawled property set identifier and property name if any. The result set MUST contain zero or more rows, each corresponding to a single crawled property.

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);
```

Propset: The crawled property set identifier associated with the crawled property category.

PropertyName: The name of the crawled property.

PropertyNameIsEnum: The value MUST be set to 1 if the *PropertyName* string value was converted from an integer. Otherwise, it MUST be set to 0**.**

IsMappedToContent: The value **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.

IsSampleCacheFull: The value MUST be set to 1 if the Sample Crawled Properties Set is complete. Otherwise, it MUST be set to 0. For the specification of Sample Crawled Properties Set see Section 3.1.1.1**.**

VariantType: The variant type for the crawled property.

CrawledPropertyId: The unique identifier of the **crawled property.**

URI: The URI associated with the crawled property**.**

3.1.4.54 proc_MSS_GetCrawledPropertyCategoriesBasic

The **proc_MSS_GetCrawledPropertyCategoriesBasic** stored procedure is called to list all **crawled property categories** from the **metadata schema.**

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

```
PROCEDURE proc_MSS_GetCrawledPropertyCategoriesBasic ();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.54.1 Crawled Property Categories Result Set

The Crawled Property Categories result set contains information about all **crawled property categories**. The result set MUST contain zero or more rows, each corresponding to a **crawled property category**. The result set MUST be sorted by *CategoryName* in ascending order.

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

```
Propset                    uniqueidentifier,
CategoryName               nvarchar(64),
CrawledPropertyCount       int,
DiscoverNewProperties      bit,
MapToContents              bit,
```

```
FullTextQueriable           bit,
Retrievable                 bit,
URINamespace                nvarchar(440);
```

Propset: The **crawled property set identifier associated with the crawled property category.**

CategoryName: The name of the **crawled property category.**

CrawledPropertyCount: The number of **crawled properties** with this *Propset* in the **metadata schema**.

DiscoverNewProperties: MUST be set to 1 if the **crawled properties** within this **crawled property** are added to the Crawled Properties Set automatically. Otherwise, it MUST be set to 0.

MapToContents: MUST be set to 1 if string data from newly discovered **crawled properties** within this **crawled property** is put in the **full-text index catalog**. Otherwise, it MUST be set to 0.

FullTextQueriable: MUST be set to 1 if string data from newly discovered **crawled properties** within this **crawled property** 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: MUST be set to 1 if string data from newly discovered **crawled properties** within this **crawled property** is mapped to a new **managed property** which will be put in the **metadata index**. Otherwise, it MUST be set to 0.

URINamespace: This parameter MUST be ignored by the protocol server.

3.1.4.55 proc_MSS_GetCrawledPropertiesamplesByPropertyID

The **proc_MSS_GetCrawledPropertiesamplesByPropertyID** stored procedure is called to retrieve the list of items associated with the specified crawled property from the Sample Crawled Property Set as specified in Section 3.1.1.1.

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

```
PROCEDURE proc_MSS_GetCrawledPropertiesamplesByPropertyID (
    @CrawledPropertyId        int,
    @SampleCount              int
);
```

@CrawledPropertyId: The unique identifier of the crawled property**.**

@SampleCount: The maximum number of rows returned in Crawled Property Samples result set

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set::

3.1.4.55.1 Crawled Property Samples Result Set

The Crawled Property Samples Result set contains information about the items associated with the specified of crawled property from the Sample Crawled Property Set. The result set MUST contain zero or up to *@SampleCount* rows. The result set MUST be sorted by *strVal* field in ascending order.

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

```
strVal                  nvarchar(2064);
```

strVal**: The** URL **of the item which contains the** crawled property.

3.1.4.56 proc_MSS_GetCrawledPropertiesUnmappedForCategory

The **proc_MSS_GetCrawledPropertiesUnmappedForCategory** stored procedure is called to retrieve crawled properties which have no mappings to the full-text index catalog or the metadata index in the metadata schema.

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

```
PROCEDURE proc_MSS_GetCrawledPropertiesUnmappedForCategory (
    @CategoryName             nvarchar(64)
);
```

@CategoryName: The name of the crawled property associated with the crawled property**.**

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.56.1 Unmapped Crawled Properties Result Set

The Unmapped Crawled Properties result set contains information about the crawled properties which have no mappings to the full-text index catalog or the metadata index in the metadata schema. The result set MUST contain zero or more rows each corresponding to single crawled property. The rows in the result set MUST be sorted in ascending order by *PropertyName* and *Propset*. For the specification of the result set see Section 2.2.5.2.

3.1.4.57 proc_MSS_GetCrawlHistory

The **proc_MSS_GetCrawlHistory** stored procedure is called to retrieve information about crawls.

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

```
PROCEDURE proc_MSS_GetCrawlHistory (
    @MaxRecords               int,
    @BeginTime                datetime,
    @EndTime                  datetime,
    @CrawlStatus              int,
    @ContentSourceID          int
);
```

@MaxRecords: The maximum number of rows in the returned result set, or NULL to indicate no limit on the number of rows in the returned result set.

@BeginTime: A **DateTime** in local time that indicates the start date of the time span for which the crawl information MUST be returned in the result set. It MUST be set to NULL to get crawl information for the last 7 days from the current local time on the

@EndTime: A datetime in local time that indicates the end date of the time span for which the crawl information MUST be returned in the result set. It MUST be NULL if @BeginTime is set to NULL. MUST NOT be set to NULL if @BeginTime is set to a value other than NULL.

@CrawlStatus: The **crawl status** to which the returned result set of crawl information is limited, or NULL to indicate that the returned result set is not limited by any crawl status. The specified value MUST be either NULL or one of the valid values specified for crawl status.

@ContentSourceID: The content source for which to return crawl information, or NULL to not be limited by content source.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.57.1 CrawlHistory Result Set

The CrawlHistory result set contains information about crawls. The result set MUST contain zero or more rows each corresponding to a crawl that matches the specified constraints. The result set MUST be ordered in descending order by *EndTime*.

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

```
CrawlID              int,
CrawlType            int,
ContentSourceID      int,
Status               int,
StartTime            datetime,
EndTime              datetime,
SuccessCount         int,
ErrorCount           int,
WarningCount         int;
```

CrawlID: The unique identifier for a crawl.

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

Value	Description
1	Full crawl
2	Incremental crawl
6	Delete crawl

ContentSourceID: The unique identifier of the content source being crawled.

Status: MUST be a crawl status data type as specified in [MS-SQLPGAT], Section 2.2.1.3.

StartTime: A datetime in local time of the server indicating when the crawl was started.

EndTime: A datetime in local time of the server indicating indicating when the crawl finished.

SuccessCount: The number of items that were successfully crawled.

ErrorCount: The number of errors that were encountered during the crawl.

WarningCount: The number of warnings that were encountered during the crawl.

3.1.4.58 proc_MSS_GetCurrentLogData

The **proc_MSS_GetCurrentLogData** stored procedure is called to retrieve the count and the list of successes, warnings and errors logged for display URLs when they were processed by the crawler The counts and the list are retrieved based on the values of the input parameters of the stored procedure. The counts and the list can be retrieved for all or specific display URL, content source, host name, error level or date range.

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

```
PROCEDURE proc_MSS_GetCurrentLogData (
    @MinDate                datetime = null,
    @MaxDate                datetime = null,
    @MessageType            int = null,
    @ContentSourceId        int = null,
    @MessageId              int = null,
    @Count                  int = null,
    @Url                    nvarchar(2048) = null,
    @IsLike                 bit = null,
    @HostName               nvarchar(300) = null,
    @MustUseHostName        bit = null,
    @CatalogID              int = null
);
```

@MinDate: The minimum date and time after which the successes, warnings and errors logged are to be retrieved.

@MaxDate: The maximum date and time before which the successes, warnings and errors logged are to be retrieved.

@MessageType: The error level, which if not NULL, MUST be an ErrorLevel data type as specified in Section 2.2.2.7.

@ContentSourceId: The unique identifier of the content source whose success, warning and error counts are to be retrieved.

@MessageId: The unique identifier for the type of error that was encountered when an item was processed by the crawler.

@Count: The integer used to restrict the number of rows of display URLs used in the query.

@Url: The display URL whose success, warning and error counts are to be retrieved.

@IsLike: A bit flag specifying how *@Url* value is matched against display URLs. Its value MUST be in the following table.

Value	Description
0	If @Url is not NULL, errors are retrieved for display URL which is an exact match to the @Url parameter value.
1	If @Url is not NULL, errors are retrieved for display URLs which match the pattern in the @Url parameter value.

@HostName: The host name whose success, warning and error counts are to be retrieved.

@MustUseHostName: A bit flag specifying whether the counts are to be retrieved for the specified host name. Its value MUST be in the following table.

Value	Description
0	Counts are retrieved for all the host names. Any host name specified as a parameter MUST be ignored.
1	Counts retrieved for the specified host name.

@CatalogID: The unique identifier for the catalog, which is a project identifier data type as specified in [MS-SQLPGAT].

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

Value	Description
0	Successful execution.
50000	Counts are to be retrieved for a specific host name but the host name is not specified or the specified host name does not exist.

Result Sets: MUST return two result sets in the following order:

3.1.4.58.1 Count Result Set

The Count Data result set contains information about the count of successes, errors and warnings logged for display URLs when they were processed by the crawler. The result set MUSTcontain 0 to 3 rows, each row corresponding to a single error level if there is a count greater than 0 for it. The result set MUST NOT be returned when:

- No input parameters are specified.
- Only the *@Count* input parameter is specified.
- Only the *@IsLike* input parameter is specified.
- Only the *@MustUseHostName* input parameter is specified.
- *@HostName* input parameter is specified but does not exist.

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

```
ErrorLevel              int,
Cnt                     int;
```

ErrorLevel: The error level which MUST be an ErrorLevel data type as specified in Section 2.2.2.7.

Cnt: The count of successes, warnings and errors when the *ErrorLevel* is 0, 1 and 2 respectively.

3.1.4.58.2 Log Data Result Set

The Log Data Result set contains information about the successes, warnings and errors logged for the specified display URL when the display URL was processed by the crawler. The result set MUST

contain zero or more rows, each row corresponding to a single success, warning or error logged for a display URL sorted by *LastTouchStart* in descending order. The result set MUST be returned only if the *@Count* input parameter is specified.

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

```
DisplayUrl              nvarchar(1500),
ErrorLevel              int,
ErrorMsg                nvarchar(2000),
HResult                 int,
ErrorDesc               nvarchar(512),
ContentSourceID         int,
LastTouchStart          datetime;
```

DisplayUrl: The specified display URL.

ErrorLevel: The error level which MUST be an ErrorLevel data type as specified in Section 2.2.2.7

ErrorMsg: The descriptive message for the success, warning or error.

HResult: The HRESULT value of the success, warning or error.

ErrorDesc: Additional description for the success, warning or error.

ContentSourceID: The unique identifier of the content source to which the display URL belongs.

LastTouchStart: The date and time the success, error or warning was logged.

3.1.4.59 proc_MSS_GetLastLocationConfigUpdate

The proc_MSS_GetLastLocationConfigUpdate stored procedure is called to retrieve the LastConfigurationChange of the Federation Set as specified in Section 3.1.1.6.

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

```
PROCEDURE proc_MSS_GetLastLocationConfigUpdate();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.59.1 LastLocationConfigUpdate Result Set

The LastLocationConfigUpdate result set contains the version indicating the most recent federated location or visualization configuration change. The result set MUST contain one row.

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

```
LastUpdate              bigint;
```

LastUpdate: The LastConfigurationChange of the Federation Set as specified in Section 3.1.1.6.

3.1.4.60 proc_MSS_GetLocationConfigurations

The **proc_MSS_GetLocationConfigurations** stored procedure is called to retrieve configuration information for all the federated locations.

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

PROCEDURE proc_MSS_GetLocationConfigurations();

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return three result sets in the following order:

3.1.4.60.1 Locations Result Set

The Locations result set contains information about all the federated locations. Each row in the result set contains information about a federated location and its associated Authentication Type data type, as specified in Section 2.2.2.1. The result set MUST contain zero or more rows.

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

```
Id                          int,
InternalName                nvarchar(60),
DisplayName                 nvarchar(60),
AdminDescription            nvarchar(256),
LocationType                tinyint,
Author                      nvarchar(60),
Version                     nvarchar(50),
IsDeletable                 bit,
IsPrefixPattern             bit,
QueryReformatPattern        nvarchar(512),
Propertieschema             ntext,
QueryRestriction            nvarchar(512),
KindsOfResults              ntext,
Languages                   ntext,
IsRestricted                bit,
AllowedSiteCollectionGuids  ntext,
CreationDate                datetime,
LastmodifiedDate            datetime,
Type                        tinyint,
Data                        ntext;
```

Id: The unique identifier of a federated location that will be retrieved.

InternalName: The unique internal name of the federated location.

DisplayName: The display name of the federated location.

AdminDescription: The site collection administrator's description of the federated location.

LocationType: Specifies the protocol that MUST be used to connect to the federated location. The value MUST be a Location Type Data Type as specified in section 2.2.2.16.

Author: The author of the federated location.

Version: The version number for the federated location. The value MUST contain at least one period (".").

IsDeletable: MUST be set to 0 when the federated location cannot be deleted. Otherwise, it MUST be set to 1.

IsPrefixPattern: A 1-bit number that indicates if the query format pattern for the federated location is a prefix pattern or a regular expression. MUST be 1 when the *@QueryReformatPattern* parameter is used as a prefix match. Otherwise, it MUST be 0, indicating that the *@QueryReformatPattern* parameter is used as a regular expression.

QueryReformatPattern: The pattern for triggering the federated location during a search.

Propertieschema: A binary large object (BLOB) that contains the property schema of the federated location.

QueryRestriction: The supplemental query restrictions to be appended to every search query.

KindsOfResults: A binary large object (BLOB) that contains the kinds of results that can be retrieved from the federated location.

Languages: The languages supported by the location. It is specified as **language code identifier (LCID)** for the languages, delimited by "#;#".

IsRestricted: A 1-bit number that indicates whether the permissions to modify the federated location are restricted or not. MUST be 1 when this federated location is restricted to only the allowed sites specified by the *AllowedSiteCollectionGuids* parameter. Otherwise, it MUST be 0.

AllowedSiteCollectionGuids: A semi-colon delimited string of site identifieres which are allowed to configure this federated location.

CreationDate: The UTC date and time when the federated location was created.

LastModifiedDate: The UTC date and time when the federated location was last modified.

Type: An 8 bit integer that specifies the type of authentication supported for federation. The value MUST be an Authentication Type data type, as specified in section 2.2.2.1.

Data: The credentials used for authentication to the federated location.

3.1.4.60.2 LocationTemplates Result Set

The LocationTemplates result set contains federated location query templates used by federated locations. Each row in the result set contains a federated location query template and the corresponding location ID of the federated location that uses the federated location query template. There is a row for each federated location query template. The result set MUST contain zero or more rows.

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

```
InternalName            nvarchar(60),
LocationId              int,
UrlTemplate             nvarchar(2048),
TemplateType            tinyint;
```

InternalName: The Internal Name of the federated location that uses the federated location query template.

LocationId: The unique identifier of the federated location.

UrlTemplate: See the UrlTemplate attribute of the federated location query template.

TemplateType: MUST be 0 when the UrlTemplate is for the Connection URL. Otherwise MUST be 1, indicating the UrlTemplate is for the More Results Link.

3.1.4.60.3 LastLocationConfigUpdate2 Result Set

The LastLocationConfigUpdate2 result set contains the version indicating the most recent federated location configuration change. Each more recent federation configuration change MUST result in a different, larger version. The result set MUST contain one row.

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

```
LastUpdate              bigint,
UseCrawlProxy           bit;
```

LastUpdate: The last version.

UseCrawlProxy: a 1-bit number that indicates whether federated locations are configured to use a proxy when retrieving search results. This flag applies to all federated locations.

3.1.4.61 proc_MSS_GetLocationDescription

The **proc_MSS_GetLocationDescription** stored procedure is called to retrieve the federated location definition**.**

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

```
PROCEDURE proc_MSS_GetLocationDescription (
    @LocationId         int
);
```

@LocationId: The unique identifier of the federated location whose federated location definition will be retrieved.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.61.1 Location Description Result Set

Location Description Result Set contains the XML that describes the specified federated location. The result set MUST contain zero or one row, with the row corresponding to the specified federated location.

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

```
LocationId              int,
Xml                     ntext;
```

LocationId: The unique identifier of the specified federated location.

Xml: The federated location definition of the specified federated location.

3.1.4.62 proc_MSS_GetLocationVisualisations

The **proc_MSS_GetLocationVisualisations** stored procedure is called to retrieve the specified visualization or all visualizations of the specified federated location.

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

```
PROCEDURE proc_MSS_GetLocationVisualisations (
    @LocationId               int,
    @VisualisationName        nvarchar(60) = NULL
);
```

@LocationId: The identifier of the specified federated location whose visualizations will be retrieved.

@VisualisationName: The name of the specific visualization that needs to be retrieved. If this parameter is NULL or empty string, all visualizations of the federated location will be retrieved.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.62.1 LocationVisualisation Result Set

The Location Visualisation Result Set contains information about visualizations of a federated location. Each row in the result set contains information about a visualization for the specified federated location. The result set MUST contain zero or more rows.

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

```
LocationId              int,
VisualisationName       nvarchar(60),
Xsl                     ntext,
Properties              ntext,
SampleData              ntext;
```

LocationId: The identifier of the specified federated location for which the visualization will be retrieved.

VisualisationName: The name of a visualization for the federated location that will be retrieved.

Xsl: The Xsl for this visualization. This MUST be an **Xsl** Data Type as specified in Section 2.2.2.24.

Properties: The properties for this visualization. This MUST be a **Properties** Data Type as specified in Section 2.2.2.18.

SampleData: The sample data for this visualization. This MUST be a **SampleData** Data Type as specified in Section 2.2.2.19.

3.1.4.63 proc_MSS_GetManagedPropertiesForOM

The **proc_MSS_GetManagedPropertiesForOM** stored procedure is called to retrieve a list of managed properties from the **Managed Properties Set** which were added or modified on or after the date and time specified by *@ldate* parameter. For the specification of Managed Properties Set see Section 3.1.1.1.

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

```
PROCEDURE proc_MSS_GetManagedPropertiesForOM (
    @ldate              datetime
);
```

@ldate: Specifies the earliest per-item LastModifiedTime value 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:

3.1.4.63.1 Managed Properties Result Set

The Managed Properties result set contains information about managed properties added or updated on or after the date and time specified by *@ldate* parameter. The result set MUST contain zero or more rows each corresponding to a managed property.

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

```
PID                     int,
FriendlyName            nvarchar(64),
PropertyDescription     nvarchar(2048),
ManagedType             int,
FullTextQueriable       bit,
Retrievable             bit,
Scoped                  bit,
RespectPriority         bit,
RemoveDuplicates        bit,
NoDelete                bit,
NoMap                   bit,
HasMultipleValues       bit,
NoWordBreaker           bit,
NameNormalized          bit,
IncludeInMD5            bit,
Mapped                  bit,
QueryIndependentRank    bit,
UserFlags               int,
WordBreakerOverride     int,
Weight                  float,
LengthNormalization     float;
```

PID: The unique identifier of the managed property.

FriendlyName: The name of the managed property.

PropertyDescription: The description of the managed property.

ManagedType: The type of the managed property, a Managed Type Data Type as specified in Section 2.2.2.17**.**

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

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

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

RespectPriority: MUST be 1 if only data from the crawled property mapped to this managed property with highest priority of its **mapping order** is used. MUST be 0 if values from all crawled properties mapped to this managed property are used.

RemoveDuplicates: MUST be ignored by the client**.**

NoDelete: MUST be 1 if this property can ever be deleted from the metadata schema. Otherwise, it MUST be 0.

NoMap: MUST be 1 if this property can never have its crawled property mappings altered. Otherwise, it MUST be 0.

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

NoWordBreaker: MUST be ignored by the client.

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

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

Mapped: MUST be 1 when the property is a URL that subject to **alternate access mappings**. Otherwise, it MUST be 0.

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

UserFlags: The flag that can be retrieved and set by an administrator that is open to custom applications that use the public schema object model to get and set these.

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.

3.1.4.64 proc_MSS_GetManagedPropertyAliasesByPid

The **proc_MSS_GetManagedPropertyAliasesByPid** stored procedure is called to list the aliases for a managed property from the metadata schema**.**

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

```
PROCEDURE proc_MSS_GetManagedPropertyAliasesByPid(
    @PID                int
);
```

@PID: The unique identifier of the managed property.

Return Code Values: An integer which MUST be 0

Result Sets: MUST return the following result set:

3.1.4.64.1 Managed Property Aliases Result Set

The Managed Property Aliases result set contains information about aliases for a managed property. The result set MUST contain zero or more rows.

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

```
alias                 nvarchar(2048);
```

alias: An alternate string name which identifies a managed property.

3.1.4.65 proc_MSS_GetManagedPropertyDocsPerPidCount

The **proc_MSS_GetManagedPropertyDocsPerPidCount** stored procedure is called to get the count of items in the metadata index which have this managed property**.**

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

```
PROCEDURE proc_MSS_GetManagedPropertyDocsPerPidCount (
    @PID              int,
    @Limit            int,
    @FoundCount       int OUTPUT
);
```

@PID: The unique identifier of the managed property.

@Limit: This parameter MUST be ignored by the server.

@FoundCount: Upon return from this stored procedure, this parameter MUST be set to the count of items with the *@PID* managed property**.**

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.66 proc_MSS_GetManagedPropertySamples

The **proc_MSS_GetManagedPropertySamples** stored procedure is called to list sample values for a managed property from the metadata schema**.**

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

```
PROCEDURE proc_MSS_GetManagedPropertiesamples (
    @PID              int,
    @SampleCount      int
);
```

@PID: The unique identifier of the managed property.

@SampleCount: The maximum number of rows allowed in the result set**.**

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.66.1 Managed Property Samples Result Set

The Managed Property Samples result set contains information about sample values for a string type managed property ordered by the item URL. The result set MUST contain zero or more rows up to the maximum specified in *@SampleCount*.

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

```
strVal                nvarchar(2064);
```

strVal: a string property value of a item**.**

3.1.4.67 proc_MSS_GetMappedCrawledProperties

The **proc_MSS_GetMappedCrawledProperties** stored procedure is called to get a list of crawled properties mapped to a managed property in the metadata schema **Mappings Set.**

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

```
PROCEDURE proc_MSS_GetMappedCrawledProperties (
    @pid              int,
    @resultsCount     int
);
```

@PID: The unique identifier of the managed property.

@resultsCount: The maximum number of rows allowed in the result set**.**

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.67.1 Mapped Crawled Properties Result Set

The Mapped Crawled Properties result set contains information about crawled properties mapped to the specified managed property in the metadata schema **Mappings Set**. The result set MUST contain zero or more rows up to the maximum specified in *@resultsCount*, each corresponding to a single crawled property. The Mapped Crawled Properties result set is specified in Section 2.2.5.2.

3.1.4.68 proc_MSS_GetMappingsForCrawledProperty

The **proc_MSS_GetMappingsForCrawledProperty** stored procedure is called to get a list of managed properties mapped to a crawled property in the metadata schema **Mappings Set.**

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

```
PROCEDURE proc_MSS_GetMappingsForCrawledProperty (
    @Propset          uniqueidentifier,
    @PropertyName     nvarchar(440),
    @VariantType      int
```

```
);
```

@Propset**:** The **crawled property set identifier associated with the crawled property category**.

@PropertyName: The name of the crawled property.

@VariantType: The variant type for the crawled property.

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

Value	Description
0	Successful execution.
100	A crawled property with the specified type and name doesn't exist in the specified **property set.**

Result Sets: MUST NOT return any result sets if a crawled property with the specified type and name doesn't exist in the specified crawled property set identifier. Otherwise MUST return the following result set:

3.1.4.68.1 Crawled Property Mappings Result Set

The Crawled Property Mappings result set contains information about the managed properties to which the crawled property is mapped. The result set MUST contain zero or more rows sorted by *FriendlyName* in ascending order.

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

```
FriendlyName           nvarchar(64);
```

FriendlyName: The name of the the managed property in a user-readable form.

3.1.4.69 proc_MSS_GetMappingsForMangedProperty

The **proc_MSS_GetMappingsForMangedProperty** stored procedure is called to get list of crawled properties mapped to a managed property in the metadata schema **Mappings Set**.

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

```
PROCEDURE proc_MSS_GetMappingsForMangedProperty (
    @pid               int
);
```

@PID: The unique identifier of the managed property.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.69.1 Managed Property Mappings Result Set

The Managed Property Mappings result set contains information about the crawled properties mapped to the specified managed property. The result set MUST contain zero or more rows ordered by mapping order priority, each corresponding to a single of crawled property. If the managed property identifier is invalid, the result set MUST contain zero rows.

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

```
Propset                uniqueidentifier,
PropertyName           nvarchar(440),
VariantType            int;
```

Propset: **The crawled property set identifier associated with the crawled property category**.

PropertyName: The name of the **crawled property**.

VariantType: The variant type for the crawled property.

3.1.4.70 proc_MSS_GetNDayAvgCrawlHistoryStats

The proc**_MSS_GetNDayAvgCrawlHistoryStats** stored procedure is called to retrieve the average statistics for completed crawls of a specific type on a specific content source, within a specific past number of days. For the specified type of crawl, on the specified content source, it returns the average duration of the crawls, average success count, average error count and average warning count of items and the total number of crawls within the past *@NumberOfDays* days.

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

```
PROCEDURE proc_MSS_GetNDayAvgCrawlHistoryStats (
    @ContentSourceId          int,
    @CrawlType                int,
    @NumberOfDays             int
);
```

@ContentSourceId: The 32 bit unique identifier for the content source whose average statistics are to be retrieved.

@CrawlType: The type of Crawl which MUST be one of the integers listed in the table below:

Value	Description
1	**full crawl**.
2	**incremental crawl**.
6	**delete crawl**.

@NumberOfDays: The integer representing the past number of days for which the average statistics need to be calculated.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.70.1 Average Statistics Result Set

The Average Statistics result set returns the average duration, average success count, average error count and average warning count and total completed crawls of the specified type on the specified content source within the specified past number of days. The result set MUST contain one row to display the average and total counts. crawls that satisfy the input parameter conditions and have at least one non-zero success, error or warning count are used in the query.

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

```
DurationAvg            int,
AvgSuccess             int,
AvgError               int,
AvgWarning             int,
TotalCrawls            int;
```

DurationAvg: The average duration, in seconds, of the completed crawls.

AvgSuccess: The average number of items that were successfully crawl ed.

AvgError: The average number of items that generated errors when they were crawl ed.

AvgWarning: The average number of items that generated warnings when they were crawled.

TotalCrawls: The total number of crawls.

3.1.4.71 proc_MSS_GetPastLogData

The **proc_MSS_GetPastLogData** stored procedure is called to retrieve the list of errors and warnings logged for a specific display URL when the display URL was processed by the crawler

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

```
PROCEDURE proc_MSS_GetPastLogData (
    @Url               nvarchar(2048)
);
```

@Url: The display URL whose logged errors and warnings are to be retrieved.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.71.1 Log Data Result Set

The Log Data result set contains information about the errors and warnings logged for the specified display URL when the display URL was processed by the crawler. The result set MUST be sorted in ascending order by *LastTouchStart*. The result set MUST contain zero or more rows, each row corresponding to a single error or warning logged for the specified display URL.

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

```
DisplayUrl             nvarchar(1500),
ErrorID                int,
ErrorMsg               nvarchar(2000),
```

```
{NullValue}             null,
LastTouchStart          datetime;
```

DisplayUrl: The specified display URL.

ErrorID: The unique identifier for the type of error or warning that was encountered when processing the specified display URL.

ErrorMsg: The descriptive error or warning message for the *ErrorID*.

{NullValue}: This value MUST be ignored by the client.

LastTouchStart: The date and time the error or warning was logged.

3.1.4.72 proc_MSS_GetSchemaRankingParameters

The **proc_MSS_GetSchemaRankingParameters** stored procedure is called to get values of ranking parameters.

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

```
PROCEDURE proc_MSS_GetSchemaRankingParameters();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.72.1 Schema Parameters Result Set

The Schema Parameters result set contains information about ranking parameters. The result set MUST contain zero or more rows, each corresponding to a single ranking parameter.

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

```
ParamName               nvarchar(40),
IsString                bit,
strValue                nvarchar(256),
fltValue                float;
```

ParamName: Name of the schema parameter. Its value MUST be the following table.

Parameter	Description
k1	Saturation constant for term frequency.
Kqir	Saturation constant for **click distance**.
wqir	Weight of click distance for calculating relevance.
Kud	Saturation constant for URL depth.
wud	Weight of URL depth for calculating relevance.
languageprior	Weight for ranking applied to content in a language that does not match the

Parameter	Description
	language of the user.
filetypepriorhtml	Weight of HTML content type for calculating relevance.
filetypepriordoc	Weight of Microsoft Office Word content type for calculating relevance.
filetypepriorppt	Weight of Microsoft Office PowerPoint content type for calculating relevance.
filetypepriorxls	Weight of Microsoft Office Excel content type for calculating relevance.
filetypepriorxml	Weight of XML content type for calculating relevance.
filetypepriortxt	Weight of plain text content type for calculating relevance.
Filetypepriorlistitems	Weight of **list item** content type for calculating relevance.
Filetypepriormessage	Weight of Microsoft Office Outlook e-mail message content type for calculating relevance.
UseSortByRankIndex	An optimization for single-term search queries to use pre-calculated rank data.<47>

IsString: If set to 1, the schema parameter has a string value: the *strValue* field contains the value of the parameter; otherwise the parameter has a floating-point value and the value of the parameter is in the *fltValue* field.

strValue: A string value of the schema parameter. This field MUST be ignored when *IsString* equals 0.

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

3.1.4.73 proc_MSS_GetScopeDisplayGroupIDFromName

The **proc_MSS_GetScopeDisplayGroupIDFromName** stored procedure is called to retrieve the identifier of a search scope display group with the specified search scope display group name and the search scope consumer name.

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

```
PROCEDURE proc_MSS_GetScopeDisplayGroupIDFromName (
    @ConsumerName             nvarchar(60),
    @Name                     nvarchar(60),
    @DisplayGroupID           int OUTPUT
);
```

@ConsumerName: The name of the search scope consumer who is the owner of the search scope display group**.**

@Name: The name of the search scope display group**.**

@DisplayGroupID: Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the search scope display group if the search scope display group is found.

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

Value	Description
0	Successful execution.
1	The search scope display group or search scope consumer name does not exist.

Result Sets: MUST NOT return any result set.

3.1.4.74 proc_MSS_GetScopeDisplayGroupInfo

The **proc_MSS_GetScopeDisplayGroupInfo** stored procedure is called to retrieve information about a search scope display group.

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

```
ROCEDURE proc_MSS_GetScopeDisplayGroupInfo (
    @DisplayGroupID           int,
    @Name                     nvarchar(60) OUTPUT,
    @Description              nvarchar(300)OUTPUT,
    @ConsumerName             nvarchar(60)OUTPUT,
    @DisplayInAdminUI         bit OUTPUT,
    @Undeletable              bit OUTPUT,
    @DefaultScopeID           int OUTPUT,
    @LastModifiedTime         datetime OUTPUT,
    @LastModifiedBy           nvarchar(60) OUTPUT
);
```

@DisplayGroupID: The unique identifier of the search scope display group to be retrieved.

@Name: Upon return from this stored procedure, this parameter MUST be set to the name of the search scope display group.

@Description: Upon return from this stored procedure, this parameter MUST be set to the description of the search scope display group.

@ConsumerName: Upon return from this stored procedure, this parameter MUST be set to the name of the search scope consumer who is the owner of the search scope display group.

@DisplayInAdminUI: Upon return from this stored procedure, this parameter MUST be set to a bit flag indicating if the search scope display group is displayed in Administration user interface. The parameter value MUST be a **DisplayInAdminUI** data type as specified in Section 2.2.2.6.

@Undeletable: Upon return from this stored procedure, this parameter MUST be set to a bit flag indicating if the search scope display group can be deleted. The parameter value MUST be an **Undeletable** data type as specified in Section 2.2.2.22**.**

@DefaultScopeID: Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the default search scope for the search scope display group.

@LastModifiedTime: Upon return from this stored procedure, this parameter MUST be set to the date and time of the last change to the search scope display group.

@LastModifiedBy: Upon return from this stored procedure, this parameter MUST be set to the name of the user who last changed the search scope display group.

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

Value	Description
0	Successful execution.
1	The search scope display group does not exist.

Result Sets: MUST NOT return any result set.

3.1.4.75 proc_MSS_GetScopeDisplayGroupListInfo

The **proc_MSS_GetScopeDisplayGroupListInfo** stored procedure is called to retrieve a list of all search scopes within the specified search scope display group.

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

```
PROCEDURE proc_MSS_GetScopeDisplayGroupListInfo (
    @DisplayGroupID           int
);
```

@DisplayGroupID: The unique identifier of the search scope display group.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.75.1 Scopes Result Set

The Scopes Result set contains information about the search scopes that the specified search scope display group contains. The result set MUST contain zero or more rows, each corresponding to a single search scope. The rows in the result set MUST have non-negative order values and MUST be sorted by order value in ascending order.

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

```
ScopeID                int;
```

ScopeID: The unique identifier of the search scope.

3.1.4.76 proc_MSS_GetScopeDisplayGroupsCount

The **proc_MSS_GetScopeDisplayGroupsCount** stored procedure is called to retrieve the total number of search scope display groups.

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

```
PROCEDURE proc_MSS_GetScopeDisplayGroupsCount (
    @Count             int OUTPUT
);
```

@Count: Upon return from this stored procedure, this parameter MUST be set to the total number of search scope display groups.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.77 proc_MSS_GetScopeDisplayGroupsForConsumer

The **proc_MSS_GetScopeDisplayGroupsForConsumer** stored procedure is called to retrieve a list of all search scope display groups owned by the specified search scope consumer.

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

```
PROCEDURE proc_MSS_GetScopeDisplayGroupsForConsumer (
    @ConsumerName       nvarchar(60)
);
```

@ConsumerName: The name of the search scope consumer.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a single result set as specified in Section 2.2.5.3.

3.1.4.78 proc_MSS_GetScopeDisplayGroupsInfo

The **proc_MSS_GetScopeDisplayGroupsInfo** stored procedure is called to retrieve a list of all search scope display groups.

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

```
PROCEDURE proc_MSS_GetScopeDisplayGroupsInfo();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a single result set as specified in Section 2.2.5.3.

3.1.4.79 proc_MSS_GetScopeIDFromName

The **proc_MSS_GetScopeIDFromName** stored procedure is called to retrieve the identifier of a search scope with the specified name of the search scope and the name of the search scope consumer that owns the search scope.

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

```
PROCEDURE proc_MSS_GetScopeIDFromName (
    @ConsumerName       nvarchar(60),
    @Name               nvarchar(60),
    @ScopeID            int OUTPUT
);
```

@ConsumerName: The name of the search scope consumer that owns the search scope.

@Name: The name of the search scope.

@ScopeID: Upon return of this stored procedure, this parameter MUST be set to the unique identifier of the search scope if a search scope with specified name exists. Otherwise this parameter MUST be set to NULL.

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

Value	Description
0	Successful execution.
1	The search scope or search scope consumer name does not exist.

Result Sets: MUST NOT return any result sets.

3.1.4.80 proc_MSS_GetScopeInfo

The **proc_MSS_GetScopeInfo** stored procedure is called to retrieve the information about the search scope with the specified identifier.

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

```
PROCEDURE proc_MSS_GetScopeInfo(
    @ScopeID                        int,
    @Name                           nvarchar(60) OUTPUT,
    @Description                    nvarchar(300) OUTPUT,
    @ConsumerName                   nvarchar(60) OUTPUT,
    @DisplayInAdminUI               bit OUTPUT,
    @AlternateResultsPageURL        nvarchar(2048) OUTPUT,
    @CompilationType                int OUTPUT,
    @CompilationState               smallint OUTPUT,
    @LastCompilationTime            datetime OUTPUT,
    @LastModifiedTime               datetime OUTPUT,
    @LastModifiedBy                 nvarchar(60) OUTPUT
);
```

@ScopeID: The unique identifier of the search scope.

@Name: Upon return from this stored procedure, this parameter MUST be set to the name of the search scope.

@Description: Upon return from this stored procedure, this parameter MUST be set to the description of the search scope.

@ConsumerName: Upon return from this stored procedure, this parameter MUST be set to the name of the search scope consumer who owns the search scope.

@DisplayInAdminUI: Upon return from this stored procedure, this parameter MUST be set to a bit flag indicating if the search scope is displayed in the Administration UI. The value MUST be a valid DisplayInAdminUI data type as specified in Section 2.2.2.6.

@AlternateResultsPageUrl: Upon return from this stored procedure, this parameter is set to the alternate URL of the web page for the search scope. This parameter can be set to NULL.

@CompilationType: Upon return from this stored procedure, this parameter MUST be set to the compilation type of the search scope. The value MUST be a valid Compilation Type data type as specified in Section 2.2.2.5.

@CompilationState: Upon return from this stored procedure, this parameter MUST be set to the search scope compilation state of the search scope. The value MUST be a valid CompilationState data type as specified in Section 2.2.2.4.

@LastCompilationTime: Upon return from this stored procedure, this parameter is set to the date and time of the last search scope compilation. This parameter can be set to NULL

@LastModifiedTime: Upon return from this stored procedure, this parameter MUST be set to the date and time of the last change to the search scope.

@LastModifiedBy: Upon return from this stored procedure, this parameter MUST be set to the name of the person who last changed the search scope.

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

Value	Description
0	Successful execution.
1	search scope with the specified identifier was not found

Result Sets: MUST NOT return any result sets.

3.1.4.81 proc_MSS_GetScopeRuleInfo

The **proc_MSS_GetScopeRuleInfo** stored procedure is called to get the information about a search scope rule given its identifier.

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

```
PROCEDURE proc_MSS_GetScopeRuleInfo(
    @RuleID                 int,
    @FilterBehavior         smallint OUTPUT,
    @RuleType               smallint OUTPUT,
    @UrlRuleType            smallint OUTPUT,
    @PropertyID             int OUTPUT,
    @UserValueString        nvarchar(2048)OUTPUT
);
```

@RuleID: The unique identifier of the search scope rule.

@FilterBehavior: Upon return from this stored procedure, this parameter MUST be set to a valid value of a ScopeFilterBehavior data type as specified in Section 2.2.2.20.

@RuleType: Upon return from this stored procedure, this parameter MUST be set to a valid value of a ScopeRuleType data type as specified in Section 2.2.2.21.

@UrlRuleType: Upon return from this stored procedure, this parameter MUST be set to a valid value of an UrlRuleType data type as specified in Section 2.2.2.23.

@ PropertyID: Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the managed property associated with the search scope rule.

@UserValueString: Upon return from this stored procedure, this parameter MUST be set to the search scope rule value of the rule.

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

Value	Description
0	Successful execution.
1	Search scope rule with the specified identifier was not found.

Result Sets: MUST NOT return any result set.

3.1.4.82 proc_MSS_GetScopeRulesCount

The **proc_MSS_GetScopeRulesCount** stored procedure is called to get the count of search scope rules defined for a search scope in the search application..

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

```
PROCEDURE proc_MSS_GetScopeRulesCount (
    @ScopeID            int,
    @Count              int OUTPUT
);
```

@ScopeID: The unique identifier of the search scope.

@Count: Upon return from this stored procedure, this parameter MUST be set to the count of search scope rules defined for the search scope.

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

Value	Description
0	Successful execution.
1	Search scope with the specified identifier was not found.

Result Sets: MUST NOT return any result set.

3.1.4.83 proc_MSS_GetScopeRulesInfo

The **proc_MSS_GetScopeRulesInfo** stored procedure is called to get the details of all the search scope rules defined for a search scope in the search application.

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

```
PROCEDURE proc_MSS_GetScopeRulesInfo (
    @ScopeID            int
);
```

@ScopeID: The unique identifier of the search scope.

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

Value	Description
0	Successful execution.

Value	Description
1	Search scope with the specified identifier was not found in the search application.

Result Sets: MUST return a single result set as specified in Section 2.2.5.5.

3.1.4.84 proc_MSS_GetScopesCount

The **proc_MSS_GetScopesCount** stored procedure is called to get the count of all search scopes defined in the search application.

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

```
PROCEDURE proc_MSS_GetScopesCount(
    @Count              int OUTPUT
);
```

@Count: Upon return from this stored procedure, this parameter MUST be set to the count of search scopes defined in the search application.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.85 proc_MSS_GetScopesForConsumer

The **proc_MSS_GetScopesForConsumer** stored procedure is called to retrieve the list of search scopes that are owned by the specified search scope consumer.

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

```
PROCEDURE proc_MSS_GetScopesForConsumer (
    @ConsumerName       nvarchar(60)
);
```

@ConsumerName: The name of the search scope consumer.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a single result set as specified in Section 2.2.5.4.

3.1.4.86 proc_MSS_GetScopesInfo

The **proc_MSS_GetScopesInfo** stored procedure is called to get the details of all the search scopes defined in the search application.

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

```
PROCEDURE proc_MSS_GetScopesInfo ();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a single result set as specified in Section 2.2.5.4.

3.1.4.87 proc_MSS_GetScopesManagerInfo

The **proc_MSS_GetScopesManagerInfo** stored procedure is called to get the details of the search scopes system in the search application.

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

```
PROCEDURE proc_MSS_GetScopesManagerInfo (
    @AverageCompilationDuration     int OUTPUT,
    @CompilationScheduleType        smallint OUTPUT,
    @CustomCompilationSchedule      nvarchar(60) OUTPUT,
    @LastCompilationTime            datetime OUTPUT,
    @NextCompilationTime            datetime OUTPUT,
    @CompilationState               int OUTPUT,
    @CompilationStartTime           datetime OUTPUT,
    @CompilationPercentComplete     smallint OUTPUT,
    @ScopesNeedingCompilation       int OUTPUT
);
```

@AverageCompilationDuration: Upon return from this stored procedure, this parameter MUST be set to the average compilation time for search scopes managed by the search scopes system.

@CompilationScheduleType: Upon return from this stored procedure, this parameter MUST be set to the Compilation Schedule Type of the search scopes system. The value MUST be a CompilationScheduleType data type as specified in Section 2.2.2.3.

@CustomCompilationSchedule: MUST be ignored by the client.

@LastCompilationTime: Upon return from this stored procedure, this parameter is set to the date and time of the last compilation of the search scopes system. This value can be set to NULL.

@ NextCompilationTime: Upon return from this stored procedure, this parameter MUST be set to the date and time of the next compilation of the search scopes system.

@CompilationState: Upon return from this stored procedure, this parameter MUST be set to the search scope compilation state of the search scopes system. The value MUST be a **CompilationState** data type as specified in Section 2.2.2.4.

@CompilationStartTime: Upon return from this stored procedure, this parameter is set to the date and time of the current compilation of the search scopes system. This value can be NULL.

@CompilationPercentComplete: Upon return from this stored procedure, this parameter MUST be set to the percentage of compilation completed for the search scopes system.

@ScopesNeedingCompilation: Upon return from this stored procedure, this parameter MUST be set to the number of search scopes belonging to the search scopes system that need compilation.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.88 proc_MSS_GetSharepointLocationVisualisations

The **proc_MSS_GetSharepointLocationVisualisations** stored procedure is called to retrieve the preselected visualizations of a federated location.

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

```
PROCEDURE proc_MSS_GetSharepointLocationVisualisations(
    @LocationId               int
);
```

@LocationId: The identifier of the federated location whose visualizations will be retrieved.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return zero, one, two or three result sets in the following order:

3.1.4.88.1 Top Answer Visualisation Result Set

The Top Answer Visualisation result set contains information about visualizations of the specified federated location. The result set MUST contain zero or one rows.

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

```
LocationId              int,
VisualisationName       nvarchar(60),
Xsl                     ntext,
Properties              ntext,
SampleData              ntext;
```

LocationId: The identifier of the specified federated location.

VisualisationName: MUST be equal to "topanswer".

Xsl: The Xsl for this visualization. This MUST be an **Xsl** Data Type as specified in Section 2.2.2.24.

Properties: The properties for this visualization. This MUST be a **Properties** Data Type as specified in Section 2.2.2.18.

SampleData: The sample data for this visualization. This MUST be a **Sample Data** Data Type as specified in Section 2.2.2.19.

3.1.4.88.2 Summary Results Visualisation Result Set

The Summary Results Answer Visualisation result set contains information about visualizations of the specified federated location. The result set MUST contain zero or one rows.

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

```
LocationId              int,
VisualisationName       nvarchar(60),
Xsl                     ntext,
Properties              ntext,
SampleData              ntext;
```

LocationId: The identifier of the specified federated location.

VisualisationName: MUST be equal to "summary".

Xsl: The Xsl for this visualization. This MUST be an **Xsl** Data Type as specified in Section 2.2.2.24.

Properties: The properties for this visualization. This MUST be a **Properties** Data Type as specified in Section 2.2.2.18.

SampleData: The sample data for this visualization. This MUST be a **Sample Data** Data Type as specified in Section 2.2.2.19.

3.1.4.88.3 Core Results Visualisation Result Set

The Core Results Answer Visualisation result set contains information about visualizations of the specified federated location. The result set MUST contain zero or one rows.

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

```
LocationId              int,
VisualisationName       nvarchar(60),
Xsl                     ntext,
Properties              ntext,
SampleData              ntext;
```

LocationId: The identifier of the specified federated location.

VisualisationName: MUST be equal to "full".

Xsl: The Xsl for this visualization. This MUST be an **Xsl** Data Type as specified in Section 2.2.2.24.

Properties: The properties for this visualization. This MUST be a Properties Data Type as specified in Section 2.2.2.18.

SampleData: The sample data for this visualization. This MUST be a Sample Data Data Type as specified in Section 2.2.2.19.

3.1.4.89 proc_MSS_GetSpecialTerm

The **proc_MSS_GetSpecialTerm** stored procedure is called to retrieve the information about a keyword from the site collection given its keyword consumer group**,** and the keyword term.

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

```
PROCEDURE proc_MSS_GetSpecialTerm (
    @ConsumerGpId        nvarchar(50),
    @Term                nvarchar(100)
);
```

@ ConsumerGpId: The unique identifier of the keyword consumer group.

@Term: The term for the keyword.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a single result set as specified in Section 2.2.5.6.

3.1.4.90 proc_MSS_GetSpecialTerms

The **proc_MSS_GetSpecialTerms** stored procedure is called to retrieve the list of keywords from the site collection for a given keyword consumer group. The list can be filtered according to criteria specified in *@View* and *@Filter* parameters.

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

```
PROCEDURE proc_MSS_GetSpecialTerms (
    @ConsumerGpId       nvarchar(50),
    @View               int,
    @Filter             int,
    @Value              nvarchar(2048) = null
);
```

@ConsumerGpId: The unique identifier for the keyword consumer group.

@View: The type of keywords to include in the result set. The value MUST be a Keyword Type data type as specified in Section 2.2.2.9.

@Filter: A Keyword Filter Type that specifies which keyword attributes are compared against the value given in parameter @Value. The value MUST be a Keyword Filter Type as specified in Section 2.2.2.10.

@**Value**: The value that is compared against the Term or Contact elements in the Keyword Set as specified in the Section 3.1.1.2, when performing the filtering. The value MUST be set to NULL to list all keywords for a given keyword consumer group.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.90.1 Special Terms Result Set

The Special Terms result set contains information about the keywords for a given keyword consumer group. The result set MUST contain zero or more rows, each corresponding to a keyword. If *@View* parameter is set to 2 the result set MUST be sorted by *ReviewDate* in ascending order; otherwise the result set MUST be sorted by *EndDate* in ascending order. For the specification of the result set see Section 2.2.5.6.

3.1.4.91 proc_MSS_GetSpecialTermsCount

The **proc_MSS_GetSpecialTermsCount** stored procedure is called to calculate the number of keywords for a given keyword consumer group. The keywords included can be filtered according to criteria specified in *@View* and *@Filter* parameters.

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

```
PROCEDURE proc_MSS_GetSpecialTermsCount (
    @ConsumerGpId       nvarchar(50),
    @View               int,
    @Filter             int,
    @Value              nvarchar(2048) = null,
    @Count              int output
);
```

@ConsumerGpId: A string that uniquely identifies the keyword consumer group.

@View: A 32 bit integer that specifies the type of keywords to include in the count. The value MUST be of type Keyword Type as specified in Section 2.2.2.9.

@Filter: A 32 bit integer that specifies which keyword attributes are compared against the value given in parameter *@Value*. The value MUST be of type Keyword Filter Type as specified in Section 2.2.2.10.

@**Value**: A string that is compared against when performing the filtering. The value MUST be set to NULL to list all keywords for a given keyword consumer group.

@Count: Upon return from this stored procedure, this parameter MUST be set to the number of keywords.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.92 proc_MSS_GetSpecialTermsCountForBestBet

The **proc_MSS_GetSpecialTermsCountForBestBet** stored procedure is called to retrieve the number of keywords that are associated with a given best bet.

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

```
PROCEDURE proc_MSS_GetSpecialTermsCountForBestBet (
    @BestBetId          int,
    @Count              int output
);
```

@BestBetId: The unique identifier of the best bet.

@ Count: Upon return from this stored procedure, this parameter MUST be set to the number of keywords.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.93 proc_MSS_GetSpecialTermsForBestBet

The **proc_MSS_GetSpecialTermsForBestBet** stored procedure is called to retrieve the list of keywords associated with a given best bet.

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

```
PROCEDURE proc_MSS_GetSpecialTermsForBestBet (
    @BestBetId          int
);
```

@BestBetId: The unique identifier of the best bet.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a single result set as specified in Section 2.2.5.6.

3.1.4.94 proc_MSS_GetSummaryByHost

The **proc_MSS_GetSummaryByHost** stored procedure is called to retrieve the success count, error count, warning count and total count of items processed within every host name.

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

```
PROCEDURE proc_MSS_GetSummaryByHost (
    @Order              int,
    @Count              int,
    @Dir                bit
);
```

@Order: The ordering of the results which MUST be one of the integers listed in the table below.

Value	Description
0	The result set is ordered by host name.
1	The result set is ordered by success count.
2	The result set is ordered by warning count.
3	The result set is ordered by error count.
4	The result set is ordered by total count.

@Count: The number of rows displayed in the result set. *@Count* **+ 1 is the maximum number of rows displayed in the result set.**

@Dir: The direction of sorting of the result set which MUST be listed in the table below.

Value	Description
0	The result set is sorted in ascending order.
1	The result set is sorted in descending order.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result sets:

3.1.4.94.1 Start At Result Set

The StartAt result set contains information about the next start at row number for the host summaries if there are more host summaries than restricted by the value of *@Count* +1. Otherwise, the result set returns -1. The result set MUST contain one row.

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

```
{StartAt}               int;
```

{StartAt}: The next start at row number for the host summaries when there are more host summaries available than displayed in the host summary result set. It is -1 when all the host summaries have been displayed in the host summary result set.

3.1.4.94.2 Host Summary Result Set

The Host Summary result set contains information about the host names processed and their respective log summaries. The ordering of the rows depends on the combination of the input parameters *@Order* and *@Dir*. The result set MUST contain as many rows as there are host names processed or the value of the *@Count* + 1 as described above, whichever is smaller.

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

```
HostName              nvarchar(300),
SuccessCount          int,
WarningCount          int,
ErrorCount            int,
TotalCount            int;
```

HostName: The host name.

SuccessCount: The total number of items within the host that were successfully processed.

WarningCount: The total number of items within the host that generated warnings when they were processed.

ErrorCount: The total number of items within the host that generated errors when they were processed.

TotalCount: The total sum of *SuccessCount*, *WarningCount* and *ErrorCount.*.

3.1.4.95 proc_MSS_GetSummaryLogData

The **proc_MSS_GetSummaryLogData** stored procedure is called to retrieve the count of successes, warnings and errors logged for display URLs when they were processed by the crawler for a specific **content source with a specific project identifier** data type as specified in [MS-SQLPGAT].

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

```
PROCEDURE proc_MSS_GetSummaryLogData (
    @CatalogID              int = null,
    @ContentSourceID        int = null,
    @Successes              int OUTPUT,
    @Errors                 int OUTPUT,
    @Warnings               int OUTPUT
);
```

@CatalogID: MUST be 1.

@ContentSourceID: The identifier for the content source whose success, warning and error counts are to be retrieved**.**

@Successes: Upon return from this stored procedure, this parameter MUST be set to the total number of items within the **Content Source** that were successfully Crawled if the **Content Source** is found; otherwise the value MUST be set to 0.

@Errors: Upon return from this stored procedure, this parameter MUST be set to the total number of items within the **Content Source** that generated errors when they were Crawl ed if the **Content Source** is found; otherwise the value MUST be set to 0.

@Warnings: Upon return from this stored procedure, this parameter MUST be set to the total number of items within the **Content Source** that generated warnings when they were Crawl ed if the **Content Source** is found; otherwise the value MUST be set to 0.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.96 proc_MSS_GetSynonym

The **proc_MSS_GetSynonym** stored procedure is called to retrieve from the site collection the information about the synonym associated with a given keyword.

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

```
PROCEDURE proc_MSS_GetSynonym(
    @SpecialTermId          int,
    @Term                   nvarchar(100)
);
```

@SpecialTermId: The unique identifier of the keyword**.**

@Term: The keyword.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a single result set as specified in Section 2.2.5.7.

3.1.4.97 proc_MSS_GetSynonyms

The **proc_MSS_GetSynonyms** stored procedure is called to retrieve from the site collection the list of synonyms associated with a given keyword.

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

```
PROCEDURE proc_MSS_GetSynonyms(
    @SpecialTermId     int
);
```

@SpecialTermId: The unique identifier of the keyword**.**

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a single result set as specified in Section 2.2.5.7.

3.1.4.98 proc_MSS_GetSynonymsCount

The **proc_MSS_GetSynonymsCount** stored procedure is called to calculate the number of synonyms associated with a given keyword.

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

```
PROCEDURE proc_MSS_GetSynonymsCount(
    @SpecialTermId            int,
    @Count                    int output
);
```

@SpecialTermId: The unique identifier of the keyword**.**

@Count: Upon return from this stored procedure, this parameter MUST be set to the number of keyword synonyms.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.99 proc_MSS_GetUnusedScopesForConsumer

The **proc_MSS_GetUnusedScopesForConsumer** stored procedure is called to retrieve the information about all search scopes owned by the specified search scope consumer that are not associated with any search scope display group.

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

```
PROCEDURE proc_MSS_GetUnusedScopesForConsumer (

    @ConsumerName      nvarchar(60)
);
```

@**ConsumerName**: The name of the search scope consumer.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a single result set as specified in Section 2.2.5.4.

3.1.4.100 proc_MSS_GetUsedMessages

The **proc_MSS_GetUsedMessages** stored procedure is called to retrieve the description of unique successes, warnings and errors encountered by the crawler when it processed items from the **crawl queue**.

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

```
PROCEDURE proc_MSS_GetUsedMessages();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.100.1 Used Messages Result Set

The Used Messages result set contains information about unique successes, warnings and errors that were encountered by the crawler when it processed items from the crawl queue. The result set MUST contain zero or more rows, each row corresponding to a single unique success, warning or error logged by the crawler.

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

```
ErrorID                int,
ErrorMsg               nvarchar(2000),
Hrresult               int,
ErrorLevel             int;
```

ErrorID: The unique identifier for the type of error that was encountered when an item was processed.

ErrorMsg: The descriptive error message for the unique error identifier *ErrorID*.

Hrresult: The HRESULT value of the error.

ErrorLevel: The error level which MUST be an ErrorLevel data type as specified in Section 2.2.2.7.

3.1.4.101 proc_MSS_GetVisibleScopesCount

The **proc_MSS_GetVisibleScopesCount** stored procedure is called to get the count of all visible scopes defined in the search application.

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

```
PROCEDURE proc_MSS_GetVisibleScopesCount (
    @Count             int OUTPUT
);
```

@Count: Upon return from this stored procedure, this parameter MUST be set to the count of visible scopes defined in the search application.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.102 proc_MSS_GetVolatileScopeInfo

The **proc_MSS_GetVolatileScopeInfo** stored procedure is called to get the details likely to change for a specified search scope.

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

```
PROCEDURE proc_MSS_GetVolatileScopeInfo (
    @ScopeID                int,
    @CompilationState       smallint OUTPUT,
    @LastCompilationTime    datetime OUTPUT
);
```

@ScopeID: The unique identifier of the search scope.

@CompilationState: Upon return from this stored procedure, this parameter MUST be set to the search scope compilation state of the search scope. The value MUST be a CompilationState data type as specified in Section 2.2.2.4.

@LastCompilationTime: Upon return from this stored procedure, this parameter is set to the Date and Time of the last compilation of the search scope. This value can be set to NULL.

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

Value	Description
0	Successful execution.
1	search scope with the specified identifier was not.

Result Sets: MUST NOT return any result set.

3.1.4.103 proc_MSS_GetVolatileScopesManagerInfo

The **proc_MSS_GetVolatileScopesManagerInfo** stored procedure is called to get the details likely to change for a search scopes system in the search application.

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

```
PROCEDURE proc_MSS_GetVolatileScopesManagerInfo (
   @AverageCompilationDuration     int OUTPUT,
   @LastCompilationTime            datetime OUTPUT,
   @NextCompilationTime            datetime OUTPUT,
   @CompilationState               int OUTPUT,
   @CompilationStartTime           datetime OUTPUT,
   @CompilationPercentComplete     smallint  OUTPUT,
   @ScopesNeedingCompilation       int OUTPUT
);
```

@AverageCompilationDuration: Upon return from this stored procedure, this parameter MUST be set to the average compilation time for search scopes inthe search scopes system.

@LastCompilationTime: Upon return from this stored procedure, this parameter is set to the date and time of the last compilation of the search scopes system. This value can be NULL.

@ NextCompilationTime: Upon return from this stored procedure, this parameter MUST be set to the date and time of the next compilation of the search scopes system.

@CompilationState: Upon return from this stored procedure, this parameter MUST be set to the search scope compilation state of the search scopes system. The value MUST be a CompilationState Data type as specified in Section 2.2.2.4.

@CompilationStartTime: Upon return from this stored procedure, this parameter is set to the date and time of the current compilation of the search scopes system. This value can be NULL.

@CompilationPercentComplete: Upon return from this stored procedure, this parameter MUST be set to the percentage of compilation completed for the search scopes system.

@ScopesNeedingCompilation: Upon return from this stored procedure, this parameter MUST be set to the number of search scopes belonging to the search scopes system.that need compilation.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.104 proc_MSS_PurgePastCrawlLog

The **proc_MSS_PurgePastCrawlLog** stored procedure is called to delete all the warnings and errors logged for all the display URLs when they were processed by the crawler.

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

```
PROCEDURE proc_MSS_PurgePastCrawlLog ();
```

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<48> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.105 proc_MSS_PutLocationVisualisation

The **proc_MSS_PutLocationVisualisation** stored procedure is called to add a new visualization definition for a federated location.

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

```
PROCEDURE proc_MSS_PutLocationVisualisation(
   @LocationId                int,
   @VisualisationName         nvarchar(60),
   @Xsl                       ntext,
   @Properties                ntext,
   @SampleData                ntext,
   @DeleteAllBeforePut        bit
);
```

@LocationId: The identifier of the federated location for which the visualization will be added.

@VisualisationName: The name of the visualization that will be added.

@Xsl: The Xsl data of the visualization. The value MUST be a Xsl Data Type as specified in Section 2.2.2.24.

@Properties: The properties data of the visualization. The value MUST be a Properties Data Type as specified in Section 2.2.2.18.

@SampleData: The sample data of the visualization. The value MUST be a SampleData Data Type as specified in Section 2.2.2.19.

@DeleteAllBeforePut: A 1-bit integer that indicates if all the existing visualizations for the specified federated location MUST be deleted before adding the new visualization. MUST be 1 if all visualizations for the Federated Location MUST be deleted first, otherwise 0.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.106 proc_MSS_RemoveFilenameFromResults

The **proc_MSS_RemoveFilenameFromResults** stored procedure is called to remove a file from the metadata index so it will not appear in the search result.

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

```
PROCEDURE proc_MSS_RemoveFilenameFromResults (
    @File               nvarchar(2000),
    @FileHash           int
);
```

@File: The name of the file.

@FileHash: The identifier of the file name.

Return Code Values: An integer which MUST be 0 .

Result Set: SHOULD NOT<49> return a result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.107 proc_MSS_SetCrawledCategoryPropertiesAllOM

The **proc_MSS_SetCrawledCategoryPropertiesAllOM** stored procedure is called to change data for the specified **crawled property category** in the **metadata schema**. This procedure also updates value of the CrawledPropertyCategoryTimestamp in the Metadata Timestamp Set as specified in Section 3.1.1.1 with the current date and time in local time of the server.

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

```
PROCEDURE proc_MSS_SetCrawledCategoryPropertiesAllOM (
    @Name                           nvarchar(64),
    @NewName                        nvarchar(64),
    @DiscoverNewProperties          bit,
    @MapToContents                  bit,
    @AutoCreateManagedProperties    bit
);
```

@Name: The name of the **crawled property category** to be updated.

@NewName: If not equal to *@Name* and if the *@NewName* does not already exist in the Crawled Property Category Set, specified in Section 3.1.1.1, then the **crawled property** name is updated with *@NewName*.

@DiscoverNewProperties: MUST be 1 if the **crawled properties** within this **crawled property** are added automatically to the Crawled Properties Set, specified in Section 3.1.1.1. Otherwise, it MUST be 0.

@MapToContents: MUST be 1 if string data from newly discovered **crawled properties** within this **crawled property** is put in the **full-text index catalog**. Otherwise, it MUST be 0.

@AutoCreateManagedProperties: MUST be 1 if string data from newly discovered **crawled properties** within this **crawled property** is mapped to a new **managed property** which will be put in the **metadata index** and in the **full-text index catalog.** Otherwise, it MUST be 0.

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

Value	Description
0	Successful execution.
1	New name for the **crawled property** already exists.

Result Sets: SHOULD NOT<50> return any result sets. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.108 proc_MSS_SetCrawledPropertyMapToContent

The **proc_MSS_SetCrawledPropertyMapToContent** stored procedure is called to update *@IsMappedToContent* flag for the specified crawled property. This procedure also updates the value of CrawledPropertyTimestamp, , specified in Section 3.1.1.1, in the Metadata Timestamp Set and the value of LastModifiedTime attribute in the Crawled Property Set for the corresponding crawled property with the local time of the server.

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

```
PROCEDURE proc_MSS_SetCrawledPropertyMapToContent (
    @crawledPropId          int,
    @IsMappedToContent      bit
);
```

@crawledPropId: Unique identifier of the crawled property.

@IsMappedToContent: When the value is **set to** 1 and the variant type is a string, the data from this crawled property will be put in the full-text index catalog. Otherwise, it MUST be 0. Upon successful execution, the *IsMappedToContent* attribute of the crawled property will be set to this value.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<51> return any result sets. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.109 proc_MSS_SetManagedPropertyAllOM

The **proc_MSS_SetManagedPropertyAllOM** stored procedure is called to set all attributes for a managed property in the metadata schema. This procedure also updates the LastModifyTime attribute in Managed Property Set, specified in Section 3.1.1.1, and the value of the ManagedPropertyAddModifyTimestamp in the Metadata Timestamp Set with the current date and time in local time of the server.

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

```
PROCEDURE proc_MSS_SetManagedPropertyAllOM (
    @PID                    int,
    @Name                   nvarchar(440),
```

```
    @Description            nvarchar(2048),
    @ManagedType            int,
    @FullTextQueriable      bit,
    @Retrievable            bit,
    @EnabledForScoping      bit,
    @RespectPriority        bit,
    @RemoveDuplicates       bit,
    @NoWordBreaker          bit,
    @IsNameNormalized       bit,
    @IncludeInMd5           bit,
    @Mapped                 bit,
    @QueryIndependentRank   bit,
    @UserFlags              smallInt,
    @WordBreakerOverrride   int,
    @Weight                 float,
    @LengthNormalization    float
);
```

@PID: The unique identifier of the managed property.

@Name: The new name for the managed property. Upon successful execution, the name attribute of the managed property MUST be set to this value.

@Description: Description of the managed property. Upon successful execution, the description attribute of the managed property MUST be set to this value.

@ManagedType: The type of the managed property as specified in Section 2.2.2.17**.** Upon successful execution, this attribute of the managed property MUST be set to this value.

@FullTextQueriable: MUST be 1 if the data for the managed property is kept in the full-text index catalog, Otherwise, it MUST be 0. Upon successful execution, the FullTextQueriable attribute of the managed property MUST be set to this value.

@Retrievable: MUST be 1 if the data for the managed property is kept in the metadata index. Otherwise, it MUST be 0. Upon successful execution, the Retrievable attribute of the managed property MUST be set to this value.

@EnabledForScoping: MUST be 1 if the data for the managed property is kept in the search scope index**.** Otherwise, it MUST be 0. Upon successful execution, the EnableForScoping attribute of the managed property MUST be set to this value.

@RespectPriority: MUST be 1 if only data from the crawled property mapped to this managed property with highest priority of its mapping order is used. It MUST be 0 if values from all crawled properties mapped to this managed property are used. Upon successful execution, the RespectPriority attribute of the managed property MUST be set to this value.

@RemoveDuplicates: MUST be ignored by the server.

@NoWordBreaker: MUST be ignored by the server.

@IsNameNormalized: MUST be 1 if the values of this managed property is normalized by the crawler. Otherwise, it MUST be 0. Upon successful execution, the NameNormalized attribute of the managed property MUST be set to this value.

@IncludeInMD5: MUST be 1 if values mapped to this managed property are used to determine if the item has changed. Otherwise, it MUST be 0. Upon successful execution, the includeMD5 attribute of the managed property MUST be set to this value.

@Mapped: MUST be 1 when the property is an URL that subject to alternate access mappings. Otherwise, it MUST be 0. Upon successful execution, the Mapped attribute of the managed property MUST be set to this value.

@QueryIndependentRank: MUST be 0.

@UserFlags: A 16 bit field that can be retrieved and set by an administrator that is open to custom applications that use the public schema object model to get and set these. Upon successful execution, the UserFlags attribute of the managed property MUST be set to this value.

@WordBreakerOverrride: MUST be ignored by the server.

@Weight: A decimal value used to adjust property oriented rank. Upon successful execution, the Weight attribute of the managed property MUST be set to this value.

@LengthNormalization: A decimal value used to adjust property oriented rank. Upon successful execution, the LengthNormalization attribute of the managed property MUST be set to this value.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<52> return any result sets. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.110 proc_MSS_SetManagedPropertyHasMultipleValues

The **proc_MSS_SetManagedPropertyHasMultipleValues** stored procedure is called to specify that the given managed property can contain multiple values for a single item. This procedure also updates the LastModifyTime attribute in Managed Property Set and the value of the ManagedPropertyAddModifyTimestamp in the Metadata Timestamp Set with the current date and time in local time of the server.

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

```
PROCEDURE proc_MSS_SetManagedPropertyHasMultipleValues (
    @PID                    int,
    @HasMultipleValues      bit
);
```

@PID: The unique identifier of the managed property.

@HasMultipleValues: 1 indicates that the managed property can have multiple values for a single item; otherwise it MUST be 0. Upon successful execution, the **HasMultipleValues** attribute of the managed property MUST be set to this value.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<53> return any result sets. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.111 proc_MSS_SetPendingMappings

The following data types are as specified in section 3.1.1.1. The **proc_MSS_SetPendingMappings** stored procedure is called to copy mappings between the specified managed property and crawled properties from the Pending Mappings Set , to the Mappings Set. This procedure updates the LastModifyTime attribute in Crawled Property Set for the corresponding crawled properties entries

and the value of the CrawledPropertyTimestamp in the Metadata Timestamp Set with the current date and time in local time of the server.

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

```
PROCEDURE proc_MSS_SetPendingMappings (
    @PID                int
);
```

@PID: The unique identifier of the managed property.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<54> return any result sets. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.112 proc_MSS_SetSchemaParameter

The **proc_MSS_SetSchemaParameter** stored procedure is called to set the specified schema parameter. This procedure also changes the Global Ranking Parameters Timestamp attribute in the Metadata Timestamp Set as specified in 3.1.1.1 to the current date and time in local time of the server.

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

```
PROCEDURE proc_MSS_SetSchemaParameter (
    @ParamName          nvarchar(40),
    @IsString           bit,
    @strValue           nvarchar(256),
    @fltValue           float
);
```

@ParamName: The name of the schema parameter.

@IsString: If set to 1, the *@strValue* parameter is used and the *@fltvalue* parameter is not used. Otherwise, the *@fltValue* parameter is used and the *@strValue* parameter is not used.

@strValue: A string value of the schema parameter.

@fltValue: A floating-point value of the parameter.

Return Code Values: An integer which MUST be 0.

Result Set: SHOULD NOT<55> return a result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.113 proc_MSS_SetScopeDisplayGroupInfo

The **proc_MSS_SetScopeDisplayGroupInfo** stored procedure is called to set the configuration information for the specified search scope display group.

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

```
PROCEDURE proc_MSS_SetScopeDisplayGroupInfo (
    @DisplayGroupID           int,
```

```
    @Name                   nvarchar(60),
    @Description            nvarchar(300),
    @DisplayInAdminUI       bit,
    @Undeletable            bit,
    @DefaultScopeID         int,
    @ModifierName           nvarchar(60)
);
```

@DisplayGroupID: The unique identifier of the search scope display group.

@Name: The name of the search scope display group.

@Description: The description of the search scope display group.

@DisplayInAdminUI: The bit flag indicating whether the search scope display group is displayed in the Administration user interface. The value MUST be a **DisplayInAdminUI** data type as specified in Section 2.2.2.6**.**

@Undeletable: The bit flag indicating whether the search scope display group can be deleted. The value MUST be an **Undeletable** data type as specified in Section 2.2.2.22**.**

@DefaultScopeID: The default search scope of the search scope display group.

@ModifierName: Name of the user who last modified the search scope display group.

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

Value	Description
0	Successful execution.
1	The search scope display group does not exist.

Result Sets: SHOULD NOT<56> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.114 proc_MSS_SetScopeDisplayGroupListItem

The **proc_MSS_SetScopeDisplayGroupListItem** stored procedure is called to add the search scope to the specified search scope display group with the specified order. The stored procedure MUST be called after the call to proc_MSS_BeginScopeDisplayGroupList stored procedure. See section 3.1.4.17 for the specification of proc_MSS_BeginScopeDisplayGroupList stored procedure.

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

```
PROCEDURE proc_MSS_SetScopeDisplayGroupListItem (
    @DisplayGroupID         int,
    @ScopeID                int,
    @Rank                   int
);
```

@DisplayGroupID: The unique identifier of the search scope display group.

@ScopeID: The unique identifier of the search scope.

@Rank: The order of the search scope within the search scope display group. This parameter MUST be set to a negative value minus 1.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<57> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.115 proc_MSS_SetScopeInfo

The **proc_MSS_SetScopeInfo** stored procedure is called to set the details of the specified search scope in the search application. Upon successful execution the stored procedure increments the LastChangeId of the search application and assigns it to the LastChangeId of the added search scope as specified in Section 2.2.2.11.

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

```
PROCEDURE proc_MSS_SetScopeInfo (
    @ScopeID                        int,
    @Name                           nvarchar(60),
    @Description                    nvarchar(300),
    @ConsumerName                   nvarchar(60),
    @DisplayInAdminUI               bit,
    @AlternateResultsPageURL        nvarchar(2048),
    @CompilationType                int,
    @ModifierName                   nvarchar(60)
);
```

@ScopeID: The unique identifier of the search scope.

@Name: The name of the search scope. Upon successful execution, the name of the search scope MUST be set to this value.

@Description: The description of the search scope. Upon successful execution, the description of the search scope MUST be set to this value.

@ConsumerName: The name of the search scope consumer of the search scope. Upon successful execution, the search scope consumer name of the search scope MUST be set to this value.

@DisplayInAdminUI: The bit flag indicating whether the search scope display group is displayed in the Administration user interface. The value MUST be a DisplayInAdminUI data type as specified in Section 2.2.2.6.

@AlternateResultsPageUrl: The URL of an alternate web page to display the results of a search performed on this search scope. This value can be NULL.

@CompilationType: The compilation type for the search scope. The value MUST be a CompilationState data type as specified in Section 2.2.2.4. Upon successful execution, the CompilationState of the search scope MUST be set to this value.

@ModifierName: The name of the person modifying the search scope.

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

Value	Description
0	Successful execution. The search scope was successfully updated.
1	Search scope with the specified identifier was not found in the search application. The search scope was not updated.
2627	The search scope with the specified name and search scope consumer already exists

Result Sets: SHOULD NOT<58> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.116 proc_MSS_SetScopesManagerInfo

The **proc_MSS_SetScopesManagerInfo** stored procedure is called to set the details of the search scopes system in the search application.

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

```
PROCEDURE proc_MSS_SetScopesManagerInfo
    @CompilationScheduleType        smallint,
    @CustomCompilationSchedule      nvarchar(60)
);
```

@CompilationScheduleType: The search scope compilation schedule type for the search scopes system. The value MUST be a CompilationScheduleType data type as specified in Section 2.2.2.3. Upon successful execution, the compilation schedule type of the search scopes system MUST be set to this value.

@CustomCompilationSchedule: MUST be ignored by the client..

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<59> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.117 proc_MSS_SetScopeRuleInfo

The **proc_MSS_SetScopeRuleInfo** stored procedure is called to set the details of the specified search scope rule in the search application. The stored procedure upon successful execution increments the LastChangeId of the search application and assigns it to the LastChangeId of the search scope associated with the specified search scope rule as specified in Section 2.2.2.11.

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

```
PROCEDURE proc_MSS_SetScopeRuleInfo (
    @RuleID                 int,
    @FilterBehavior         smallint,
    @UrlRuleType            smallint,
    @PropertyID             int,
    @UserValueString        nvarchar(2048),
    @ModifierName           nvarchar(60)
 );
```

@RuleID: The unique identifier of the search scope rule**.**

@FilterBehavior: The filter behavior of the search scope rule. The value MUST be a ScopeFilterBehavior **d**ata type as specified in Section 2.2.2.20. Upon successful execution, the filter behavior of the search scope rule MUST be set to this value.

@UrlRuleType: The URL type of the search scope rule. The value MUST be a UrlRuleType Data type as specified in Section 2.2.2.23. Upon successful execution, the url rule type of the search scope MUST be set to this value.

@ PropertyID: The unique identifier of the the managed property that is associated with the search scope rule. Upon successful execution, the **property identifier** of the search scope rule MUIST be set to this value.

@UserValueString: A user value string for the search scope rule. Upon successful execution, the user value string of the search scope rule MUST be set to this value.

@ ModifierName: The name of the person modifying the search scope rule.

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

Value	Description
0	Successful execution. The search scope rule was successfully updated.
1	Search scope rule with the specified identifier was not found in the search application. The search scope rule was not updated.

Result Sets: SHOULD NOT<60> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.118 proc_MSS_StartScopesCompilation

The **proc_MSS_StartScopesCompilation** stored procedure is called to start search scope compilation as soon as possible.

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

```
PROCEDURE proc_MSS_StartScopesCompilation();
```

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<61> return any result sets. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.119 proc_MSS_UpdateBestBet

The **proc_MSS_UpdateBestBet** stored procedure is called to change the information about the specified best bet.

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

```
PROCEDURE proc_MSS_UpdateBestBet(
    @ConsumerGpId      nvarchar(50),
    @BestBetId         int,
    @Title             nvarchar(100),
    @Url               nvarchar(2048),
```

```
    @Description        nvarchar(500)
);
```

@ConsumerGpId: A string that uniquely identifies the keyword consumer group for the best bet.

@BestBetId: The unique identifier of the newly added Best Bet.

@Title: A string that contains the title for the best bet.

@Url: A string that contains the URL for the best bet.

@Description: A string that contains the description for the best bet.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<62> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.120 proc_MSS_UpdateBestBetOrder

The **proc_MSS_UpdateBestBetOrder** stored procedure is called to update the order of the specified best bet within all best bets associated with the specified keyword.

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

```
PROCEDURE proc_MSS_UpdateBestBetOrder(
    @SpecialTermId          int,
    @BestBetId              int,
    @Order                  int
);
```

@SpecialTermId: The unique identifier of the keyword.

@BestBetId: The unique identifier of the best bet.

@Order: The new order of the specified best bet within all best bets associated with the specified keyword.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<63> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.4.121 proc_MSS_UpdateLocationConfiguration

The **proc_MSS_UpdateLocationConfiguration** stored procedure is called to change the attributes of the federated location.

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

```
PROCEDURE proc_MSS_UpdateLocationConfiguration(
    @InternalName                nvarchar(60),
    @DisplayName                 nvarchar(60),
    @AdminDescription            nvarchar(256),
    @LocationType                tinyint,
```

```
    @Author                         nvarchar(60),
    @Version                        nvarchar(50),
    @IsDeletable                    bit,
    @IsPrefixPattern                bit,
    @QueryReformatPattern           ntext,
    @Propertieschema                ntext,
    @QueryRestriction               nvarchar(512),
    @KindsOfResults                 ntext,
    @Languages                      ntext,
    @IsRestricted                   bit,
    @AllowedSiteCollectionGuids     ntext,
    @Type                           tinyint,
    @Data                           ntext,
    @ConnectionUrlTemplate          nvarchar(2048),
    @MoreUrlTemplate                nvarchar(2048),
    @CreationDate                   datetime OUTPUT,
    @LastModifiedDate               datetime OUTPUT,
    @LocationId                     int
);
```

@InternalName: The unique internal name of the federated location that will be updated.

@DisplayName: The display name of the federated location.

@AdminDescription: The description of the federated location for the site collection administrator.

@LocationType: An 8 bit integer that specifies the protocol that MUST be used to connect to this federated location. The value MUST be a Location Type Data Type as specified in Section 2.2.2.16.

@Author: The author of the federated location.

@Version: An optional version number for this federated location. If a value is specified, it MUST contain at least one period (".").

@IsDeletable: MUST be 0 when the federated location cannot be deleted. Otherwise, it MUST be 1.

@IsPrefixPattern: MUST be 1 when the *@QueryReformatPattern* parameter is used as a prefix match. Otherwise, it MUST be 0, indicating that the *@QueryReformatPattern* parameter is used as a regular expression.

@QueryReformatPattern: The pattern for triggering the federated location during a search.

@Propertieschema: This parameter MUST be ignored by the server.

@QueryRestriction: The supplemental query restrictions to be appended to every search query.

@KindsOfResults: This parameter MUST be ignored by the server.

@Languages: The languages supported by the federated location.

@IsRestricted: MUST be 1 when this federated location is restricted to only the allowed sites specified by the *@AllowedSiteCollectionGuids* parameter. Otherwise, it MUST be 0.

@AllowedSiteCollectionGuids: A semi-colon delimited string of sites which are allowed to configure this federated location.

@Type: An 8 bit integer that specifies the type of authentication supported for federation. The value MUST be an Authentication Type data type, as specified in Section 2.2.2.1.

@Data: The authentication credential data.

@ConnectionUrlTemplate: The template for passing search queries to this federated location.

@MoreUrlTemplate: The template that specifies the URL of the HTML page that displays results for a search query.

@CreationDate: Upon return from this stored procedure, this parameter MUST be set to the UTC date and time when the federated location was created.

@LastModifiedDate: Upon return from this stored procedure, this parameter MUST be set to the UTC date and time when the federated location was last modified.

@LocationId: The identifier of the federated location.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.122 proc_MSS_UpdateProxy

The **proc_MSS_UpdateProxy** stored procedure is called to change whether or not to use the proxy settings for federated locations. The proxy settings MUST be used by all the federated locations.

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

```
PROCEDURE proc_MSS_UpdateProxy (
   @UseCrawlProxy             bit
);
```

@UseCrawlProxy: A 1-bit number that indicates whether all the federated locations MUST use the proxy settings or not. MUST be 1 **if all** federated locations MUST use the Proxy settings**. Otherwise, it MUST be 0.**

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.4.123 proc_MSS_UpdateSpecialTerm

The **proc_MSS_UpdateSpecialTerm** stored procedure is called to update the information about the specified keyword.

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

```
PROCEDURE proc_MSS_UpdateSpecialTerm(
    @ConsumerGpId       nvarchar(50),
    @SpecialTermId      int,
    @Term               nvarchar(100),
    @Definition         nvarchar(500) = null,
    @Contact            nvarchar(50) = null,
    @StartDate          datetime,
    @EndDate            datetime = null,
```

```
    @ReviewDate         datetime = null
);
```

@ConsumerGpId: A string that uniquely identifies the keyword consumer group**.**

@SpecialTermId: The unique identifier of the newly added keyword.

@Term: The term for the keyword.

@Definition: The description of the keyword.

@Contact: The contact name for the keyword.

@StartDate: A datetime value that specifies when the keyword begins to appear in search result.

@EndDate: A datetime value that specifies when the keyword stops appearing in search result.

@ReviewDate: A datetime value that specifies when the keyword is expected to be reviewed.

Return Code Values: An integer which MUST be 0.

Result Sets: SHOULD NOT<64> 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.

4 Protocol Examples

This section provides specific example scenarios for end-to-end administration tasks.

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.

4.1 Crawled Properties Administration

This section illustrates the protocol operations required to add, change and retrieve **crawled property categories** and their associated crawled properties, from the database on the back-end database server**.**

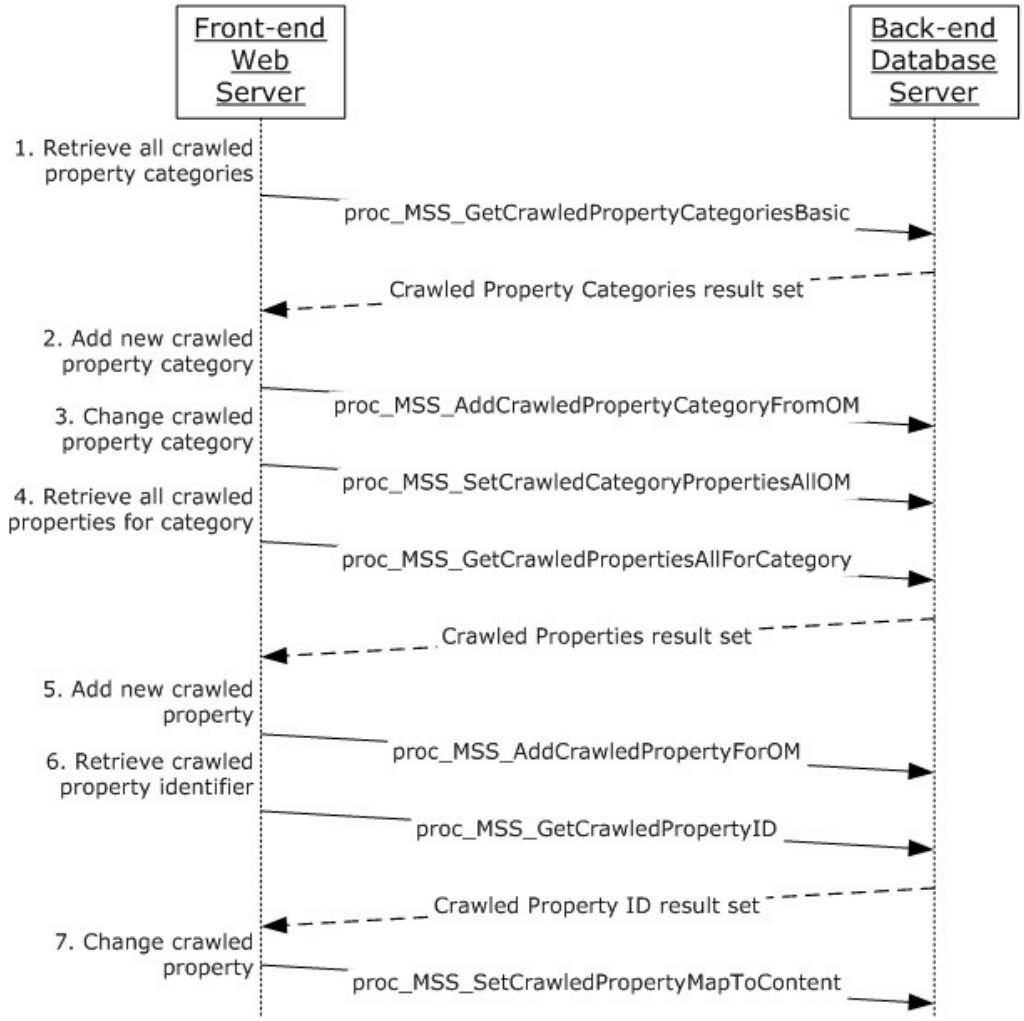

Figure 19: Crawled Properties Administration

The steps in the preceding diagram are explained below.

1. This example begins with the protocol client retrieving the list of all crawled property categories from the metadata schema to determine if a crawled property category already exists.

2. If the crawled property category does not exist in the result set returned from the call to **proc_MSS_GetCrawledPropertyCategoriesBasic** stored procedure, the protocol client calls **proc_MSS_AddCrawledPropertyCategoryFromOM** stored procedure to add the new crawled property category to the metadata schema.

3. In order to change data for the existing **crawled property category, the protocol client calls proc_MSS_SetCrawledCategoryPropertiesAllOM** stored procedure.

4. To add a new crawled property to the metadata schema, the protocol client first retrieves a list of crawled properties associated with the specified crawled property category to determine if the crawled property already exists.

5. If the crawled property is not returned as part of Crawled Properties result set, the client calls **proc_MSS_AddCrawledPropertyForOM** stored procedure to add it to the metadata schema and associate it with the crawled property category.

6. To change *IsMappedToContent* attribute for the existing crawled property, the protocol client first calls **proc_MSS_GetCrawledPropertyID** stored procedure to retrieve the unique identifier for the specified crawled property.

7. The client then calls **proc_MSS_SetCrawledPropertyMapToContent** stored procedure passing crawled property and the new value for *IsMappedToContent* attribute.

4.2 Managed Properties Administration

This section illustrates the protocol operations required to add, change and retrieve **managed properties and** associate them with the crawled properties, in the database on the back-end database server**.**

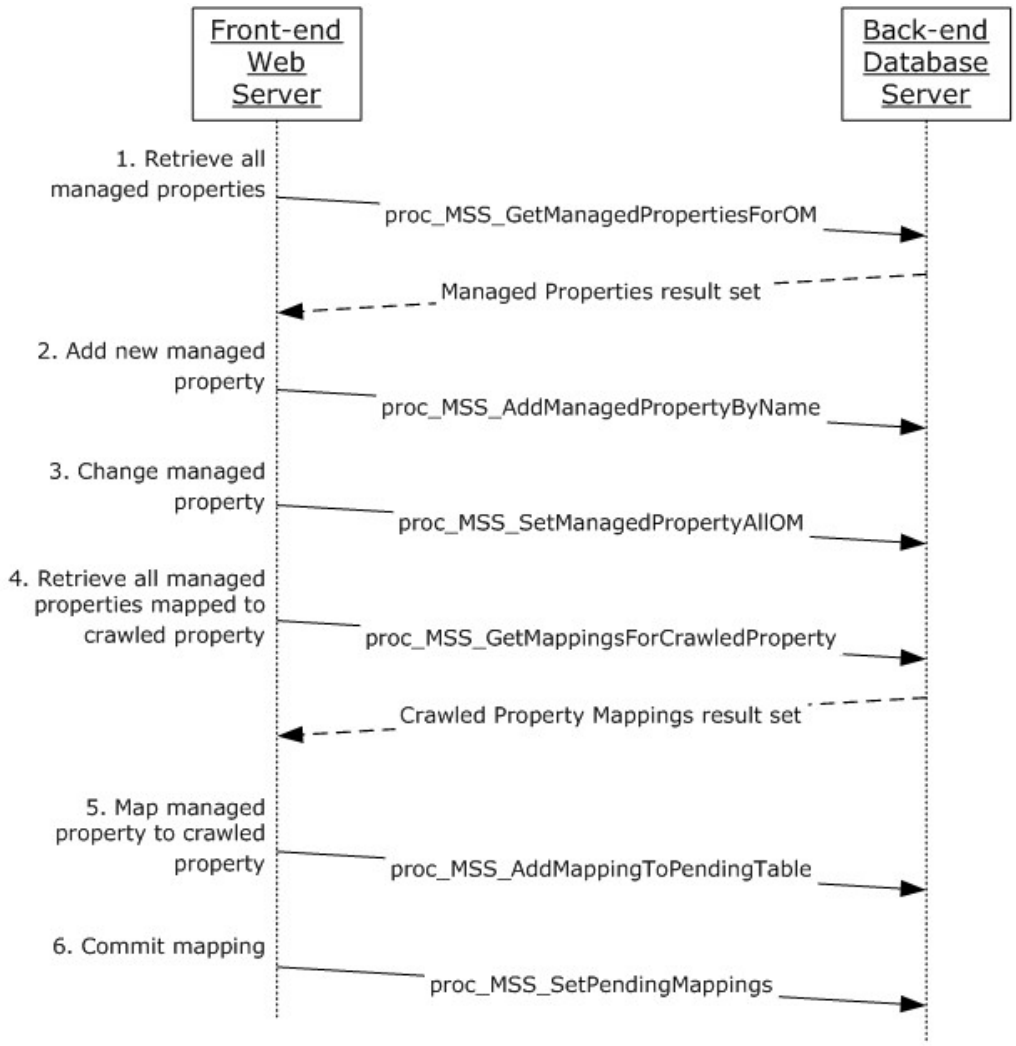

Figure 20: Managed Properties Administration

The steps in the preceding diagram are explained below.

1. This example begins with the protocol client retrieving the list of all managed properties from the metadata schema to determine if a managed property already exists.

2. If the managed property does not exist in the result set returned from the call to **proc_MSS_GetManagedPropertiesForOM** stored procedure, the protocol client calls **proc_MSS_AddManagedPropertyByName** stored procedure to add the new managed property to the metadata schema.

3. In order to change data for the existing **managed property, the protocol client calls proc_MSS_SetManagedPropertyAllOM** stored procedure.

4. To map the managed property to the crawled property, the protocol client first retrieves a list of all managed properties associated with the specified crawled property to determine if the managed property is already mapped.

5. If the managed property is not returned as part of Crawled Property Mappings result set, the client calls **proc_MSS_AddMappingToPendingTable** stored procedure to set the mapping between the managed property and the crawled property.

6. The mapping between the managed property and the crawled property is commited with the call to **proc_MSS_SetPendingMappings** stored procedure.

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.

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 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<2> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<3> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<4> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<5> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<6> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<7> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<8> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<9> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<10> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<11> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<12> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<13> Section 3.1.4: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<14> 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.

<15> 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.

<16> Section 3.1.4.3: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<17> Section 3.1.4.4: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<18> Section 3.1.4.5: 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 3.1.4.6: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<20> Section 3.1.4.7: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<21> Section 3.1.4.8: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<22> Section 3.1.4.9: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<23> Section 3.1.4.10: 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.

<24> Section 3.1.4.12: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<25> Section 3.1.4.13: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<26> Section 3.1.4.15: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<27> Section 3.1.4.16: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<28> Section 3.1.4.17: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<29> Section 3.1.4.18: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<30> Section 3.1.4.19: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<31> Section 3.1.4.20: 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.

<32> Section 3.1.4.21: 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.

<33> Section 3.1.4.22: 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.

<34> Section 3.1.4.23: 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.

<35> Section 3.1.4.24: 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.

<36> Section 3.1.4.26: 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.

<37> Section 3.1.4.27: 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.

<38> Section 3.1.4.28: 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.

<39> Section 3.1.4.29: 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.

<40> 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.

<41> 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.

<42> 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.

<43> 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.

<44> Section 3.1.4.34: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<45> 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.

<46> 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.

<47> Section 3.1.4.72.1: This functionality was introduced as part of the Office SharePoint Server 2007 Infrastructure Update.

<48> Section 3.1.4.104: 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.

<49> Section 3.1.4.106: 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.

<50> Section 3.1.4.107: 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.

<51> Section 3.1.4.108: 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.

<52> Section 3.1.4.109: 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.

<53> Section 3.1.4.110: 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.

<54> Section 3.1.4.111: 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.

<55> Section 3.1.4.112: 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.

<56> Section 3.1.4.113: 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.

<57> Section 3.1.4.114: 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.

<58> Section 3.1.4.115: 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.

<59> Section 3.1.4.116: 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.

<60> Section 3.1.4.117: 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.

<61> Section 3.1.4.118: 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.

<62> Section 3.1.4.119: 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.

<63> Section 3.1.4.120: 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.

<64> Section 3.1.4.123: 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

N

O

P

R

S

T

U

V

X