[MS-SQLPGAT2]:
SQL Gatherer
Version 2 Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
07/13/2009	0.1	Major	Initial Availability
08/28/2009	0.2	Editorial	Revised and edited the technical content
11/06/2009	0.3	Editorial	Revised and edited the technical content
02/19/2010	1.0	Major	Updated and revised the technical content
03/31/2010	1.01	Editorial	Revised and edited the technical content
04/30/2010	1.02	Editorial	Revised and edited the technical content
06/07/2010	1.03	Editorial	Revised and edited the technical content
06/29/2010	1.04	Minor	Clarified the meaning of the technical content.
07/23/2010	1.05	Editorial	Changed language and formatting in the technical content.
09/27/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.06	Editorial	Changed language and formatting in the technical content.
12/17/2010	1.06	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	1.6.1	Editorial	Changed language and formatting in the technical content.
06/10/2011	1.6.1	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	1.6.1	No change	No changes to the meaning, language, or formatting of the technical content.
04/11/2012	1.6.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	1.6.1	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction 9
1.1 Glossary 9
1.2 References 11
1.2.1 Normative References 11
1.2.2 Informative References 11
1.3 Protocol Overview (Synopsis) 12
1.3.1 Status Changes 13
1.3.2 Configuration Properties 13
1.3.3 Scope Compilation 14
1.4 Relationship to Other Protocols 14
1.5 Prerequisites/Preconditions 15
1.6 Applicability Statement 15
1.7 Versioning and Capability Negotiation 15
1.8 Vendor-Extensible Fields 15
1.9 Standards Assignments 15

2 Messages 16
2.1 Transport 16
2.2 Common Data Types 16
2.2.1 Simple Data Types and Enumerations 16
2.2.1.1 Project Identifier 16
2.2.1.2 Crawl Type 16
2.2.1.3 Crawl Request Type 16
2.2.1.4 Crawl Status 17
2.2.1.5 Crawl Starting Sub Status 17
2.2.1.6 Crawl Crawling Sub Status 17
2.2.1.7 Crawl Completing Sub Status 17
2.2.1.8 Crawl Stopping Sub Status 18
2.2.1.9 Crawl Error Logging Levels 18
2.2.1.10 Crawl Error Levels 18
2.2.1.11 Crawl Error Codes 18
2.2.1.12 Crawl Component State 19
2.2.1.13 Managed Type 19
2.2.1.14 Transaction Type 19
2.2.1.15 Transaction Scope 20
2.2.1.16 Search Catalog Status 20
2.2.1.17 Content Source Status 20
2.2.1.18 Update Queue Type 21
2.2.1.19 Project Name 21
2.2.1.20 Index Type 21
2.2.1.21 Item Type 21
2.2.1.22 Item Delete Reasons 22
2.2.1.23 Anchor Change Type 22
2.2.2 Bit Fields and Flag Structures 22
2.2.2.1 Pause Crawls Reasons 22
2.2.2.2 Crawl Stores with Links 23
2.2.2.3 Transaction Flags 23
2.2.2.4 End Path Flags 24
2.2.3 Binary Structures 24
2.2.3.1 Update Anchor Add Record 24

2.2.3.2 Update Anchor Add Structure 25
2.2.3.3 DocProps Add Record 25
2.2.3.4 DocProps Add Structure 26
2.2.3.5 DocProps Delete Record 27
2.2.3.6 DocProps Delete Structure 27
2.2.3.7 DocProps Delete With Hash Record 28
2.2.3.8 DocProps Delete With Hash Structure 28
2.2.3.9 DocResults Update Record 29
2.2.3.10 DocResults Update Structure 32
2.2.3.11 DocResults Anchor Update Record 33
2.2.3.12 DocResults Anchor Update Structure 33
2.2.3.13 DocSdIds Update Record 34
2.2.3.14 DocSdIds Update Structure 34
2.2.3.15 Alert History Update Record 35
2.2.3.16 Alert History Update Structure 36
2.2.3.17 Definitions Update Record 36
2.2.3.18 Definitions Update Structure 37
2.2.3.19 Transaction Commit Record 37
2.2.3.20 Transactions Commit Structure 44
2.2.3.21 Transaction Data Blob 44
2.2.3.22 Transaction string 45
2.2.4 Result Sets 45
2.2.4.1 Scope Compilation Result Set 45
2.2.5 Tables and Views 45
2.2.6 XML Structures 45
2.2.6.1 Namespaces 46
2.2.6.2 Simple Types 46
2.2.6.3 Complex Types 46
2.2.6.4 Elements 46
2.2.6.5 Attributes 46
2.2.6.6 Groups 46
2.2.6.7 Attribute Groups 46

3 Protocol Details 47
3.1 Server Details for Office SharePoint Server 47
3.1.1 Abstract Data Model 47
3.1.1.1 Crawl Status 47
3.1.1.2 Crawl Url History 47
3.1.1.3 Deleted URL 50
3.1.1.4 Crawl Queue 50
3.1.1.5 Links 51
3.1.1.6 Colleague Links 53
3.1.1.7 Crawled Hosts 53
3.1.1.8 Anchor Text Info 53
3.1.1.9 Pending Annotations 54
3.1.1.10 Pending Annotations Status 55
3.1.1.11 Social Distance Property 55
3.1.1.12 Social Distance Deleted Property 55
3.1.1.13 Social Distance New Property 56
3.1.1.14 Anchor Change 56
3.1.1.15 Pending Anchor Change 56
3.1.1.16 Reported Crawl Error 56
3.1.1.17 Current DocID Groups 56

3.1.1.18 Current DocID Chunk 57
3.1.1.19 Requested Delete Crawls 57
3.1.1.20 Changed Anchor Target Documents 57
3.1.1.21 Changed Anchor Source Documents 57
3.1.1.22 Changed Anchor Committed Documents 57
3.1.1.23 Changed Anchor Deleted Documents 58
3.1.1.24 UserID DocID Pair 58
3.1.1.25 Crawl Component Status 58
3.1.1.26 Local Crawl Components 58
3.1.1.27 Local Completed Crawls 59
3.1.1.28 Crawl Store Status 59
3.1.1.29 Scopes to Compile 59
3.1.1.30 Scope Rules to Compile 59
3.1.1.31 Deleted Scopes 60
3.1.1.32 Deleted Scopes to Compile 60
3.1.1.33 Compiled Scopes 60
3.1.1.34 Compiled Scope Rules 60
3.1.1.35 Component Activity 61
3.1.1.36 Configuration Properties 61
3.1.1.37 Annotations 62
3.1.1.38 Anchor Document Properties Blob 62
3.1.1.39 Search Component Name 62
3.1.1.40 Definitions 62
3.1.1.41 Commands 63
3.1.1.42 Metadata Store Crawl Components 63
3.1.1.43 Social Distance Unchanged Property 63
3.1.2 Timers 63
3.1.3 Initialization 63
3.1.4 Higher-Layer Triggered Events 63
3.1.5 Message Processing Events and Sequencing Rules 64
3.1.5.1 proc_MSS_AddAndReturnCrawledProperty 64
3.1.5.2 proc_MSS_AddCrawledPropertyCategoryWithDefaults 65
3.1.5.3 proc_MSS_CheckIfStatusChangeComplete 66
3.1.5.4 proc_MSS_IsSystemPausedForRefactoring 67
3.1.5.5 proc_MSS_CommitConfigTransaction 68
3.1.5.6 proc_MSS_CompleteStatusChange 68
3.1.5.7 proc_MSS_Crawl 69
3.1.5.8 proc_MSS_CrawlAdmin 87
3.1.5.9 proc_MSS_CrawlStoreToJoinAllOngoingCrawls 97
3.1.5.10 proc_MSS_CreateNewComponentRecord 98
3.1.5.11 proc_MSS_CreateConfigTransaction 98
3.1.5.12 proc_MSS_DeleteConfigurationProperties 99
3.1.5.13 proc_MSS_DeleteConfigurationProperty 99
3.1.5.14 proc_MSS_FlushTemp0 100
3.1.5.15 proc_MSS_GetAnchorDocIDs 113
3.1.5.15.1 GetAnchorDocIDs Result Set 113
3.1.5.16 proc_MSS_GetCompiledScopeRules 114
3.1.5.16.1 Compiled Scopes Result Set 114
3.1.5.17 proc_MSS_GetCompiledScopes 114
3.1.5.17.1 Scopes Result Set 114
3.1.5.18 proc_MSS_GetCompletedScopesCompilationId 114
3.1.5.19 proc_MSS_GetComponentStatusUpToDate 115
3.1.5.20 proc_MSS_GetConfigurationProperties 116

3.1.5.20.1 Configuration Properties Result Set 116
3.1.5.21 proc_MSS_GetConfigurationProperty 116
3.1.5.22 proc_MSS_GetContentSourceDocCount 117
3.1.5.23 proc_MSS_GetCrawledProperties 117
3.1.5.23.1 GetCrawledProperties Result Set 118
3.1.5.24 proc_MSS_GetCrawledPropMappingUpdates 119
3.1.5.24.1 GetCrawledPropertyMappingUpdates Result Set 119
3.1.5.25 proc_MSS_GetCrawls 120
3.1.5.25.1 Get Crawls Result Set 120
3.1.5.26 proc_MSS_GetCurrentRegistryVersion 121
3.1.5.27 proc_MSS_GetDatabaseID 122
3.1.5.28 proc_MSS_GetAllGathererDatabaseIDs 122
3.1.5.29 Gatherer DB IDs Result Set 122
3.1.5.30 proc_MSS_GetAllPropertyStoreIDs 122
3.1.5.31 Property Store IDs Result Set 123
3.1.5.32 proc_MSS_GetDeleteCrawl 123
3.1.5.32.1 Delete Crawl Result Set 123
3.1.5.33 proc_MSS_GetDeletedScopesForCompilation 124
3.1.5.33.1 Deleted Search Scopes Result Set 124
3.1.5.34 proc_MSS_GetDocCount 124
3.1.5.35 proc_MSS_GetDocStatus 125
3.1.5.35.1 Document Status Result Set 125
3.1.5.36 proc_MSS_GetError 125
3.1.5.37 proc_MSS_GetGathererDBs 126
3.1.5.37.1 Gatherer DBs Result Set 126
3.1.5.38 proc_MSS_GetHost 127
3.1.5.39 proc_MSS_GetIncompleteCommands 127
3.1.5.39.1 Commands Result Set 128
3.1.5.40 proc_MSS_GetManagedProperties 128
3.1.5.40.1 GetManagedProperties Result Set 128
3.1.5.41 proc_MSS_GetMappingsForCrawledPropertiesById 131
3.1.5.42 Get Mappings for Crawled Properties by Identifier Result Set 132
3.1.5.43 proc_MSS_GetMasterRole 132
3.1.5.44 proc_MSS_GetMatrixPIAliases 132
3.1.5.44.1 MatrixPI Aliases Result Set 133
3.1.5.45 proc_MSS_GetNextCrawlBatch 133
3.1.5.45.1 GetNextCrawlBatch Result Set 134
3.1.5.46 proc_MSS_InvalidateAdminDocIDChunks 136
3.1.5.47 proc_MSS_GetNextDocIDChunk 137
3.1.5.48 proc_MSS_GetPauseCrawlsReasons 138
3.1.5.49 proc_MSS_GetPauseStatus 138
3.1.5.50 proc_MSS_GetPreferredNameList 138
3.1.5.50.1 Preferred Name List Result Set 139
3.1.5.51 proc_MSS_GetRankingModels 139
3.1.5.52 proc_MSS_GetSampleExtremes 139
3.1.5.52.1 Sample Extremes Result Set 139
3.1.5.53 proc_MSS_GetSchemaHighLevelInfo 140
3.1.5.53.1 Schema High Level Info Result Set 140
3.1.5.54 proc_MSS_GetSchemaMappings 141
3.1.5.54.1 GetSchemaMappings Result Set 141
3.1.5.55 proc_MSS_GetSchemaParameters 141
3.1.5.55.1 GetSchemaParameters Result Set 142
3.1.5.56 proc_MSS_GetScopeRulesForCompilation 142

3.1.5.57 proc_MSS_GetScopesForCompilation 142
3.1.5.58 proc_MSS_GetSDID 143
3.1.5.58.1 GetSDID Result Set 143
3.1.5.59 proc_MSS_GetStaticRankingFeatures 143
3.1.5.60 proc_MSS_GetStatus 143
3.1.5.61 proc_MSS_GetStatusChangeRequest 144
3.1.5.62 proc_MSS_GetTopAnchorLinks 146
3.1.5.62.1 Top Anchor Links Result Set 146
3.1.5.63 proc_MSS_IsPauseCrawlsPortalContentDone 147
3.1.5.64 proc_MSS_PrepareForAnnotationsUpdate 148
3.1.5.65 proc_MSS_CommitTransactions 148
3.1.5.66 proc_MSS_ProcessCommitted 149
3.1.5.67 proc_MSS_PushSD 162
3.1.5.68 proc_MSS_QLog_GetRelevanceUpdateData 163
3.1.5.68.1 proc_MSS_QLog_GetRelevanceUpdateData Result Set 1 163
3.1.5.68.2 proc_MSS_QLog_GetRelevanceUpdateData Result Set 2 164
3.1.5.69 proc_MSS_Recompile 164
3.1.5.70 proc_MSS_DefragGathererIndexes 165
3.1.5.71 proc_MSS_RemoveConfigTransactions 165
3.1.5.72 proc_MSS_RemoveCrawlStoreFromAllCrawls 165
3.1.5.73 proc_MSS_ReportCommandCompleted 166
3.1.5.74 proc_MSS_ResetCatalog 166
3.1.5.75 proc_MSS_SetConfigPropertyWithTransaction 166
3.1.5.76 proc_MSS_SetConfigurationProperty 167
3.1.5.77 proc_MSSSetCrawlComponentStatus 167
3.1.5.78 proc_MSS_SetCrawledPropertyIsSampleCacheFull 168
3.1.5.79 proc_MSS_SetRequestToPauseCrawlsPortalContent 168
3.1.5.80 proc_MSS_SetStatusChangeRequest 169
3.1.5.81 proc_MSS_ShareScopesCompilationInfo 169
3.1.5.82 proc_MSS_TruncateCleanupTable 171
3.1.5.83 proc_MSS_UnloadTransactions 171
3.1.5.84 proc_MSS_SetAnnotationsPendingStatus 172
3.1.5.85 proc_MSS_TruncateAnnotationsTables 173
3.1.5.86 proc_MSS_UpdateAnchorDocPropsBlob 173
3.1.5.87 proc_MSS_UpdateCrawlStatistics 173
3.1.5.88 proc_MSS_UpdateDocCount 175
3.1.5.89 proc_MSS_UpdateLinksBitmapField 175
3.1.5.90 proc_MSS_UpdatePropertyStore 175
3.1.5.91 proc_MSS_CleanupWithInterval 178
3.1.6 Timer Events 178
3.1.7 Other Local Events 178
3.2 Office SharePoint Server Client Details 179
3.2.1 Abstract Data Model 179
3.2.2 Timers 179
3.2.3 Initialization 179
3.2.4 Higher-Layer Triggered Events 179
3.2.5 Message Processing Events and Sequencing Rules 179
3.2.5.1 Full Crawl Sequence 179
3.2.5.2 Incremental Crawl Sequence 182
3.2.5.3 Delete Crawl Sequence 185
3.2.5.4 Anchor Text Crawl 189
3.2.5.5 Status Changes 191
3.2.5.6 Configuration Properties 192

3.2.5.7 Scope Compilation 193
3.2.5.8 Request Master Merge 193
3.2.6 Timer Events 194
3.2.7 Other Local Events 194
3.3 Server Details for Windows SharePoint Server 194
3.3.1 Abstract Data Model 194
3.3.2 Timers 195
3.3.3 Initialization 195
3.3.4 Higher-Layer Triggered Events 195
3.3.5 Message Processing Events and Sequencing Rules 196
3.3.6 Timer Events 198
3.3.7 Other Local Events 198
3.4 Windows SharePoint Server Client Details 198
3.4.1 Abstract Data Model 198
3.4.2 Timers 198
3.4.3 Initialization 198
3.4.4 Higher-Layer Triggered Events 198
3.4.5 Message Processing Events and Sequencing Rules 198
3.4.5.1 Full Crawl Sequence 198
3.4.5.2 Incremental Crawl Sequence 199
3.4.5.3 Delete Crawl Sequence 199
3.4.5.4 Configuration Properties 199
3.4.5.5 Scope Compilation 199
3.4.6 Timer Events 199
3.4.7 Other Local Events 199

4 Protocol Examples 200
4.1 Add a New Content Source 200
4.2 Add a New Host Distribution Rule 201

5 Security 203
5.1 Security Considerations for Implementers 203
5.2 Index of Security Parameters 203

6 Appendix A: Product Behavior 204

7 Change Tracking 207

8 Index 208

1 Introduction

This document specifies the SQL Gatherer Protocol, which allows for communication between the protocol client (Web and application servers) and protocol server (database server) to perform data query and update commands on the protocol server in relation to search crawl (content indexing) functions.

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

1.1 Glossary

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

access control list (ACL)
big-endian
Coordinated Universal Time (UTC)
GUID
GUIDString
HRESULT
language code identifier (LCID)
Unicode

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

administration component
anchor crawl
anchor text
application server
back-end database server
binary large object (BLOB)
Business Data Connectivity (BDC)
clickthrough
compact URL
component registry version
configuration property
content source
crawl
crawl component
crawl queue
crawl status
crawl store
crawl URL history
crawled property
crawled property category
crawled property set identifier
datetime
delete crawl
display URL
document identifier
folder
full crawl
full-text index catalog

host distribution rule
host name
incremental crawl
index server
item
link
managed property
mapping order
master crawl component
master merge
metadata index
metadata schema
metadata store
object model
portal content
property identifier
property oriented rank
query component
query independent rank
result set
return code
search application
search catalog
search component
search scope
search scope compilation
search scope compilation identifier
search scope consumer
search scope index
search scope rule
search security descriptor
search service application
site hop
start address
stored procedure
transaction
Transact-Structured Query Language (T-SQL)
Uniform Resource Identifier (URI)
Uniform Resource Locator (URL)
user profile record identifier
variant type

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.

catalog reset: The process of removing information about all crawled items from a full-text index catalog.

compiled search scope: A search scope that is the result of the search scope compilation process.

search property mapping: A mapping that defines the relationship between a crawled property and a managed property. See also mapping order.

search service instance: An object that represents the specifications of the search service on a specific server.

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

1.2 References

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

1.2.1 Normative References

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

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

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

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

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

[MS-SQLPADM] Microsoft Corporation, "SQL Administration Protocol Specification".

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

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

[MS-SRCHTP] Microsoft Corporation, "Search Topology 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

[RFC5234] Crocker, D., Ed., and Overell, P., "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008, http://www.rfc-editor.org/rfc/rfc5234.txt

1.2.2 Informative References

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

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

1.3 Protocol Overview (Synopsis)

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

- Microsoft® SharePoint® 2010 Products and Technologies role
- Microsoft® SharePoint® Foundation 2010 role

This protocol is used for performing full, incremental, delete and **anchor text** crawls.

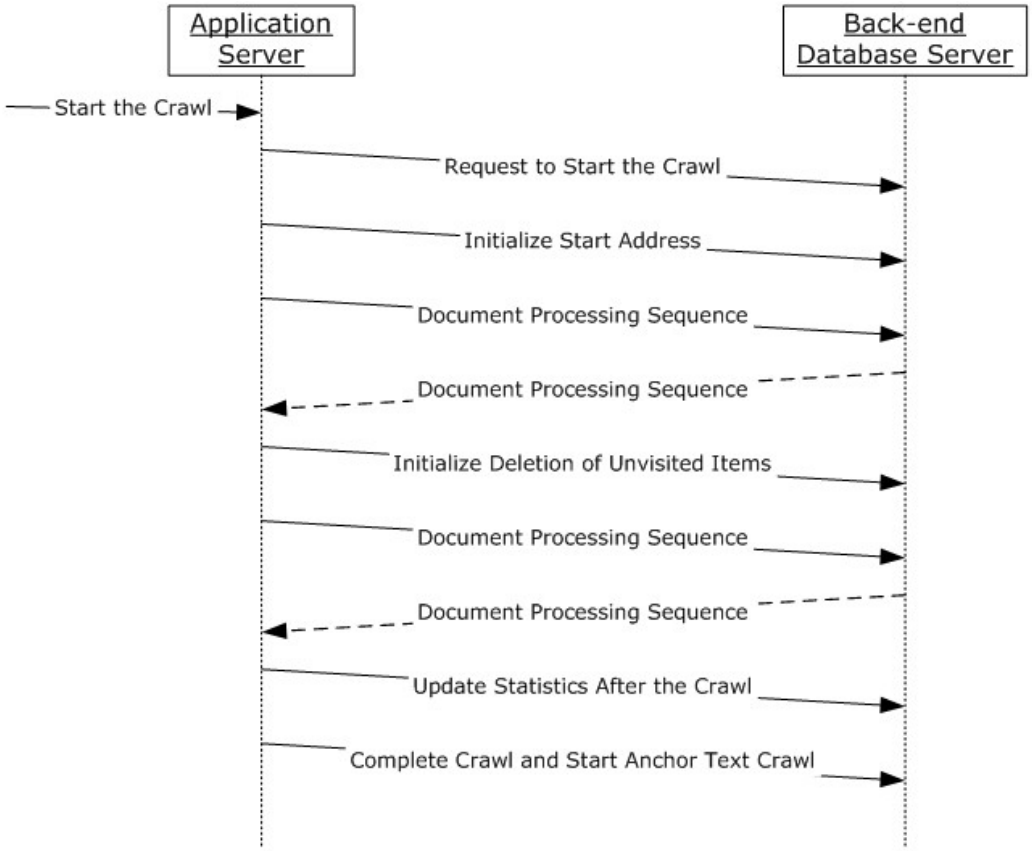


Figure 1: Crawl sequence diagram

The sequence of the crawl is as follows:

- The protocol client creates a new crawl.
- The protocol client makes a call to initialize **start addresses**.
- The protocol client attempts to start the crawl, ending it if there is another crawl of the same **content source** already in progress.

- The protocol client performs repeatedly the "Document Processing Sequence", in which the protocol client:
  - Calls the database server to retrieve items to crawl from the Crawl Queue.
  - Adds new crawl links to the Link Set.
  - Calls the database server to move crawl links from the Link Set to the Crawl Queue.
  - Calls the database server to update statistics and report progress
- The protocol client moves the crawl into the next stage to delete documents which were not touched during the last crawl, new items are added into Crawl Queue (which should be removed) and "Document Processing Sequence" performed again.
- The protocol client calls database server to update statistics and mark crawl to a status of Done. The client then calls database server to start anchor text crawl.

In addition to the crawl functionality, the following subsections give an overview of three other types of functionality supported by this protocol:

- Status changes
- Management of configuration properties
- Search scope compilation.

1.3.1 Status Changes

Status changes reflect when crawl components move from one state to the next.

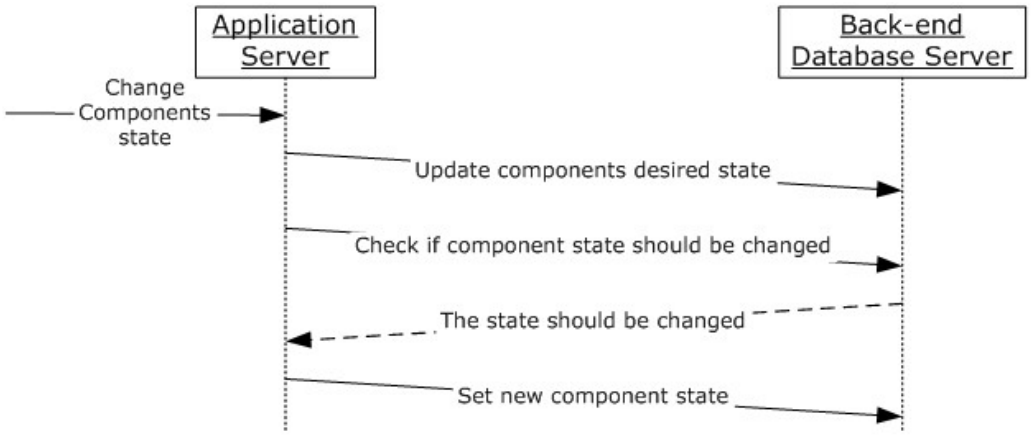


Figure 2: Status change diagram

The client requests to change status of the component. The client makes a call to update desired state. The client makes a call to check if component state should be changed. The client makes a call to set new state.

1.3.2 Configuration Properties

Configuration properties are used for storing system attributes. They can be added/modified.

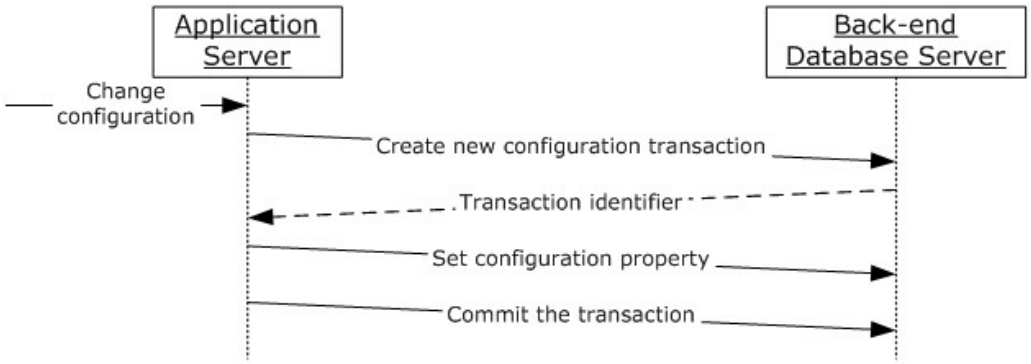


Figure 3: Configuration properties diagram

The client receives the request to change set of configuration properties. The client makes a call to create new configuration property transaction. The client makes a set of calls to set new values for all properties it needs to update. The client makes a call to commit the transaction.

1.3.3 Scope Compilation

Search scopes are subject to **search scope compilation**. The application server makes a call to the back-end database server every 30 seconds to start or update search scope compilation if necessary. When a search scope compilation is started, the application server makes a call to the back-end database server to retrieve:

- search scopes to be compiled
- deleted search scopes to be compiled
- **search scope rules** to be compiled in the current search scope compilation

1.4 Relationship to Other Protocols

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

This relationship is shown in the following diagram:

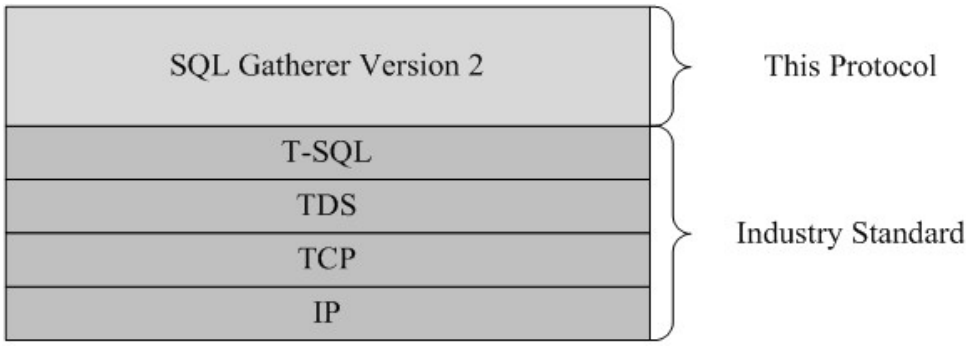


Figure 4: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

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

1.6 Applicability Statement

This protocol is applicable for the following configurations:

There can be up to 4 Crawl Stores; up to 32 crawl components. The system can process up to 100 million documents.

1.7 Versioning and Capability Negotiation

Supported Transports

This protocol uses the TDS protocol, as described in [MS-TDS], for SQL stored procedure calls, as specified in section 2.1.

Version Negotiation

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

Security and Authentication Methods

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

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

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

2.2 Common Data Types

This section contains common definitions used by this protocol.

2.2.1 Simple Data Types and Enumerations

2.2.1.1 Project Identifier

A unique identifier for a search project. MUST be one of the values in the following table:

Value	Description
1	Portal content.
2 This value is allowed only when the server is running Microsoft® SharePoint® 2010 Products and Technologies role.	Anchor Project.

2.2.1.2 Crawl Type

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

Value	Description
1	**Full crawl**.
2	**Incremental crawl**.
3	**Delete crawl**

2.2.1.3 Crawl Request Type

The request for crawl action. The value MUST be in the following table:

Value	Description
0	No Request
1	Request to Start
2	Request to Stop
3	Request to Pause
4	Request to Resume

2.2.1.4 Crawl Status

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

Value	Description
1	**Initializing:** The crawl in state 1 is in starting phase.
4	**Started:** The crawl in state 2 is in crawling phase.
5	**Forbid:** The crawl cannot start either because another crawl of the same content is already in progress, or there are no valid start addresses in the content source that got the crawl start request.
7	Resuming.
8	Pausing.
9	Paused.
11	Done.
12	Stopped.
13	**Stopping:** The crawl in state 13 is in stopping phase.
14	**Completing:** The crawl in state 14 is referred to be in completing phase.

2.2.1.5 Crawl Starting Sub Status

Detailed state of the crawl while it is in starting phase.

Value	Description
1	Adding Start Address.
2	Wait For crawl components.

2.2.1.6 Crawl Crawling Sub Status

Detailed state of the crawl while it is in crawling phase.

Value	Description
1	Crawling.
2	Move Unvisited Items to Crawl Queue (items which were not processed during the last crawl).
3	Delete Unvisited (items which were not processed during the last crawl).
4	Wait All Crawl Stores.

2.2.1.7 Crawl Completing Sub Status

Detailed state of the crawl while it is in the completing phase. MUST be one of the values from the following table.

Value	Description
1	Wait For crawl components.
2	Completing.
4	Deletes Pending.

2.2.1.8 Crawl Stopping Sub Status

Detailed state of the crawl while it is in Stopping phase. MUST be one of the values in the following table.

Value	Description
1	Wait for crawl components to Stop.
2	Complete Stop.

2.2.1.9 Crawl Error Logging Levels

This enumeration specifies importance of the item as determined by components internal to the search service instance. Importance is measured by the number of items that can be crawled under a particular item. Assuming item fails to get crawled, all of its child objects also won't be crawled. The higher the number of items that will be missed, the more important the log level is. MUST be one of the values in the following table:

Value	Description
0	Least important item.
1	Item of medium importance.
2	The most important item.

2.2.1.10 Crawl Error Levels

Severity levels for crawl errors.

Value	Description
0	Success (no errors).
1	Warning.
2	Error.

2.2.1.11 Crawl Error Codes

Value	Description
0x41203	Path not modified.
0x80040D07	URL is excluded because of crawl rule.

Value	Description
0x80040D08	URL is excluded because of **site hops**.
0x40D90	the item was marked with no-index meta-tag.
0x810200BC	Change Log Cookie is too old.
0x80041205	Access denied.
0x80041201	Object not found.

2.2.1.12 Crawl Component State

State of a crawl component.

Value	Description
1	OK.
0	Not Initialized.
3	Disabled.
4	Remount.
5	Inactive.
6	Disabled For Remove.

2.2.1.13 Managed Type

Managed Type identifies the data type of the **managed property**. MUST be a value from the following table.

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

2.2.1.14 Transaction Type

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

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

2.2.1.15 Transaction Scope

The scope of the given **transaction (1)**. The value MUST be in the following table:

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

2.2.1.16 Search Catalog Status

Search Catalog Status is an aggregate state describing all the crawls associated with a **search catalog**. The value MUST be in the following table:

Value	Description
0	Idle.
1	All crawls in progress are full crawls.
6	There’s at least one incremental crawl in progress and no delete crawls.
10	There’s at least one delete crawl in progress.

2.2.1.17 Content Source Status

The state of the crawl associated with a content source. The value MUST be in the following table:

Value	Description
0	Idle.
1	A full crawl is in progress.
2	The crawl is paused.
6	An incremental crawl is in progress.
8	The crawl is stopping.
9	The crawl is pausing.

Value	Description
10	A delete crawl is in progress
11	The crawl is resuming.
12	The crawl is starting.
13	The crawl is completing.

2.2.1.18 Update Queue Type

Value	Description
1	Inserts: the data being persisted is for documents that the crawl component has identified as new.
2	Repairs: the data being persisted is for documents that already exist in the **metadata store** or for documents that do not exist in the metadata store or for documents that are only partially persisted in the property store because of a previous error.
3	Updates: the data being persisted is updates, adds or deletes for documents existing in the metadata store.
4	Anchor: the data being persisted is from an **anchor crawl**.

2.2.1.19 Project Name

The project name corresponding to a project identifier. Its value MUST be in the following table.

Value	Project Identifier	Description
Portal_Content	1	Portal content.
AnchorProject	2	Anchor Project.

2.2.1.20 Index Type

Index Type specifies whether the item can be returned in search results. Its value MUST be in the following table:

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

2.2.1.21 Item Type

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

Value	Name	Description
1	Start Address	The link is a start address.

Value	Name	Description
2	Link	The link has been discovered in portal content crawl; the crawl will follow this link.
4	Link Shallow	The link has been discovered in portal content crawl; the crawl will not follow this link.
6	Anchor Link	The link has been discovered in anchor text crawl.
7	Last Link	Last link.

2.2.1.22 Item Delete Reasons

Item delete reasons. Its value MUST be in the following table:

Value	Name	Description
1	Error	The item was deleted because an error was received while crawling it.
2	Delete Crawl	The item was deleted by a Delete Crawl.
3	Delete Unvisited Full	The item was deleted because a full crawl on the content source did not reach this item.
4	Delete Unvisited Incremental	The item was deleted because an incremental crawl on the content source did not reach this item.
5	Delete Unvisited Log	The item was deleted because an change log crawl did not reach this item.
6	Delete Recursive	The item was deleted because the parent item was deleted.
7	Delete Excluded	The item was deleted because it was excluded by a crawl rule.
8	Delete Change Log	The item was deleted because of delete event in the change log.

2.2.1.23 Anchor Change Type

Indicator of link status for the given in Anchor Change Set. The value MUST be in the following table

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

2.2.2 Bit Fields and Flag Structures

2.2.2.1 Pause Crawls Reasons

A 32 bit integer bit mask of flags. The Pause Crawls Reasons bit mask represents the reasons for a **search service application** to pause crawls. The only valid bits of the Pause Crawls Reasons bit mask are specified in the following table:

Value	Description
0x00000000	No Pause.
0x00000001	Pause for initialization of query server.
0x00000002	Pause for backup or restore of search service application.
0x00000004	Pause for Volume Shadow Service backup.
0x00000008	Pause because query server is unavailable.
0x00000010	Pause for index propagation.
0x00000020	Pause for database refactoring.
0x00000080	Pause because of an external request.
0x7FFFFFFF	Pause for all reasons.

2.2.2.2 Crawl Stores with Links

A 32 bit integer bit mask of flags. The Crawl Stores with Links bit mask represents the list of crawl stores which have links to current document. The only valid bits of the Crawl Stores with Links bit mask are specified in the following table:

Value	Description
0x00000000	None of the crawl stores has a link to this document.
0x00000001	The crawl store with Ordinal equal to "0" in Set, as specified in [MS-SRCHTP] section 3.3.1.3, has a link to this item.
0x00000002	The crawl store with Ordinal equal to "1" has a link to this item.
0x00000004	The crawl store with Ordinal equal to "2" has a link to this item.
0x00000008	The crawl store with Ordinal equal to "3" has a link to this item.
0x00000010	The crawl store with Ordinal equal to "4" has a link to this item.
0x00000020	The crawl store with Ordinal equal to "5" has a link to this item.
0x00000080	The crawl store with Ordinal equal to "6" has a link to this item.
0x00000100	The crawl store with Ordinal equal to "7" has a link to this item.

2.2.2.3 Transaction Flags

A 32 bit integer bit mask of flags. The only valid bits are specified in the following table:

Value	Description
0x00000004	The item is a **folder**. Also may be referred to as a directory or container.
0x00000008	If the item is a folder, the links in the item have to be processed by the current crawl.
0x00000040	The item has a valid last modified timestamp.

Value	Description
0x00000200	This item was discovered as a result of enumeration the parent folder which returns timestamps along with items.
0x02000000	The current crawl updates only the item access control list (ACL); the item content is not processed.

2.2.2.4 End Path Flags

A 32 bit integer bit mask of flags. The only valid bits are specified in the following table:

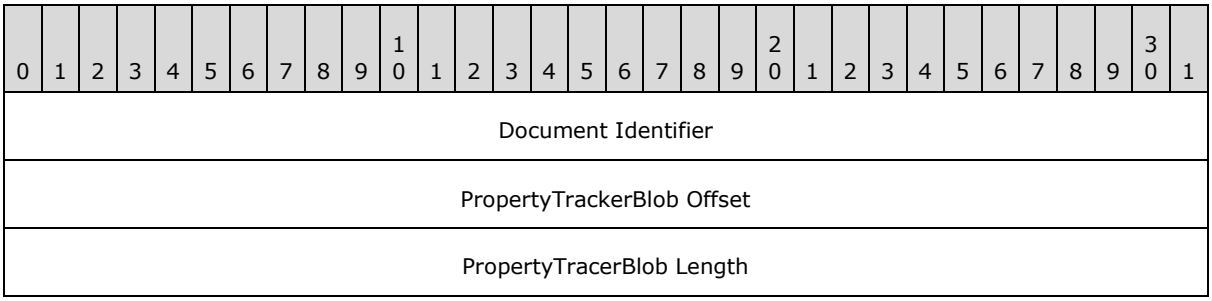
Value	Description
0x00000001	The access URL ends with a slash.
0x00000002	The display URL ends with a slash.

2.2.3 Binary Structures

The binary structures in the following sections are used by the protocol defined in this document.

2.2.3.1 Update Anchor Add Record

The **Update Anchor Add** record holds a document identifier as well as an offset and the length of the **propertyTrackerBlob**, as specified in section 3.1.1.38, for each document. The **propertyTrackerBlob** offset is the offset into a stream of **propertyTrackerBlobs** and the **propertyTrackerBlob** length is the length of the **propertyTrackerBlob**. When the **propertyTrackerBlob** length is greater than zero, the document information MUST be either inserted or updated to the **Anchor Document Properties** set. When the **propertyTrackerBlob** length is 0, the document MUST be deleted from the **Anchor Document Properties** set. When the **propertyTrackerBlob** length is 0, the **propertyTrackerBlob** offset MUST be ignored.



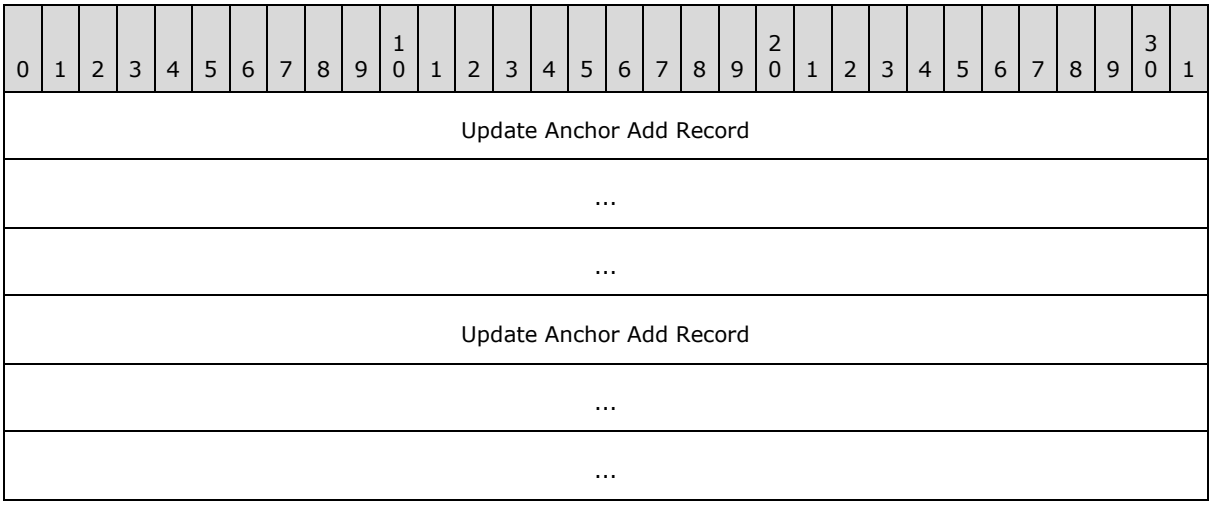
Document Identifier (4 bytes): A 4 byte integer in **big-endian** format that identifies a document.

PropertyTrackerBlob Offset (4 bytes): When **propertyTrackerBlob** length is greater than zero, MUST be a 4 byte integer in big-endian format which is the offset into a stream of **propertyTrackerBlobs** of the **propertyTrackerBlob** for the document identified in **Document Identifier** in this table. When the **propertyTrackerBlob** length is zero, MUST be ignored.

PropertyTrackerBlob Length (4 bytes): MUST be a 4 byte integer in big-endian format that is the length of the **propertyTrackerBlob** for the document identified in **Document Identifier** in the preceding diagram.

2.2.3.2 Update Anchor Add Structure

This structure is an array of Update Anchor Add Records, as specified in section 2.2.3.1.

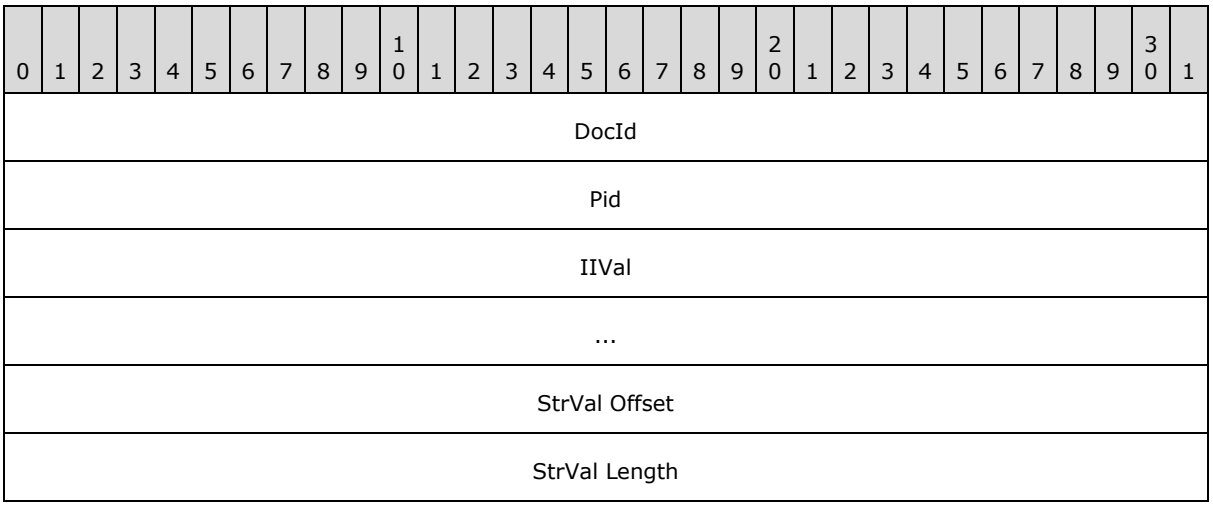


Update Anchor Add Record (12 bytes): First Update Anchor Add Record.

Update Anchor Add Record (12 bytes): Second Update Anchor Add Record.

2.2.3.3 DocProps Add Record

Contains information needed to update or add a row for a document in **MSSDocProps** specified in [MS-SQLPQ2] section 2.2.5.2. When the property type is a string, strVal Offset and length, as well as the **StrVal2** offset and length, are used to parse a stream of Unicode characters that are passed as a separate parameter.



StrVal2 Offset
StrVal2 Length

DocId (4 bytes): A 4-byte integer in big-endian format that is put in the **docid** column of the **MSSDocProps** table which identifies a document.

Pid (4 bytes): A 4-byte integer in big-endian format that is put in the **pid** column of the **MSSDocProps** table which identifies a managed property of the document identified in **DocId**.

llVal (8 bytes): An 8-byte integer in big-endian format that is put in the **llVal** column of the **MSSDocProps** table.

StrVal Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is put into the **StrVal** column of the **MSSDocProps** table starts. MUST be ignored if the **StrVal** Length is 0.

StrVal Length (4 bytes): MUST be a 4-byte integer in big-endian format that is the length of string that is put into the **StrVal** column of the **MSSDocProps** table. When the length is 0, a null value MUST be assigned to the string.

StrVal2 Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is put into the **StrVal2** column of the **MSSDocProps** table starts. MUST be ignored if the **StrVal2** Length is 0.

StrVal2 Length (4 bytes): MUST be a 4-byte integer in big-endian format that is the length of string that is put into the StrVal2 column of the MSSDocProps table. When the length is 0, a null value MUST be assigned to the string.

2.2.3.4 DocProps Add Structure

An array of DocProps Add Records, as specified in section 2.2.3.3.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
DocProps Add Record (32 bytes)																															
...																															
...																															
...																															
...																															
...																															
...																															

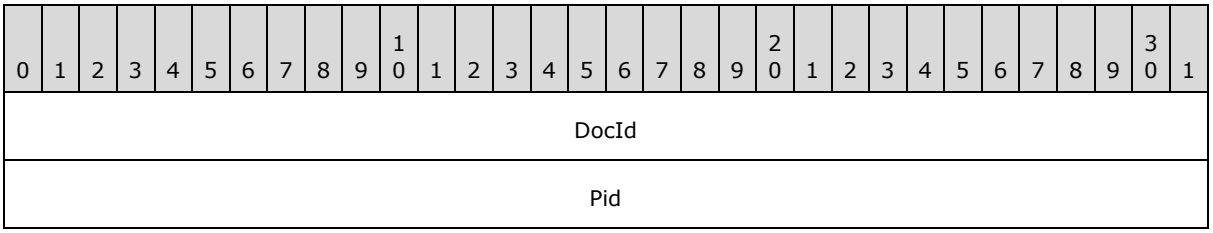
...
DocProps Add Record (32 bytes)
...
...
...
...
...
...
...
...

DocProps Add Record (32 bytes): First **DocProps Add Record**.

DocProps Add Record (32 bytes): Second **DocProps Add Record**.

2.2.3.5 DocProps Delete Record

Contains information needed to delete the rows for a property of a document in MSSDocProps specified in [MS-SQLPQ2] section 2.2.5.2.

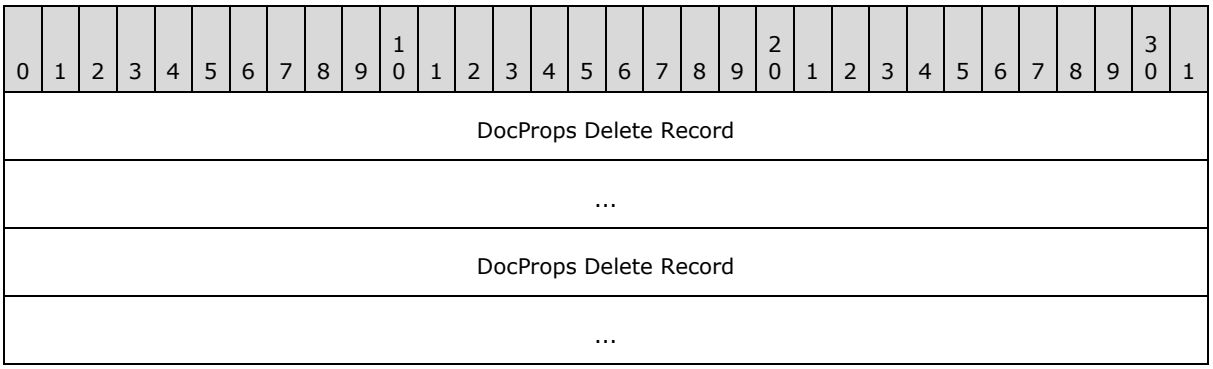


DocId (4 bytes): A 4-byte integer in big-endian format that is compared to the **docid** column of the **MSSDocProps** table which identifies a document.

Pid (4 bytes): A 4-byte integer in big-endian format that is compared to the **pid** column of the **MSSDocProps** table which identifies a managed property of the document identified in **DocId**.

2.2.3.6 DocProps Delete Structure

An array of DocProps Delete Records, as specified in section 2.2.3.5.

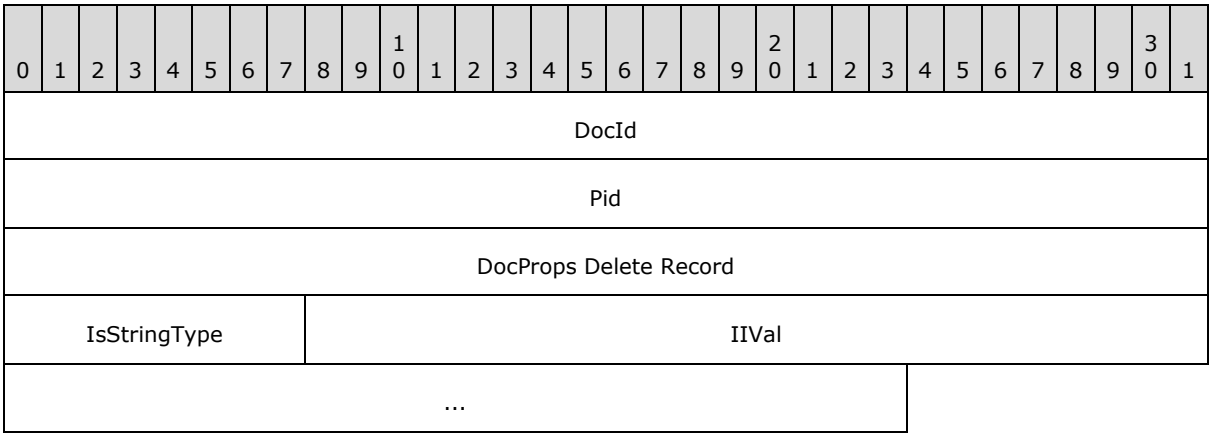


DocProps Delete Record (8 bytes): First **DocProps Delete Record**.

DocProps Delete Record (8 bytes): Second **DocProps Delete Record**.

2.2.3.7 DocProps Delete With Hash Record

Contains information needed to delete a row for a property of a document in MSSDocProps specified in [MS-SQLPQ2] section 2.2.5.2.



DocId (4 bytes): A 4-byte integer in big-endian format that is compared to the **docid** column of the **MSSDocProps** table which identifies a document.

Pid (4 bytes): A 4-byte integer in big-endian format that is compared to the **pid** column of the **MSSDocProps** table which identifies a managed property of the document identified in **DocId**.

IsStringType (1 byte): A 1-byte integer that MUST be 1 when the property type is a string and MUST be 0 when the property type is not a string.

llVal (8 bytes): When **IsStringType** is equal to 1, an 8-byte integer in big-endian format whose high order 4 bytes are compared to the high order 4 bytes of the **llVal** column of the **MSSDocProps** table, when **IsStringType** is set to zero, an 8-byte integer in big-endian format which is compared to the **llVal** column of the **MSSDocProps** table.

2.2.3.8 DocProps Delete With Hash Structure

An array of DocProps Delete With Hash Records, as specified in section 2.2.3.7.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
DocProps Delete With Hash Record (17 bytes)																															
...																															
DocProps Delete With Hash Record (17 bytes)																															
...																															

DocProps Delete With Hash Record (17 bytes): First **DocProps Delete With Hash Record**.

DocProps Delete With Hash Record (17 bytes): Second **DocProps Delete With Hash Record**.

2.2.3.9 DocResults Update Record

Contains information needed to update a row for a document identified by the DocId column in MSSDocResults specified in [MS-SQLPQ2] section 2.2.5.2.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
DocId																															
SummaryBlobSize																															
Size																															
...																															
Last Modified																															
...																															
IsDocument									IsPictureUrl																						
...																															
Author Offset																															
Author Length																															
Title Offset																															
Title Length																															
Url Offset																															
...																															
Url Length																															
Picture Thumbnail Url Offset																															

File Extension Length	Content Class Offset
...	Content Class Length
File Extension Offset	
File Extension Length	Tags Offset
...	Tags Length
Property Blob Offset	
Property Blob Length	
Unused Place Holder	
...	
Site Name Offset	
Site Name Length	Description Offset
...	Description Length
Unused Place Holder 2	
...	Number Of Members
...	Picture Height And Width
...	Display Date
...	

DocId (4 bytes): A 4-byte integer in big-endian format that is used to compare to the **docid** column of the **MSSDocResults** table.

Summary Blob Size (4 bytes): A 4-byte integer in big-endian format that is used to update the **summaryBlobSize** column of the **MSSDocResults** table.

Size (8 bytes): An 8-byte integer in big-endian format that is used to update the size column of the **MSSDocResults** table.

Last Modified (8 bytes): An 8-byte integer in big-endian format that is used to update the lastModified column of the **MSSDocResults** table.

Is Document (1 byte): A 1-byte integer that is used to update the **isDocument** column of the **MSSDocResults** table.

Is PictureUrl (1 byte): A 1-byte integer that is used to update the **isPictureUrl** column of the **MSSDocResults** table.

Author Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the author column of the **MSSDocResults** table starts. MUST be ignored if the **Author Length** is 0.

Author Length (2 bytes): MUST be a 2-byte integer in big-endian format that is the length of the string that is used to update the author column of the **MSSDocResults** table. When the length is 0, a null value MUST be assigned to the string.

Title Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the title column of the **MSSDocResults** table starts. MUST be ignored if the **Title Length** is 0.

Title Length (2 bytes): MUST be a 2-byte integer in big-endian format that is the length of the string that is used to update the title column of the **MSSDocResults** table. When the length is 0, a null value MUST be assigned to the string.

Url Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the **url** column of the **MSSDocResults** table starts. MUST be ignored if the **Url Length** is 0.

Url Length (2 bytes): MUST be a 2-byte integer in big-endian format that is the length of the string that is used to update the **url** column of the **MSSDocResults** table. When the length is 0, a null value MUST be assigned to the string.

Picture Thumbnail Url Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the **pictureThumbnailUrl** column of the **MSSDocResults** table starts. MUST be ignored if the **Picture Thumbnail Url Length** is 0.

Picture Thumbnail Url Length (2 bytes): MUST be a 2-byte integer in big-endian format that is the length of the string that is used to update the **pictureThumbnailUrl** column of the **MSSDocResults** table. When the length is 0, a null value MUST be assigned to the string.

Content Class Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the **contentClass** column of the **MSSDocResults** table starts. MUST be ignored if the **Content Class Length** is 0.

Content Class Length (2 bytes): MUST be a 2-byte integer in big-endian format that is the length of the string that is used to update the **contentClass** column of the **MSSDocResults** table. When the length is 0, a null value MUST be assigned to the string.

File Extension Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the **fileExtension** column of the **MSSDocResults** table starts. MUST be ignored if the **File Extension Length** is 0.

File Extension Length (2 bytes): MUST be a 2-byte integer in big-endian format that is the length of the string that is used to update the **fileExtension** column of the **MSSDocResults** table. When the length is 0, a null value MUST be assigned to the string.

Tags Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the tags column of the **MSSDocResults** table starts. MUST be ignored if **Tags Length** is 0.

Tags Length (2 bytes): MUST be a 2-byte integer in big-endian format that is the length of the string that is used to update the tags column of the **MSSDocResults** table. When the length is 0, a null value MUST be assigned to the string.

Property BLOB Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a binary stream of bytes where the byte array that is used to update the **propertyBlob** column of the **MSSDocResults** table starts. MUST be ignored if **Property Blob Length** is 0.

Property BLOB Length (4 bytes): MUST be a 4-byte integer in big-endian format that is the length of the byte array that is used to update the **propertyBlob** column of the **MSSDocResults** table. When the length is 0, a null value MUST be assigned to the column.

Unused Place Holder (8 bytes): Unused 8 bytes that MUST be ignored.

Site Name Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the **siteName** column of the **MSSDocResults** table starts. MUST be ignored if **Site Name Length** is 0.

Site Name Length (2 bytes): MUST be a 2-byte integer in big-endian format that is the length of the string that is used to update the **siteName** column of the **MSSDocResults** table. When the length is 0, a null value MUST be assigned to the string.

Description Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the description column of the **MSSDocResults** table starts. MUST be ignored if **Description Length** is 0.

Description Length (2 bytes): MUST be a 2-byte integer in big-endian format that is the length of the string that is used to update the description column of the **MSSDocResults** table. When the length is 0, a null value MUST be assigned to the string.

Unused Place Holder 2 (6 bytes): Unused 6 bytes that MUST be ignored.

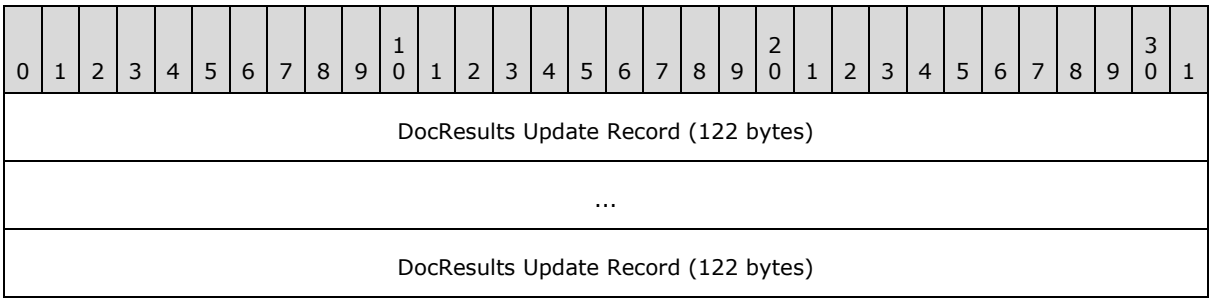
Number Of Members (4 bytes): A 4-byte integer in big-endian format that is used to update the **numberOfMembers** column of the **MSSDocResults** table.

Picture Height and Width (8 bytes): An 8-byte integer in big-endian format that is used to update the **pictureHeightAndWidth** column of the **MSSDocResults** table.

Display Date (8 bytes): An 8-byte in big-endian format that is used to update the **displayDate** column of the **MSSDocResults** table.

2.2.3.10 DocResults Update Structure

An array of **DocResults Update Records**, as specified in section 2.2.3.9.



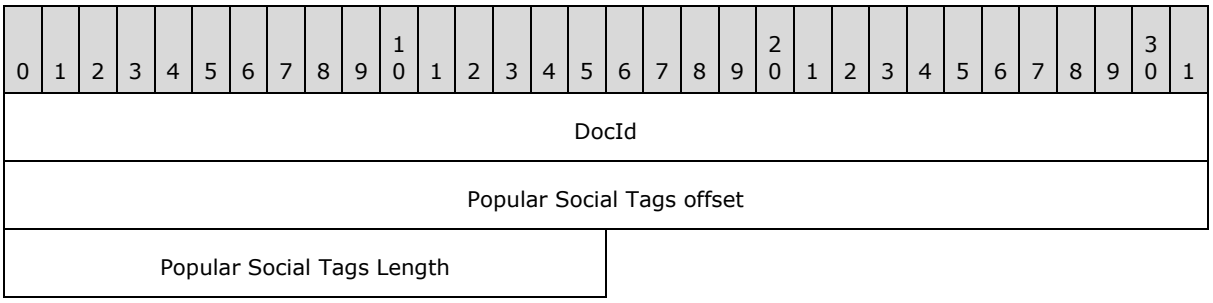
…

DocResults Update Record (122 bytes): **First DocResults Update Record**.

DocResults Update Record (122 bytes): Second **DocResults Update Record**.

2.2.3.11 DocResults Anchor Update Record

Contains information needed to update the popular social tags for a document identified by the DocId column in MSSDocResults specified in [MS-SQLPQ2] section 2.2.5.2.



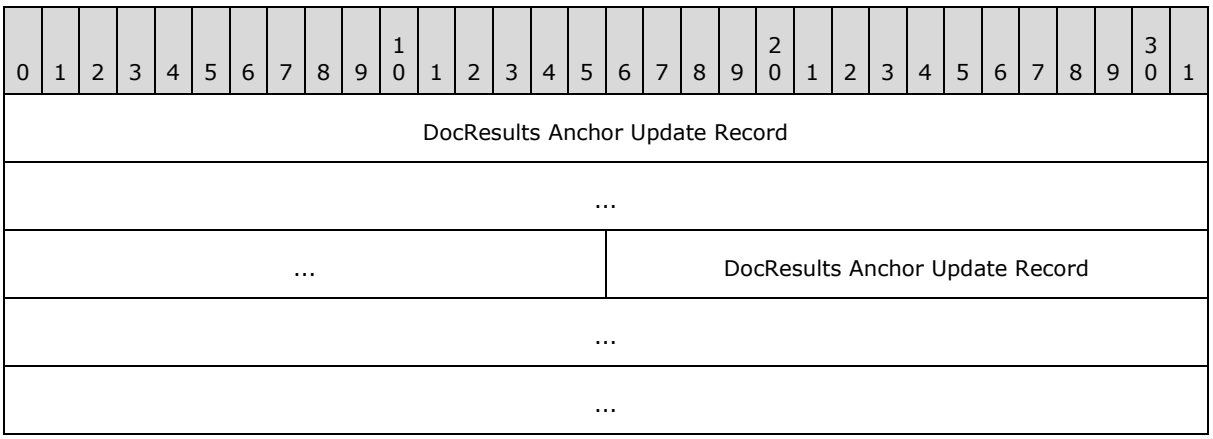
DocId (4 bytes): A 4-byte integer in big-endian format that is compared to the **docid** column of the **MSSDocResults** table.

Popular Social Tags Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the **popularSocialTags** column of the **MSSDocResults** table starts. MUST be ignored if the **Popular Social Tags Length** is 0.

Popular Social Tags Length (2 bytes): MUST be a 2-byte integer in big-endian format that is the length of the string that is used to update the **popularSocialTags** column of the **MSSDocResults** table. When the length is 0, a null value MUST be assigned to the string.

2.2.3.12 DocResults Anchor Update Structure

An array of **DocResults Anchor Update Records**, as specified in section 2.2.3.11.

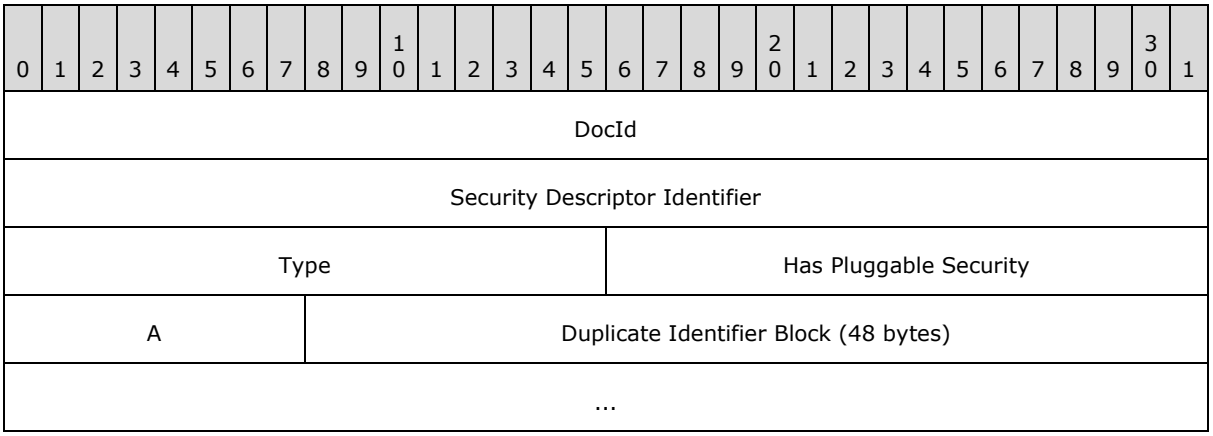


DocResults Update Record (10 bytes): First **DocResults Update Record**.

DocResults Update Record (10 bytes): Second DocResults Update Record.

2.2.3.13 DocSdIds Update Record

Contains information needed to update or insert a row for a document identified by the DocId column in MSSDocSdIds specified in [MS-SQLPQ2] section 2.2.5.3.



DocId (4 bytes): A 4-byte integer in big-endian format that is compared to or inserted in the **docid** of the **MSSDocSdIds** table.

Security Descriptor Identifier (4 bytes): A 4-byte integer in big-endian format that updates the security descriptor identifier of the **MSSDocSdIds** table for the document identified by **docid**.

Type (2 bytes): A 2-byte integer in big-endian format that updates the type column of the **MSSDocSdIds** table for the document identified by **docid**.

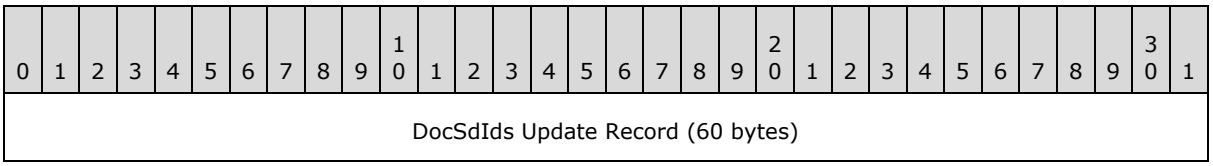
Has Pluggable Security (2 bytes): A 2-byte integer in big-endian format that updates the has pluggable security column of the **MSSDocSdIds** table for the document identified by **docid**.

A - Duplicate Identifiers Action (1 byte): A 1-byte integer that describes how the **duplicateHashes** column of the **MSSDocSdIds** table is updated. MUST be 2 when the **duplicateHashes** column is not changed and **Duplicate Identifier Block** is ignored. MUST be 1 when the **Duplicate Identifier Block** is used to update the **duplicateHashes** column of the **MSSDocSdIds** table. Otherwise, a NULL is used to update the **duplicateHashes** column of the **MSSDocSdIds** table.

Duplicate Identifier Block (48 bytes): A **Duplicate Identifier Block** as specified in [MS-SQLPQ2] section 2.2.1.1.

2.2.3.14 DocSdIds Update Structure

An array of **DocSdIds Update Records**, as specified in section 2.2.3.13.



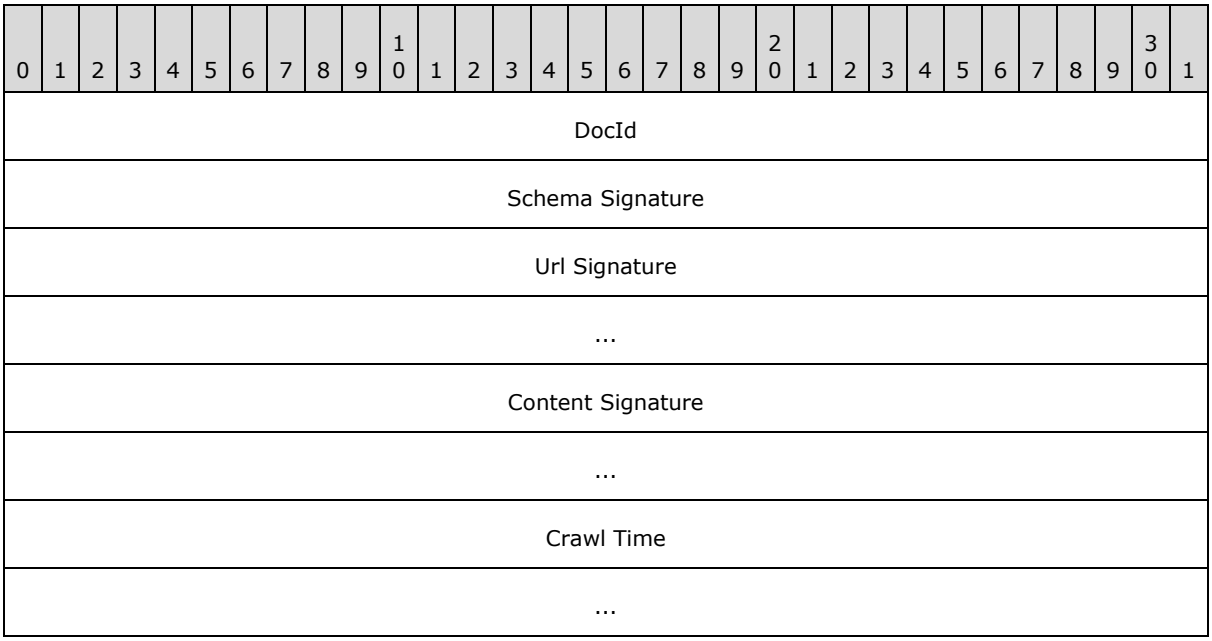
...
DocSdIds Update Record (60 bytes)
...

DocSdIds Update Record (60 bytes): First **DocSdIds Update Record**.

DocSdIds Update Record (60 bytes): Second **DocSdIds Update Record**.

2.2.3.15 Alert History Update Record

Contains identifiers for a document's **URL**, metadata property combination, content and the date and time that the document is last crawled.



DocId (4 bytes): A 4-byte integer in big-endian format that identifies a document.

Schema Signature (4 bytes): A 4-byte signature determined by the client protocol in big-endian format that identifies the combination of metadata properties for the document identified by **docid**.

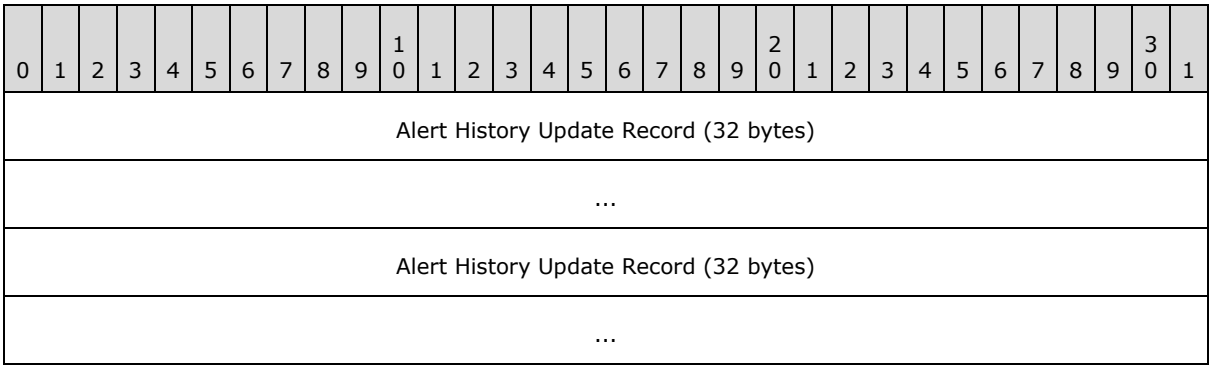
Url Signature (8 bytes): An 8-byte signature determined by the client protocol in big-endian format that identifies the URL of the document identified by **docid**.

Content Signature (8 bytes): An 8-byte signature determined by the client protocol in big-endian format that identifies the content of the document identified by **docid**.

CrawlTime (8 bytes): An 8-byte **Managed Property Time** as specified in [MS-SQLPQ2] section 2.2.1.5 that is the date and time in **UTC** when the document was last crawled.

2.2.3.16 Alert History Update Structure

An array of **Alert History Update Records**, as specified in section 2.2.3.15.

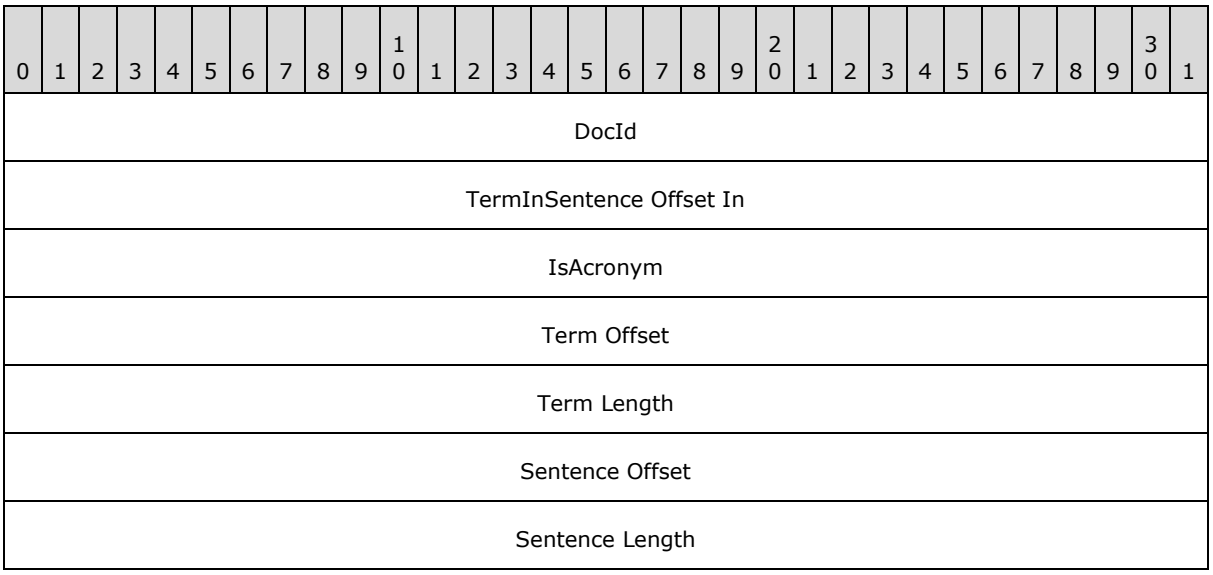


Alert History Update Record (32 bytes): First **Alert History Update Record**.

Alert History Update Record (32 bytes): Second **Alert History Update Record**.

2.2.3.17 Definitions Update Record

Contains information needed to update a row for a document identified by the **DocId** attribute specified in the Definitions set section 3.1.1.40. The row contains a term and a sentence that defines the term. The term string is extracted from a stream of Unicode characters using the **Term Offset** and **Term Length**. The sentence string is extracted from the same stream of Unicode characters using the **Sentence Offset** and **Sentence Length**.



DocId (4 bytes): A 4-byte integer in big-endian format that identifies a document.

TermInSentence Offset (4 bytes): A 4-byte integer that MUST be the offset of the term within the sentence string.

IsAcronym (4 bytes): A 4-byte integer that MUST be 1 when the term is an acronym and MUST be 0 otherwise.

Term Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the term attribute of the **Definitions** set starts. MUST be ignored if **Term Length** is 0.

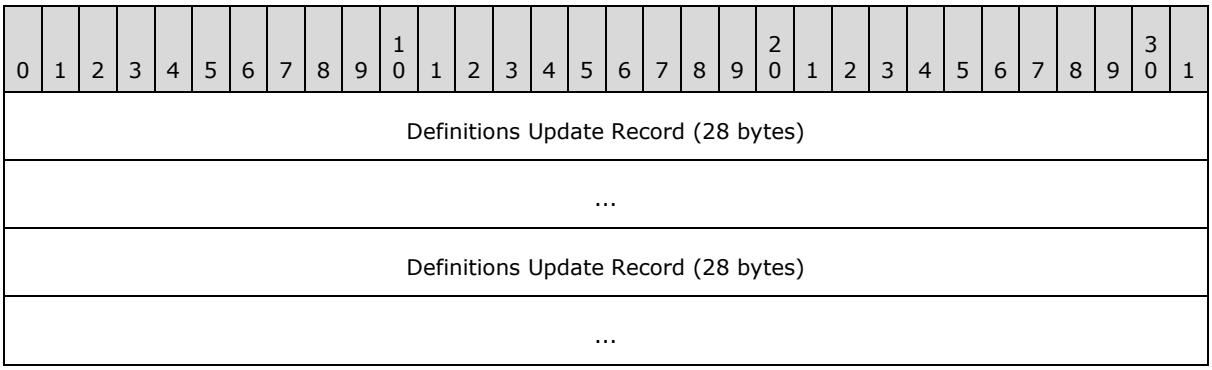
Term Length (4 bytes): MUST be a 4-byte integer in big-endian format that is the length of the string that is used to update the term attribute of the **Definitions** set. When the length is 0, a null value MUST be assigned to the string.

Sentence Offset (4 bytes): MUST be a 4-byte integer in big-endian format that is the offset within a stream of Unicode characters where the string that is used to update the sentence attribute of the **Definitions** set starts. MUST be ignored if **Sentence Length** is 0.

Sentence Length (4 bytes): MUST be a 4-byte integer in big-endian format that is the length of the string that is used to update the sentence attribute of the **Definitions** set. When the length is 0, a null value MUST be assigned to the string.

2.2.3.18 Definitions Update Structure

An array of **Definitions Update Records**, as specified in section 2.2.3.17.

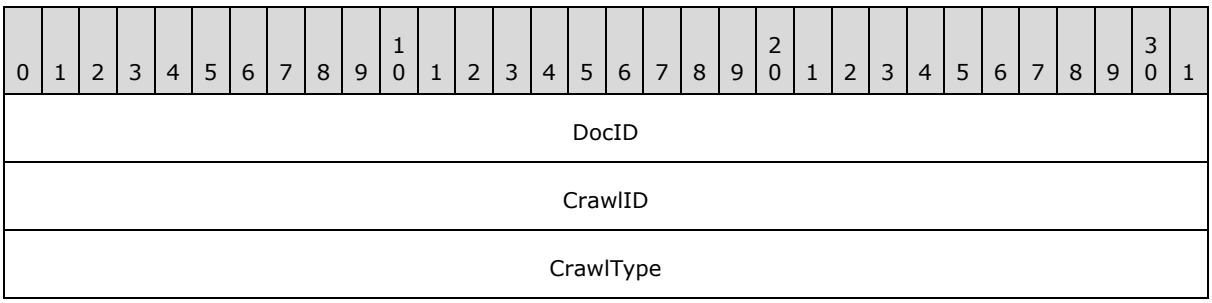


Definitions Update Record (28 bytes): First **Definitions Update Record**.

Definitions Update Record (28 bytes): Second **Definitions Update Record**.

2.2.3.19 Transaction Commit Record

Contains all the information needed to commit a transaction (1) including the offsets and lengths needed to access Transaction Data Blob, as specified in section 2.2.3.21, and the Transaction String, as specified in section 2.2.3.22.



TransactionType	
Scope	
TransactionFlags	
CompactURL offset	
CompactURL length	CompactHash
...	ParentCompactURL offset
...	ParentCompactURL length
ParentCompactHash	
DisplayURL offset	
DisplayURL length	DisplayHash
...	LastModifiedTime
...	EndPathFlag
...	PropMD5
...	Unused Place Holder
...	HostDepth
...	EnumerateDepth
...	RetryCount
...	DelayRetryCount
...	IndexType
...	SeqID
...	
...	UseChangeLog
...	ChangeLogBatchID

...	ChangeLogCookie offset
	ChangeLogCookie length
...	ChangeLogCookieType
...	ErrorDescription offset
...	ErrorDescription length
hrResult	
DocPropsMD5	
DocPropsBlob offset	
DocPropsBlob length	
TransactionStatus	
ErrorID	
ErrorLevel	

MarkDelete	LinksBitmap	
...	Title offset	
...	Title length	LCIDTitle
...		SecurityID offset
...		SecurityID length

...	PHFlags
...	UseSecurityInfo
...	ProtocolLength
...	LogLevel
...	ErrorSource
...	RecrawlErrorCount

...	RecrawlErrorInterval
...	ErrorIntervalAllowed
...	ErrorCountAllowed
...	ErrorDeleteCountAllowed
...	ErrorDeleteIntervalAllowed
...	ChangeLogCookieEnd offset
...	ChangeLogCookieEnd length
...	ChildSecurityBlob offset
...	ChildSecurityBlob length
...	

DocID (4 bytes): MUST be a 4-byte integer in big-endian format, the identifier of the document.

CrawlID (4 bytes): MUST be a 4-byte integer in big-endian format, the unique identifier of the crawl.

CrawlType (4 bytes): MUST be a 4-byte integer in big-endian format, the type of the crawl. The value of this parameter MUST be a **Crawl Type** data type as specified in Section 2.2.1.2.

TransactionType (4 bytes): MUST be a 4-byte integer in big-endian format, the type of transaction. The value of this parameter MUST be a **Transaction Type** data type as specified in Section 2.2.1.14.

Scope (4 bytes): MUST be a 4-byte integer in big-endian format, the transaction scope. The value of this parameter MUST be a **Transaction Scope Type** as specified in Section 2.2.1.15.

TransactionFlags (4 bytes): MUST be a 4-byte integer in big-endian format, the bit mask of flags for the crawl transaction. The value of this parameter MUST be a valid **Transaction Flags** type as specified in Section 2.2.2.3.

CompactURL offset (4 bytes): MUST be a 4-byte integer in big-endian format, in case **CompactURL** length is greater than 0 defines offset of **CompactURL** in the **Transaction** string, as specified in section 2.2.3.22.

CompactURL length (2 bytes): MUST be a 2-byte integer in big-endian format, length of **CompactURL** in the **Transaction** string, as specified in section 2.2.3.22.

CompactHash (4 bytes): MUST be a 4-byte integer in big-endian format, the signature of the **CompactURL** of the item, as specified in section 3.1.1.5.

ParentCompactURL offset (4 bytes): MUST be a 4-byte integer in big-endian format, in case **ParentCompactURL** length is greater than 0 defines the offset of **ParentCompactURL** in the **Transaction** string, as specified in section 2.2.3.22.

ParentCompactURL length (2 bytes): MUST be a 2-byte integer in big-endian format, length of **ParentCompactURL** in the **Transaction** string, as specified in section 2.2.3.22.

ParentCompactHash (4 bytes): MUST be a 4-byte integer in big-endian format, the signature of **ParentCompactURL**, The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same URL.

DisplayURL offset (4 bytes): MUST be a 4-byte integer in big-endian format, in case **DisplayURL** length is greater than 0 defines offset of **DisplayURL** in the **Transaction** string, as specified in section 2.2.3.22.

DisplayURL length (2 bytes): MUST be a 2-byte integer in big-endian format, length of **DisplayURL** in the **Transaction** string, as specified in section 2.2.3.22.

DisplayHash (4 bytes): MUST be a 4-byte integer in big-endian format, the hash of the display URL of the item.

LastModifiedTime (4 bytes): MUST be a 4-byte integer in big-endian format, a UTC that specifies when the item was modified.

EndPathFlag (4 bytes): MUST be 4-byte integer in big-endian format, the value of this parameter MUST be a valid **End Path Flag** as specified in Section 2.2.2.4.

PropMD5 (4 bytes): MUST be a 4-byte integer in big-endian format, the signature of the item metadata, as specified in section 3.1.1.5.

MD5 (4 bytes): MUST be a 4-byte integer in big-endian format, the signature of the item content, as specified in section 3.1.1.5.

Unused Place Holder (4 bytes): Unused 4-byte integer in big-endian format that MUST be ignored.

HostDepth (4 bytes): MUST be a 4-byte integer in big-endian format, the value that specifies how many hosts the crawl should traverse when discovering links.

EnumerationDepth (4 bytes): MUST be a 4-byte integer in big-endian format, the value that shows how many links are followed during the crawl.

RetryCount (4 bytes): MUST be a 4-byte integer in big-endian format, count of times the **search service instance** tried to process the document.

DelayRetryCount (4 bytes): MUST be a 4-byte integer in big-endian format, count of retries that internal search service components required during the processing of this document.

IndexType (4 bytes): MUST be a 4-byte integer in big-endian format, the value of this parameter MUST be a valid Index Type as specified in Section 2.2.1.20.

SeqID (8 bytes): MUST be an 8-byte integer in big-endian format, the identifier of the record in **Crawl Queue** as specified in section 3.1.1.4.

UseChangeLog (4 bytes): MUST be a 4-byte integer in big-endian format, an integer which MUST be 1 if the item belongs to a site that supports incremental crawl based on change log. Otherwise, it MUST be 0.

ChangeLogBatchID (4 bytes): MUST be a 4-byte integer in big-endian format, this parameter MUST be ignored if **@UseChangeLog** is equal to 0. The identifier of the subset of the change log to which the current item belongs.

ChangeLogCookie offset (4 bytes): MUST be a 4-byte integer in big-endian format, in case **ChangeLogCookie length** is greater than 0 defines offset of **ChangeLogCookie** in transaction data blob, as specified in section 2.2.3.21.

ChangeLogCookie length (4 bytes): MUST be a 4-byte integer in big-endian format, length of **ChangeLogCookie** in transaction data blob, as specified in section 2.2.3.21.

ChangeLogCookieType (4 bytes): MUST be a 4-byte integer in big-endian format, this parameter MUST be ignored when **UseChangeLog** equals 0. When **UseChangeLog** equals 1, this parameter contains the type of **ChangeLogCookie**, as specified in section 3.1.1.5.

ErrorDescription offset (4 bytes): MUST be a 4-byte integer in big-endian format, in case **ErrorDescription length** is greater than 0 defines offset of **ErrorDescription** in the **Transaction** string, as specified in section 2.2.3.22.

ErrorDescription length (2 bytes): MUST be a 2-byte integer in big-endian format, length of **ErrorDescription** in the **Transaction** string, as specified in section 2.2.3.22.

hrResult (4 bytes): MUST be a 4-byte integer in big-endian format, code of an error which occurred during document processing as specified in section 3.1.1.5. If there were no errors, this parameter is 0.

DocPropsMD5 (4 bytes): MUST be a 4-byte integer in big-endian format, the identifier of the item metadata.

DocPropsBlob offset (4 bytes): MUST be a 4-byte integer in big-endian format if **DocPropsBlob length** is greater than 0 defines the offset of **DocPropsBlob** in the data blob, as specified in section 2.2.3.21

DocPropsBlob length (4 bytes): MUST be a 4-byte integer in big-endian format that represents the length of **DocPropsBlob** in the data blob, as specified in section 2.2.3.21

TransactionStatus (4 bytes): MUST be a 4-byte integer in big-endian format that contains the status of the transaction.

ErrorID (4 bytes): MUST be a 4-byte integer in big-endian format, identifier of the error from the **Crawl Error Set** that occurred in process of crawling of this document. Is set to 0 if there were no errors.

ErrorLevel (4 bytes): MUST be a 4-byte integer in big-endian format, level of the error that occurred while crawling this document (as specified in section 3.1.1.5). Is set to 0 if there were no errors.

MarkDelete (1 byte): MUST be a 1-byte integer in big-endian format. The value MUST be set to 1 if there is a delete request, and it MUST be set to 0 otherwise.

LinksBitmap (4 bytes): MUST be a 4-byte integer in big-endian format, the crawl stores that have links to this item as specified in section 3.1.1.5.

Title offset (4 bytes): MUST be a 4-byte integer in big-endian format. If the **Title** length is greater than 0, it defines the offset of **Title** in the **Transaction** string, as specified in section 2.2.3.22.

Title length (2 bytes): MUST be a 2-byte integer in big-endian format, length of **Title** in the **Transaction** string, as specified in section 2.2.3.22.

LCIDTitle (4 bytes): Language identifier of **Title** in the transaction string, as specified in section 2.2.3.22.

SecurityID offset (4 bytes): MUST be a 4-byte integer in big-endian format. If the **SecurityID** length is greater than 0, it defines the offset of **SecurityID** in the **Transaction** string, as specified in section 2.2.3.22.

SecurityID length (2 bytes): MUST be a 2-byte integer in big-endian format that defines the length of **SecurityID** in the **Transaction** string, as specified in section 2.2.3.22.

PHFlags (4 bytes): MUST be a 4-byte integer in big-endian format that defines flags used by the protocol handler. The value of this field is implementation details of the protocol client.

UseSecurityInfo (4 bytes): MUST be a 4-byte integer in big-endian format. The value is not used but SHOULD be 0.

ProtocolLength (4 bytes): MUST be a 4-byte integer in big-endian format that defines the number of characters from the beginning of **DisplayURL** to the first colon, excluding the colon itself.

LogLevel (4 bytes): MUST be a 4-byte integer in big-endian format that defines the error logging level as specified in 2.2.1.9.

ErrorSource (4 bytes): MUST be a 4-byte integer in big-endian format that is a unique identifier of the internal search component that reported the error for the item.

RecrawlErrorCount (4 bytes): MUST be a 4-byte integer in big-endian format that defines the number attempts to crawl the item that SHOULD be made before the change log cookie MUST be set to NULL.

RecrawlErrorInterval (4 bytes): MUST be a 4-byte integer in big-endian format that defines the time interval, in hours, after which the first error in the **ChangeLogCookie** MUST be set to NULL.

ErrorIntervalAllowed (4 bytes): MUST be a 4-byte integer in big-endian format that defines the time interval, in hours, after what the first error in the document is deleted unless this error is either **Access Denied Error** or **Object Not Found Error** and the value of **ErrorDeleteIntervalAllowed** is greater than the value of **ErrorIntervalAllowed**.

ErrorCountAllowed (4 bytes): MUST be a 4-byte integer in big-endian format that defines the number of errors in the document that are deleted unless this error is either **Access Denied Error** or **Object Not Found Error** and the value of **@ErrorDeleteCountAllowed** is greater than **@ErrorCountAllowed**.

ErrorDeleteCountAllowed (4 bytes): MUST be a 4-byte integer in big-endian format that defines the number of errors allowed in the document unless they are either **Access Denied Error** or **Object Not Found Error** unless the value of **@ErrorCountAllowed** is greater than **@ErrorDeleteCountAllowed**.

ErrorDeleteIntervalAllowed (4 bytes): MUST be a 4-byte integer in big-endian format that defines the time interval, in hours, after which the first error in the document is deleted that is either **Access Denied** or **Object Not Found**. One of these errors SHOULD be deleted unless the value of **ErrorIntervalAllowed** is greater than **ErrorDeleteIntervalAllowed**.

ChangeLogCookieEnd offset (4 bytes): MUST be a 4-byte integer in big-endian format, in case **ChangeLogCookieEnd length** is greater than 0 defines offset of **ChangeLogCookieEnd** in transaction data blob, as specified in section 2.2.3.21.

ChangeLogCookieEnd length (4 bytes): MUST be a 4-byte integer in big-endian format that defines the length of **ChangeLogCookieEnd** in the transaction data blob, as specified in section 2.2.3.21.

ChildSecurityBlob offset (4 bytes): MUST be a 4-byte integer in big-endian format, in case **ChildSecurityBlob** length is greater than 0 defines offset of **ChildSecurityBlob** in transaction data blob, as specified in section 2.2.3.21.

ChildSecurityBlob length (4 bytes): MUST be a 4-byte integer in big-endian format that defines the length of **ChildSecurityBlob** in the transaction data blob, as specified in section 2.2.3.21.

2.2.3.20 Transactions Commit Structure

An array of **Transaction Commit Records**, as specified in section 2.2.3.19.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Transaction Commit Record (245 bytes)																															
...																															
Transaction Commit Record (245 bytes)																															
...																															

Transaction Commit Record (245 bytes): First **Transaction Commit Record**.

Transaction Commit Record (245 bytes): Second **Transaction Commit Record**.

2.2.3.21 Transaction Data Blob

Data BLOB, as specified in section 2.2.3.19.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
ChangeLogCookie (optional)																															
DocPropBlob (optional)																															
...																															
ChangeLogCookieEnd (optional)																															
ChildSecurityBlob (optional)																															

...

2.2.3.22 Transaction string

UNICODE string as specified in section 2.2.3.19.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
CompactURL (optional)																															
ParentCompactURL (optional)																															
DisplayURL (optional)																															
ErrorDescription (optional)																															
Title (optional)																															
SecurityID (optional)																															
...																															

2.2.4 Result Sets

2.2.4.1 Scope Compilation Result Set

The **Scope Compilation Result Set** MUST contain one row corresponding to each search scope.

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

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

ScopeID: An integer that uniquely identifies a search scope.

CompilationState: The search scope compilation state for the search scope. The value MUST be a Compilation State data type as specified in [MS-SQLPADM2] section 2.2.1.4.

2.2.5 Tables and Views

No common table or view structures are defined in this protocol.

2.2.6 XML Structures

No common XML structures are defined in this protocol.

2.2.6.1 Namespaces

None.

2.2.6.2 Simple Types

None.

2.2.6.3 Complex Types

None.

2.2.6.4 Elements

None.

2.2.6.5 Attributes

None.

2.2.6.6 Groups

None.

2.2.6.7 Attribute Groups

None.

3 Protocol Details

3.1 Server Details for Office SharePoint Server

Microsoft® SharePoint® 2010 Products and Technologies role is described in this section. SharePoint 2010 Products role serves requests for search crawl operations.

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 Crawl Status

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

- **CrawlID:** crawl unique identifier
- **ProjectID:** project identifier as specified in section 2.2.1.1
- **CrawlType:** crawl type as specified in section 2.2.1.2
- **ContentSourceID:** identifier of the content source associated with the crawl
- **Status:** crawl status as specified in section 2.2.1.4
- **SubStatus:** crawl sub status. Depending on the value of Status the value of SubStatus MUST belong to the following sets:
  - Crawl Starting Sub Status if Status equals 1, as specified in section 2.2.1.5
  - Crawl Crawling Sub Status or Crawl Completing Sub Status or Crawl Stopping Sub Status if Status equals 4, as specified in section 2.2.1.6
  - Not used for other values of Status
- **Request:** request for crawl action, for details see Crawl Request Type as specified in section 2.2.1.3
- **RequestTime:** time the request for crawl action was issued
- **StartTime:** time the crawl was started
- **EndTime:** time the crawl finished
- **MainCrawlID:** in case ProjectID is 2 (Anchor Project), MainCrawlID contains the unique identifier of the crawl with ProjectID equals 1 (Portal Content) which started the current crawl

3.1.1.2 Crawl Url History

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

- **DocID:** unique identifier of the item when ProjectID equals 1, -1 when ProjectID equals 2.
- **StartAddressID:** unique identifier of the start address, from which the item was discovered.
- **ContentSourceID:** unique identifier of the content source, that contains the start address whose identifier is the startaddressID.
- **ProjectID:** project identifier, for details see Project Identifier as specified in section 2.2.1.1.
- **CrawlID:** identifier of the crawl that is going to process the item.
- **SecurityUpdateCrawlID:** identifier of the crawl which is going to update security information of the item.
- **AccessURL:** the item's access URL, which is used internally to retrieve the item.
- **DisplayURL:** the item's display URL, which is shown in search results and crawl log.
- **CompactURL:** compact identifier of the item.
- **TransactionFlags:** See the explanation of the transaction flags as specified in section 2.2.2.3.
- **UseChangeLog:** An integer which MUST be 1 if the item belongs to a site that supports incremental crawl based on change log. Otherwise, it MUST be 0.
- **ChangeLogCookie:** in case where UseChangeLog is 1 ChangeLogCookie contains a BLOB which was obtained during the last crawl which processed the item, MUST be NULL otherwise. The format of the BLOB is an implementation detail of the protocol client.
- **ChangeLogCookieType:** In case where UseChangeLog is 1, ChangeLogCookieType contains the type of ChangeLogCookie, MUST be NULL otherwise. The values of the ChangeLogCookieType is implementation detail of protocol client.
- **HostDepth:** specifies how many hosts the crawl should traverse when discovering links.
- **EnumerationDepth:** shows how many links are followed during the crawl.
- **DeletePending:** identifies if the request to delete this item was issued.
- **ErrorLevel:** level of the error which occurred in process of crawling this document as specified in section 2.2.1.10.
- **DisplayHash:** signature of the DisplayURL of the item. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same URL.
- **MD5:** The signature of the item content. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same content of the document.
- **PropMD5:** The signature of the item metadata. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same metadata of the document. In the incremental crawl if the value of the property is different than the existing value the item and any child objects MUST be re-crawled.
- **EndPathFlag:** see explanation in End Path Flags as specified in section 2.2.2.4.
- **IndexType:** see explanation Index Type as specified in section 2.2.1.20.

- **LastModifiedTime:** A UTC time that indicates when the item was modified.
- **LCID:** **LCID** of the document content.
- **ItemType:** see explanation of Item Type as specified in section 2.2.1.21.
- **DocPropsMD5:** The signature of the item properties. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same metadata of the document.
- **Retry:** counter of time processing of the document by the search service has failed.
- **RetryCount:** counter of times the search service tried to process the document.
- **DocPropsBlob:** BLOB containing properties of the document, the format of the BLOB is implementation detail of the protocol client.
- **HostID:** Identifier of the host from Crawled Host set as specified in section 3.1.1.7 which contains this document.
- **ParentHostID:** unique identifier of the host name which contains a document which has a link to this document.
- **LinksBitmap:** the crawl stores which have links to this item as specified in section 2.2.2.2.
- **ParentUpdateCrawlID:** unique identifier of the last crawl which process parent of this document.
- **CommitCrawlID:** unique identifier of the last crawl to process this document.
- **ParentDocID:** unique identifier of document which is the parent of this document.
- **SecurityID:** security identifier, used by internal component to the search service instance.
- **PHFlags:** flags used by internal component to the search service instance.
- **DelayRetryCount:** Count of retries that internal search service components required during the processing of this document.
- **ErrorDeleteCount:** counter of errors which MAY result in deleting of the document after number of errors exceed an implementation specific threshold.
- **FirstErrorDeleteTime:** A UTC time that specifies when the first error which MAY result in the deletion of the document has occurred.
- **ErrorCount:** count of all errors which happened while processing the document.
- **FirstErrorTime:** A UTC time that specifies when the first error with this document occurred.
- **AccessHash:** signature of AccessURL. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same AccessURL.
- **Title:** title of the document discovered during the crawl.
- **TitleLCID:** LCID of the title.

- **FolderDelCount:** in case the item is a folder MAY specifies number of deletes which happened inside this folder. When this counter changes all items contained in the folder MUST be checked whether they were deleted or not.
- **SecurityUpdateErrorID:** error identifier, from Crawl Error Set, as specified in [MS-SQLPADM2] section 3.1.1.3.
- **SiteID:** A unique identifier of the site where the document was discovered.
- **CompactHash:** signature of CompactURL. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same CompactURL.
- **ErrorID:** error identifier, from Crawl Error Set, as specified in [MS-SQLPADM2] section 3.1.1.3.

3.1.1.3 Deleted URL

Deleted URL Set: Deleted URL Set maintains list of URLs deleted during the crawl

- **TrackID:** unique identifier of deleted item
- **AccessURL:** the item's access URL, which is used internally to retrieve the item
- **AccessHash:** signature of AccessURL. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same AccessURL.
- **HostID:** Identifier of the host from Crawled Host set as specified in section 3.1.1.7 which contains this document
- **ContentSourceID:** identifier of content source of deleted item from which the deleted item has been discovered

3.1.1.4 Crawl Queue

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

- **CrawlID:** Crawl unique identifier
- **DocID:** Item unique identifier
- **SourceDocID:** **document identifier (1)** of the parent item
- **StartAddressID:** Identifier of start address from which the item was discovered
- **ContentSourceID:** Identifier of content source, that contains the start address whose identifier is the startaddressID
- **ProjectID:** project identifier, for details see Project Identifier as specified in section 2.2.1.1
- **TransactionType:** As specified in section 2.2.1.14
- **Scope:** As specified in section 2.2.1.15
- **BatchID:** unique identifier of the batch that this document belongs to, 0 if the item does not belong to any batches

- **ComponentID:** identifier of crawl component in Crawl Component Set ([MS-SRCHTP] section 3.1.1.3) which is crawling this item
- **HostDepth:** A number that specifies how many crawl hops between different hosts are allowed during crawl
- **EnumerationDepth:** A number specifying how many crawl hops are allowed during crawl
- **TransactionFlags:** As specified in section 2.2.2.3
- **SeqID:** The unique identifier of the entry inside the Crawl Queue
- **ChangeLogBatchID:** The identifier of the subset of the change log to which the current item belongs. This value is used by internal component to the search service instance
- **DeleteReason:** As specified in section 2.2.1.22
- **CachedBlob:** cached document from the queue, the format of the BLOB is implementation detail of the protocol client
- **SecurityUpdateCrawlID:** identifier of the crawl which is going to update security information of the item
- **CachedSecurityUpdateCrawlID:** identifier of the crawl which updated security information of the item
- **SecurityID:** security identifier, used by internal component to the search service instance

3.1.1.5 Links

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

- **CrawlID:** Crawl identifier
- **DocID:** unique identifier of the item when ProjectID equals 1 (Portal Content), -1 otherwise
- **SourceDocID:** Identifier of the source item
- **HostID:** Identifier of the host from Crawled Host set as specified in section 3.1.1.7 which contains this document
- **StartAddressID:** Identifier of start address from which the item was discovered
- **ContentSourceID:** Identifier of the content source identifier, that contains the start address whose identifier is the startaddressID
- **ProjectID:** Project identifier as specified in section 2.2.1.1
- **AccessURL:** the item's access URL which is used internally to retrieve the item
- **DisplayURL:** the item's display URL which is shown in search results and crawl log
- **AccessHash:** signature of AccessURL. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same AccessURL.

- **DisplayHash:** signature of DisplayURL. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same DisplayURL.
- **CompactURL:** the item's compact identifier
- **CompactHash:** signature of CompactURL. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same CompactURL.
- **ItemType:** as specified in section 2.2.1.21
- **CrawlType:** as specified in section 2.2.1.2
- **TransactionType:** as specified in section 2.2.1.14
- **TransactionFlags:** as specified in section 2.2.2.3
- **Scope:** a Transaction Scope as specified in section 2.2.1.15
- **HostDepth:** a count of the number of crawl hops between different hosts that are allowed during a crawl
- **EnumerationDepth:** a count of the number of crawl hops that are allowed during a crawl
- **EndPathFlag:** as specified in section 2.2.2.4
- **IndexType:** as specified in section 2.2.1.20
- **LCID:** LCID of the item content
- **UseChangeLog:** An integer which MUST be 1 if the item belongs to a site that supports incremental crawl based on change log. Otherwise, it MUST be 0.
- **hrResult:** code, as specified in section 2.2.1.11, of an error which occurred during the last crawl of the item
- **ParentProcessChangeLog:** If this value is 1, the link is a change that comes from the change log processed by the parent.
- **PropMD5:** An signature of the item metadata. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same document. In the incremental crawl, if the value of the parameter is different than the existing value, the item and any child objects will be re-crawled.
- **SiteID:** A unique identifier of the site where the link was discovered.
- **SourceHostID:** unique identifier of the host which contains source item
- **ProtocolSwitch:** if URL scheme of the item is different of the URL scheme of source item this value is 1 otherwise this value is 0
- **SourceIsStartAddress:** specify if source item is a start address
- **SourceHostHop:** value of HostHop of the source item
- **Host:** name of the host

- **ChangeLogBatchID:** the identifier of the subset of the change log to which the current item belongs. This value is used by internal component to the search service instance
- **CachedBlob:** cached document from the queue, the format of the BLOB is implementation detail of the protocol client
- **FirstLink:** ordinal of the link for the given source of the link

3.1.1.6 Colleague Links

Colleague Links Set: Colleague Links set maintains list of items specifying that the user with user profile identifier @SourceUserID is a colleague of the user with user profile identifier @TargetUserID,

- **CrawlID:** unique identifier crawl which will process the link
- **SourceDocID:** identifier of the document where link was found
- **SourceUserID:** The unique identifier of the user profile user corresponding to the document identifier @SourceDocId.
- **TargetUserID:** The **user profile record identifier** for the colleague of the user profile user @SourceUserID when @Pid is greater than 0.
- **Pid:** identifier if the relationship is public or private. Pid = 394 means public relationship; Pid=395 means private relationship
- **LinkID:** link unique identifier

3.1.1.7 Crawled Hosts

Crawled Hosts Set: The crawled Hosts Set maintains list of crawled host and host related statistics.

- **HostID:** unique identifier of the host
- **HostName:** name of the host
- **SuccessCount:** success counter for document inside the host
- **ErrorCount:** error counter for documents inside the host
- **WarningCount:** warning counter for documents inside the host
- **DeleteCount:** delete counter for documents inside the host
- **LevelHighErrorCount:** error counter for documents of high importance, as specified in section 2.2.1.10, inside the host

3.1.1.8 Anchor Text Info

Anchor Text Info Set: This data structure contains all the links discovered during the **crawl**. Each link has the following attributes:

- **SourceDocID:** source item identifier
- **TargetDocID:** target item identifier

- **Link:** DisplayURL of the Crawl History Set as specified in section 3.1.1.2.
- **LinkID:** unique identifier of the link
- **CrawlID:** identifier of the crawl which is going to process the link
- **InterSite:** 0 if Source Document and Target Document located within the same site, 1 otherwise
- **SourceDocSiteID:** identifier of the site where the source document is located
- **LinkHash:** signature of the link. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same Link.
- **AnchorText:** anchor text
- **AnchorHash:** signature of anchor text. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same AnchorText.
- **LinkOrdinal:** ordinal of the link for the given source of the link
- **LCID:** LCID of the item content
- **Pid:** managed property the link is applied to, the specific values are implementation details of the protocol client

3.1.1.9 Pending Annotations

Pending Annotations Set: This data structure contains annotations discovered during the crawl that will be processed during the anchor crawl.

- **TargetDisplayURL:** URL of the target item (similar to DisplayURL in Crawl History Set (section 3.1.1.2))
- **TargetDisplayHash:** signature of TargetDisplayURL. The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same TargetDisplayURL.
- **TargetDocID:** target item identifier
- **AnnotationText:** text of the annotation
- **AnnotationNumeric:** if AnnotationText is not NULL AnnotationNumeric shows number of times this text occurred otherwise it is number associated with annotation
- **Pid:** annotation type. It MUST be one of the values listed in the following table:

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

- **LCID:** Language Code Identifier of the annotation text.

- **UpdateTime:** UTC time when the link was last updated

3.1.1.10 Pending Annotations Status

Pending Annotations Status: This data structure contains information about the Pending Annotations Set.

- **ObsoleteTime:** UTC time which states which annotations in the Annotations set (see section 3.1.1.37) are considered obsolete. The records of Annotation set are considered obsolete if UpdateTime is smaller than ObsoleteTime.
- **LastUpdateTime:** UTC time which is the last time when one of the records in the Annotations set was updated. It is also the most recent value of the UpdateTime in Annotations Set.
- **Status:** status of pending annotations. It MUST be one of the values listed in the following table:

Value	Description
1	Import of annotations is done
2	Full import of annotations is done
3	Pending annotations are processed

- **LastFullUpdateTime:** UTC time which is the last time when all of the records in the Annotations set were updated.

3.1.1.11 Social Distance Property

Social Distance Property Set: This data structure contains information about user relationships that were found during previous crawls. Each record has the following attributes:

- **SourceDocID:** document identifier (1) specifying the source user
- **SourceUserID:** user profile record identifier of the source user
- **TargetDocID:** document identifier (1) specifying the target user
- **TargetUserID:** user profile record identifier of the target user
- **Pid:** identifier if the relationship is public or private. Pid = 394 means public relationship; Pid=395 means private relationship

3.1.1.12 Social Distance Deleted Property

Social Distance Deleted Property set: Social Distance Deleted Property set maintains the list of outgoing links which were deleted and which MUST be deleted from Social Distance Property set, as specified in section 3.1.1.11.

- **CrawlID:** identifier of the crawl which processed (deleted) this link
- **SourceDocID:** document identifier (1) specifying the source user
- **TargetUserID:** user profile record identifier of the target user

3.1.1.13 Social Distance New Property

Social Distance New Property Set: Social Distance New Property set maintains the list of outgoing links which were changed after the last anchor crawl and which MUST be updated in Social Distance Property set, as specified in section 3.1.1.11.

- **CrawlID:** identifier of the crawl with processed this link
- **SourceDocID:** document identifier (1) specifying the source user
- **TargetUserID:** user profile record identifier of the target user

3.1.1.14 Anchor Change

Anchor Change Set: The Anchor Change Set maintains list of document identifiers which MUST be processed in the upcoming or the ongoing anchor crawl.

- **CrawlID:** The unique identifier of the crawl which included the change related to this item.
- **TargetDocID:** The unique identifier of an item which needs to be processed by the anchor crawl.
- **ChangeType:** as specified in section 2.2.1.23

3.1.1.15 Pending Anchor Change

Pending Anchor Change Set: The Pending Anchor Change Set maintains list of document identifiers denoting items which MUST be processed by the anchor crawl.

- **CrawlID:** The unique identifier of the crawl which included the change related to this item.
- **TargetDocID:** The unique identifier of an item which needs to be processed by the anchor crawl.

3.1.1.16 Reported Crawl Error

Reported Crawl Error Set: The reported Crawl Error Set maintains list of errors which occurred during crawling.

- **DocID:** identifier of the document which experienced the problem
- **CrawlID:** identifier of the crawl which reported the error
- **ErrorID:** error identifier, from Crawl Error Set ([MS-SQLPADM2] section 3.1.1.3).
- **ChildrenCount:** number of child objects of the document with identifier DocID

3.1.1.17 Current DocID Groups

Current DocID Groups Set: Current DocID Chunks Set maintains the list of all created intervals of document identifiers:

- **FirstDocID:** First document of the group.
- **LastDocID:** Last document of the group.
- **DBID:** Identifier of the crawl store in the Crawl Store Set, as specified in [MS-SRCHTP] section 3.1.1.3, where documents with identifiers with the current group are located.

- **InvalidatedByRefactoring:** Identifies the availability of the group, "0" if the group is valid, "1" if the group was invalidated by previous operations.

3.1.1.18 Current DocID Chunk

Current DocID Chunk: Current DocID Chunk maintains the interval of available document identifiers which will be used during processing of items:

- **FirstDocID:** first document identifier of the interval
- **LastDocID:** last document identifier of the interval

3.1.1.19 Requested Delete Crawls

Requested Delete Crawls set: The Requested Delete Crawls set maintains the list of items, such as Content Sources and Start Addresses, for which the delete crawl was requested:

- **CrawlID:** identifier of the crawl, which is going to delete requested documents.
- **ContentSourceID:** identifier of the content source for which documents MUST be deleted from the index.
- **StartAddressID:** identifier of the start address for which documents MUST be deleted, if start address delete is requested, 0 if entire content source delete is requested.

3.1.1.20 Changed Anchor Target Documents

Changed Anchor Target Documents set: Changed Anchor Target Documents set maintains list of documents identifiers of the documents which pointed by a link from a document which was changed during the crawl.

- **CrawlID:** identifier of the crawl which discovered modified links
- **DocID:** identifier of the document pointed by a link from a document that was changed during the crawl. All target documents from changed documents MUST be re-crawled during the anchor crawl.

3.1.1.21 Changed Anchor Source Documents

Changed Anchor Source Documents set: Changed Anchor Source Documents set maintains list of documents that have links that were discovered during the crawl.

- **CrawlID:** Identifier of the crawl which discovered modified links.
- **DocID:** Identifier of the document which was changed.

3.1.1.22 Changed Anchor Committed Documents

Changed Anchor Committed Documents set: Changed Anchor Committed Documents set maintains list of documents which were committed during the crawl.

- **CrawlID:** identifier of the crawl which discovered modified links
- **DocID:** identifier of the document which was changed

3.1.1.23 Changed Anchor Deleted Documents

Changed Anchor Deleted Documents set: Changed Anchor Deleted Documents set maintains list of documents which were deleted during the crawl.

- **CrawlID:** identifier of the crawl which discovered modified links
- **DocID:** identifier of the document which was deleted

3.1.1.24 UserID DocID Pair

UserID DocID Pair Set maintains user and associated document

- **UserID:** user profile record identifier
- **DocID:** identifier of document

3.1.1.25 Crawl Component Status

Crawl Component Status Set: The Crawl Component Status Set maintains the list of crawl components which participate in active crawls.

- **ComponentID**: identifier of crawl component in Crawl Component Set ([MS-SRCHTP] section 3.1.1.3)
- **CrawlID**: crawl unique identifier
- **Status**: as specified in section 2.2.1.4
- **Activity Info**: data structure that contains information about the component activity, the format of this structure is implementation specific.
- **SuccessCount**: Count of items successfully crawled by the crawl component.
- **WarningCount**: Count of items which generated warnings while crawled by the crawl component.
- **ErrorCount**: Count of items which generated errors while crawled by the crawl component.
- **DeleteCount**: Count of items deleted by the crawl component.
- **NotModifiedCount**: Count of items for which the crawl detected no change after the previous processing by this crawl component.
- **SecurityOnlyErrorCount**: count of items which generated errors while performing a security only crawl (crawl which is performed to update security information about the items) by this crawl component.
- **RetryCount**: number of times the crawl component retried the processing of an item.

3.1.1.26 Local Crawl Components

Local Crawl Component Set: maintains list of crawl components from Crawl Component Set ([MS-SRCHTP] section 3.1.1.3) located in the given crawl database

- **ComponentID**: unique identifier of the crawl component
- **Status**: as specified in section 3.1.1.25

3.1.1.27 Local Completed Crawls

Local Completed Crawl Set: Local Completed crawl Set maintains list of crawls finished in the given crawl store

- **CrawlID**: unique identifier of crawl
- **Time**: UTC when the crawl was done

3.1.1.28 Crawl Store Status

Crawl Store Status Set contains information about which crawl store participating in which crawl

- **GthrDBID**: identifier of the crawl store in Crawl Stores Set ([MS-SRCHTP] section 3.1.1.3)
- **CrawlID**: crawl unique identifier, in which the crawl store is participating
- **JoinTime**: UTC when crawl store joined the crawl

3.1.1.29 Scopes to Compile

The **Scope to Compile Set** contains all the search scopes that are involved in a current search scope compilation. For the specification of the search Scope Set see [MS-SQLPADM2] Section 3.1.1.4. Each item in the Scopes to Compile Set MUST include the following elements:

- **ScopeID**: The unique identifier of the search scope.
- **ConsumerID**: The identifier to the corresponding **search scope consumer**.
- **Name**: The name for the search scope.
- **CompilationState**: The state of compilation of the search scope. The value MUST be a CompilationState as specified in [MS-SQLPADM2] Section 2.2.1.4.

3.1.1.30 Scope Rules to Compile

The **Scope Rules to Compile Set** contains all the search scope rules that are involved in a current search scope compilation. For the specification of the search Scope Rule Set see [MS-SQLPADM2] Section 3.1.1.4. Each item in the Scope Rules to Compile Set MUST include the following elements:

- **ScopeID**: The unique identifier of the search scope.
- **FilterBehavior**: The value MUST be a ScopeFilterBehavior data type as specified in [MS-SQLPADM2] section 2.2.1.18.
- **RuleType**: The type of search scope rule which MUST be a ScopeRuleType data as specified in [MS-SQLPADM2] section 2.2.1.19.
- **UrlRuleType**: The part of the item URL that is matched against the value in the UserValueString attribute. The value MUST be a UrlRuleType as specified in [MS-SQLPADM2] section 2.2.1.21.
- **PropertyID**: The property identifier, which is a reference to a managed property as defined in Section 2.2.1.13.
- **UserValueString**: The search scope rule value.

3.1.1.31 Deleted Scopes

The **Deleted Scopes Set** contains all the search scopes that are to be deleted. For the specification of the search Scope Set see [MS-SQLPADM2] Section 3.1.1.4. Each item in the Deleted Scopes Set MUST include the following elements:

- **ScopeID**: The unique identifier of the search scope.
- **ChangeID**: The version of the search scope that corresponds to its last deletion, which is a LastChangeID in Scopes Set.

3.1.1.32 Deleted Scopes to Compile

The **Deleted Scopes to Compile Set** contains all the search scopes to be deleted that are involved in a current search scope compilation. For the specification of the search Scope Set see [MS-SQLPADM2] Section 3.1.1.4. Each item in the Deleted Scopes to Compile Set MUST include the following elements:

- **ScopeID**: The unique identifier of the search scope.

3.1.1.33 Compiled Scopes

The **Compiled Scopes Set** contains all the search scopes that have been compiled. For the specification of the search Scope Set see [MS-SQLPADM2] Section 3.1.1.4. Each item in the Compiled Scopes Set MUST include the following elements:

- **ScopeID**: The unique identifier of the search scope.
- **ConsumerID**: The reference to the corresponding search scope consumer.
- **Name**: The name for the search scope
- **CompilationState**: The state of compilation of the search scope. The value MUST be a CompilationState as specified in [MS-SQLPADM2] Section 2.2.1.4.

3.1.1.34 Compiled Scope Rules

The **Compiled Scope Rules Set** contains all the search scope rules that have been compiled. For the specification of the search Scope Rule Set see [MS-SQLPADM2] Section 3.1.1.4. Each item in the Compiled Scope Rules Set MUST include the following elements:

- **ScopeID**: The unique identifier of the search scope.
- **FilterBehavior**: The value MUST be a ScopeFilterBehavior data type as specified in [MS-SQLPADM2] section 2.2.1.18.
- **RuleType**: The type of search scope rule which MUST be a ScopeRuleType data as specified in [MS-SQLPADM2] section 2.2.1.19.
- **UrlRuleType**: The part of the item URL that is matched against the value in the UserValueString attribute. The value MUST be a UrlRuleType as specified in [MS-SQLPADM2] section 2.2.1.21.
- **PropertyID**: The property identifier, which is a reference to a managed property as defined in Section 2.2.1.13.
- **UserValueString**: The search scope rule value.

3.1.1.35 Component Activity

The **Components Activity Set** contains one record for each project of each component of the search service application. The information stored in this set reflects the current state and the ongoing state transitions in regard to the crawling activity and the resetting of the crawled data.

- **ComponentName**: name of the component
- **ProjectName**: name of the project
- **HaveResetRequest**: MUST be 1 if component reset has been requested 0 otherwise
- **DPR_MasterMerge**: desired reasons as specified in section 2.2.2.1 to pause master merge
- **DPR_Crawls**: desired reasons as specified in section 2.2.2.1 to pause crawls
- **CPR_MasterMerge**: current reasons as specified in section 2.2.2.1 why master merge is paused
- **CPR_Crawls**: current reasons as specified in section 2.2.2.1 why craws are paused
- **ComponentType**: type of the component, MUST be 0 for Admin Component, 1 for Query Component, 2 for Crawl Component and 3 for Standalone install
- **ComponentID**: identifier of the component

The records stored in this set MUST observe the following restrictions:

- The values of the DPR_MasterMerge MUST be the same for all records with the same ProjectName.
- The values of the DPR_Crawl MUST be the same for all records with the same ProjectName.

3.1.1.36 Configuration Properties

The management of the **configuration properties** uses 3 data sets:

Configuration Properties set:

The Configuration Properties set is a list of properties used for storing settings for the search service application. Each item in this list MUST include the following elements:

- **Name**: a name of the property as a **Unicode** string that MUST be 300 characters in length or less.
- **Value**: a value of the property.

Each configuration name MUST correspond to a single value. A new value set for an existing property MUST overwrite the previous value.

Uncommitted Transactions of Configuration Properties set:

Each ongoing transactional change on the Configuration Properties set is tracked by a record in the Uncommitted Transactions of Configuration Properties set.

- **TransactionID**: a unique identifier of the transaction

Configuration Properties Transactions set:

The Configuration Properties Transactions set contains the ongoing changes to the Configuration Properties set. One transactional change is defined by a subset records of the Configuration Properties Transactions set which have the same TransactionID.

- **Name**: a name of the property
- **Value**: a value of the property
- **TransactionID**: A unique identifier of the transaction. The value of this field has to be present in Uncommitted Transactions of Configuration Properties set.

3.1.1.37 Annotations

Annotations set: Holds text that annotates a document.

- **TargetDocId**: the document identifier (1) of the document that this item applies to.
- **Pid**: a **property identifier**.
- **AnnotationText:** a Unicode string that MUST be 1024 characters in length or less.
- **UpdateTime:** the date and time at which this item was created or modified, in UTC.
- **LCID:** LCID of the AnnotationText field
- **AnnotationNumeric:** if AnnotationText is not NULL AnnotationNumeric shows number of times this text occurred otherwise it is number associated with annotation

3.1.1.38 Anchor Document Properties Blob

Holds a BLOB that tracks properties of a document.

- **DocId**: the document identifier (1) of the document that this item applies to.
- **PropertyTrackerBlob**: binary large object (BLOB) holding property tracking information. The format of the BLOB is an implementation detail of the protocol client

3.1.1.39 Search Component Name

The name of the **search component**. The syntax of the string is specified as follows in Augmented BNF notation [RFC5234] (assuming that the term GUID and Number are implicitly defined):

ComponentIdentifier = Number

SearchComponentName = GUID [- ('crawl' | 'query') - ComponentIdentifier]

3.1.1.40 Definitions

Definitions set: Holds the terms and their definitions that are found in a document.

- **DocId**: The document identifier (1) of the document where the term was found.
- **Term**: term found in the document identified by the DocId.
- **Sentence**: definition sentence that defines the term
- **Term Offset**: MUST be the offset of the term within the sentence.

- **IsAcronym**: MUST be 1 if the term is an acronym and MUST be 0 if the term is not an acronym.

3.1.1.41 Commands

Commands Set: A list of commands to be processed by an object that exists on another physical server.

- **CommandID**: A number that uniquely identifies the command.
- **ObjectName**: A name that uniquely identifies the object that will process the command.
- **Description**: A string which will be interpreted and acted upon by the object identified by ObjectName.
- **Completed**: A Boolean value that indicates whether or not the command has already been processed.

3.1.1.42 Metadata Store Crawl Components

Metadata Store Crawl Components Set: maintains list of crawl components that access the metadata store.

- **ComponentID**: unique identifier of the crawl component.
- **Status**: The state of the crawl component indicated by ComponentID as defined in section 2.2.1.12.
- **ModifiedByComponentID**: unique identifier of the crawl component that last changed the status.

3.1.1.43 Social Distance Unchanged Property

Social Distance Unchanged Property Set: Social Distance Unchanged Property set maintains the list of outgoing links that were not changed after the last anchor crawl.

- **CrawlID:** Identifier of the crawl with processed this link.
- **SourceDocID:** document identifier (1) specifying the source user.
- **TargetUserID:** user profile record identifier of the target user.

3.1.2 Timers

None.

3.1.3 Initialization

None.

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

3.1.5.1 proc_MSS_AddAndReturnCrawledProperty

The **proc_MSS_AddAndReturnCrawledProperty** stored procedure is called to add a **crawled property** to the **metadata schema** if and only if it doesn't exist, and to return the crawled property**.** If @FullTextQueriable is set to 1, or @Retrievable is set to 1, or @Scoped is set to 1, then a managed property MUST also be created if and only if it doesn't already exist, and a mapping between the crawled property and the managed property MUST be created if and only if it doesn't already exist in the metadata schema mapping set as defined in [MS-SQLPADM2] Section 3.1.1.1.

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

```
PROCEDURE proc_MSS_AddAndReturnCrawledProperty(
    @Propset                        uniqueidentifier,
    @PropertyName                   nvarchar (440),
    @PropertyNameIsEnum             bit,
    @VariantType                    int,
    @FriendlyName                   nvarchar (64),
    @ManagedType                    int,
    @FullTextQueriable              bit,
    @Retrievable                    bit,
    @Scoped                         bit,
    @MaxIndexedStringLength         int,
    @MaxNonIndexedStringLength      int,
    @crawledPropId                  int OUTPUT,
    @pid                            int OUTPUT,
    @MappingOrder                   int OUTPUT,
    @IsMappedToContent              bit OUTPUT,
    @IsSampleCacheFull              bit OUTPUT,
    @crawledPropExists              bit OUTPUT,
    @managedPropExists              bit OUTPUT,
    @ManagedPropDecimalPlaces       tinyInt
);
```

@Propset: The **crawled property set identifier** for this property.

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

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

@VariantType: The **variant type (2)** for the crawled property**.**

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

@ManagedType: The type of the managed property as defined in section 2.2.1.13**.**

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

@Retrievable: This parameter MUST be set to 1 if the data for the managed property is kept in the metadata index. Otherwise, it MUST be set to 0.

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

@MaxIndexedStringLength: The maximum number of characters persisted in the string value column in the *MSSDocProps* table defined in [MS-SQLPQ2] section 2.2.5.2. If the string value length is greater than *MaxIndexedStringLength*, the string is truncated from the end

@MaxNonIndexedStringLength: If the string value length in the *MSSDocProps* table, defined in [MS-SQLPQ2] section 2.2.5.2, is greater than the size allowed by the *MaxIndexedStringLength*, the string is truncated from the end and the string overflow is stored in the strVal2 column, in the *MSSDocProps* table defined in [MS-SQLPQ2] section 2.2.5.2, up to the size allowed by the *MaxNonIndexedStringLength*.

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

@pid: Upon return from this stored procedure, this parameter MUST be set to the unique identifier of the managed property, if a managed property was added or a managed property already existed for the crawled property, otherwise this parameter MUST be set to -1.

@MappingOrder: Upon return from this stored procedure, this parameter MUST be set the **mapping order** of the crawled property to the managed property, if such a mapping exists.

@IsMappedToContent: If a crawled property is being added, then this parameter MUST be set to 1 if data from this crawled property is kept in the full-text index catalog as property identifier 1 which is the contents property, otherwise IsMappedToContent MUST be set to 0.

@IsSampleCacheFull: Upon return from this stored procedure, this parameter MUST be set to 1 if the Sample Crawled Properties Set as defined in [MS-SQLPADM2] section 3.1.1.1 is complete. Otherwise, it MUST be set to 0.

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

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

@ManagedPropDecimalPlaces: The number of floating point decimal places for the managed property in the metadata schema, if a managed property is added.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the Managed Properties Result Set with exactly one row for the managed property that corresponds to the specified crawled property, if a managed property exists or was added. For the specification of the Managed Properties Result Set, see [MS-SQLPQ2] Section 3.1.5.3.1.

3.1.5.2 proc_MSS_AddCrawledPropertyCategoryWithDefaults

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

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

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

```
    @MapToContents                  bit OUTPUT,
    @FullTextQueriable              bit OUTPUT,
    @Retrievable                    bit OUTPUT,
    @Propset                        uniqueidentifier
);
```

@CategoryName: Upon return from this stored procedure, this parameter MUST be set to the name of the crawled property category added.

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

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

@FullTextQueriable: Upon return from this stored procedure, this parameter MUST be set to 0.

@Retrievable: Upon return from this stored procedure, this parameter MUST be set to 0.

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

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

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

3.1.5.3 proc_MSS_CheckIfStatusChangeComplete

The **proc_MSS_CheckIfStatusChangeComplete** is called to determine whether all active components in the Component Activity Set (specified in Section 3.1.1.35) have completed the specified status change. The stored procedure MUST ignore records from Component Activity Set which correspond to crawl components from Crawl Components set with states equal Disabled and query components from Query Components set with states equal Offline.

If the specified status change is a reset, the stored procedure MUST determine that the status change is complete if the **HaveResetRequest** flag of all active components in the Component Activity Set is not equal to 1. Otherwise the status change is not complete.

If the specified status change is a pause, the stored procedure MUST determine that it is complete if the following conditions are satisfied for all active components in the Component Activity Set:

- The current (**CPR_Crawls** or **CPR_MasterMerge**) as specified in section 2.2.2.1 and desired (**DPR_Crawls** or **DPR_MasterMerge**) as specified in section 2.2.2.1 pause reasons for the specified level are equal.
- The desired (**DPR_Crawls** or **DPR_MasterMerge**) pause reason is not equal to 0.

Otherwise the status change is not complete.

If the specified status change is a resume, the stored procedure MUST determine that it is complete if the following conditions are satisfied for all active components in the Component Activity Set:

- The current (**CPR_Crawls** or **CPR_MasterMerge**) and desired (**DPR_Crawls** or **DPR_MasterMerge**) pause reasons for the specified level are equal.
- The desired (**DPR_Crawls** or **DPR_MasterMerge**) pause reason does not have the flag in @Reason set.

Otherwise the status change is not complete.

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

```
PROCEDURE proc_MSS_CheckIfStatusChangeComplete (
    @CheckIfResetDone        int,
    @CheckIfPauseDone        int,
    @CheckIfResumeDone       int,
    @Level                   int,
    @Reason                  int
);
```

@CheckIfResetDone: If the status change to be verified is a reset, then this parameter MUST be set to 1, otherwise it MUST be set to 0. When this parameter is 1, this stored procedure MUST ignore all the other parameters.

@CheckIfPauseDone: If the status change to be verified is a pause, then this parameter MUST be set to 1, otherwise it MUST be set to 0. If this parameter is set to 1, then @CheckIfResumeDone MUST be ignored.

@CheckIfResumeDone: If the status change to be verified is a pause, then this parameter MUST be set to 1, otherwise it MUST be set to 0.

@Level: If the pause or resume status of master merge needs to be verified then this parameter MUST be set to 1. If the pause or resume status of crawls needs to be verified then this parameter MUST be set to 2.

@Reason: If the status change to be verified is a resume, then this parameter MUST be a value of type pause reason as specified in Section 2.2.2.1.

Return Code Values: This stored procedure returns an integer return code which MUST be one of the values listed in the following table.

Value	Description
0	The specified status change is not complete.
1	The specified status change is complete.

Result Sets: The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.5.4 proc_MSS_IsSystemPausedForRefactoring

The **proc_MSS_IsSystemPausedForRefactoring** is called to determine whether all active components in the Component Activity Set (specified in Section 3.1.1.35) have completed the pause. The stored procedure MUST ignore records from Component Activity Set which correspond to crawl components from Crawl Components set with states equal to Disabled and query components from Query Components set with states equal to Offline.

The stored procedure MUST determine that pause for specified reason is complete if the following conditions are satisfied for all active components in the Component Activity Set:

- The current **CPR_Crawls** as specified in section 2.2.2.1 and desired **DPR_Crawls** as specified in section 2.2.2.1 pause reasons for the specified level are equal.
- The desired **DPR_Crawls** pause reason has a flag specified by @Reason set.

Otherwise the system is not paused for specified reason

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

```
PROCEDURE proc_MSS_CheckIfStatusChangeComplete (
@Reason                 int
);
```

@Reason: value of type pause reason as specified in Section 2.2.2.1.

Return Code Values: This stored procedure returns an integer return code which MUST be one of the values listed in the following table.

Value	Description
0	The system is not paused for the specified reason
1	The system is paused for the specified reason

Result Sets: The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.5.5 proc_MSS_CommitConfigTransaction

Search components use this stored procedure to update a group of configuration properties within a single T-SQL transaction.

This stored procedure MUST update Configuration Properties (section 3.1.1.36) with the name, value pairs from Configuration Properties Transactions (section 3.1.1.36) that have a TransactionID equal to the value passed in as @TransactionID.

The stored procedure MUST remove all items with TransactionID equal to the value passed in from Configuration Properties Transactions (section 3.1.1.36).

The stored procedure MUST remove all items with TransactionID equal to the value passed in from Uncommitted Transactions of Configuration Properties set (section 3.1.1.36).

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

```
PROCEDURE proc_MSS_CommitConfigTransaction(
    @TransactionID          int
);
```

@TransactionID: Unique identifier of the transaction, the transaction MUST be from Uncommitted Transactions of Configuration Properties set (section 3.1.1.36)

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

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

3.1.5.6 proc_MSS_CompleteStatusChange

This stored procedure modifies the Component Activity Set (section 3.1.1.35) to indicate a status change has completed. It MUST set HaveResetRequest to 0, CPR_MasterMerge to DPR_MasterMerge

and CPR_Crawls to DPR_Crawls in the record with ComponentName equal to @ComponentName and ProjectName equal to @ProjectName.

If @RemoveRecord is equal to 1, the stored procedure MUST remove the record from Component Activity Set (section 3.1.1.35) with ComponentName equal to @ComponentName and ProjectName equal to @ProjectName.

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

```
PROCEDURE proc_MSS_CompleteStatusChange (
    @ComponentName          nvarchar(100),
    @ProjectName            nvarchar(100),
    @RemoveRecord           int
);
```

@ComponentName: Name of the crawl component or **query component (2)**.

@ProjectName: Name of the project. See section 2.2.1.19.

@RemoveRecord: Input parameter specifying that the corresponding record should be removed from Component Activity Set (section 3.1.1.35).

Return Code Values: this stored procedure returns an integer that MUST be ignored.

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

3.1.5.7 proc_MSS_Crawl

The **proc_MSS_Crawl** stored procedure is called by a crawl component to update the crawl in the associated crawl store.

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

```
PROCEDURE proc_MSS_Crawl(
    @ComponentID            int,
    @ProjectID              int,
    @CrawlStage             int,
    @CrawlType              int,
    @CrawlID                int,
    @ContentSourceID        int,
    @ApplicationType        int,
    @MiscInputData          int,
    @MiscOutputData         int OUTPUT,
    @CrawlStatus            int OUTPUT,
    @CrawlSubStatus         int OUTPUT
);
```

@ComponentID: The unique identifier of the crawl component.

@ProjectID: The identifier of the project. The value of this parameter MUST be a Project Identifier data type as specified in Section 2.2.1.1.

@CrawlStage: The action that the stored procedure executes. The value of this parameter MUST be one of the values specified in the following table.

@CrawlType: The type of the crawl. The value of this parameter MUST be a Crawl Type data type as specified in Section 2.2.1.2.

@CrawlID: The unique identifier of the crawl.

@ContentSourceID: The unique identifier of the content source associated with the crawl.

@ApplicationType: The value of this parameter MUST be 0.

@MiscInputData: Input data. The value of this parameter depends on the value of @CrawlStage, as specified in the following table.

@MiscOutputData: Upon return from the stored procedure, this value of this parameter MUST be set to @CrawlID, unless otherwise specified in the following table.

@CrawlStatus: Upon return from the stored procedure, this parameter MUST be set to 0, unless otherwise specified in the following table.

@CrawlSubStatus: Upon return from the stored procedure, this parameter MUST be set to 0.

The following table specifies how the input parameters are used, for each possible value of @CrawlStage.

Value of @CrawlStage	Description
93	Input: @ComponentID MUST be set to the unique identifier of crawl component. Other input parameters MUST be ignored. Actions: The stored procedure notifies the crawl store that the specified crawl component belongs to the crawl store. The stored procedure MUST create a new record in the Local Crawl Components set (specified in Section 3.1.1.26) with Component identifier set to input parameter @ComponentID and Status set to 1 (OK). For the specification of Crawl Component State see Section 2.2.1.12.
90	Input: @ComponentID MUST be set to the unique identifier of crawl component. @MiscInputData MUST be set to 0 if the Crawl Queue (specified in Section 3.1.1.4) needs to be recovered, otherwise it MUST be set to 1. Other input parameters MUST be ignored. Actions: The stored procedure recovers the specified crawl component. If the specified crawl component does not exist in the Local Crawl Components set (specified in Section 3.1.1.26), the stored procedure MUST create a new record with Component identifier set to input parameter @ComponentID and Status set to 1 (OK). Otherwise, the specified crawl component already exists in the Local Crawl Components Set. In this case the stored procedure MUST set the Status to 1 (OK) if and only if the existing Status is 3 (Disabled). For the specification of Crawl Component State see Section 2.2.1.12. If @MiscInputData is 0, the stored procedure MUST set value of BatchID in the Crawl Queue set (specified in Section 3.1.1.4) to 0 for all documents in the Crawl URL History set (specified in Section 3.1.1.2) which were being crawled by the specified crawl component. The stored procedure MUST increment the counter Retry of the Crawl URL History Set by 1 for all documents that were updated.
107	Input:

Value of @CrawlStage	Description
	@CrawlID MUST be set to the unique identifier of crawl. @ContentSourceID MUST be set to the unique identifier of content source associated with the crawl. @CrawlType MUST be set to the type of crawl and MUST be a Crawl Type data type as specified in Section 2.2.1.2. @MiscInputData MUST be set to 0 if the entire content source is to be deleted, otherwise it MUST be set to the unique identifier of the start address to be deleted. Other input parameters MUST be ignored. Actions: The stored procedure updates the Crawl Queue set (specified in Section 3.1.1.4) and Crawl URL History set (specified in Section 3.1.1.2) for a delete crawl. If @CrawlType is set to 3 (Delete Crawl), a new record MUST be added to the Crawl Queue Set for every record in the Crawl URL History Set with ▪ ContentSourceID equal to @ContentSourceID. ▪ StartAddressID equal to @MiscInputData if @MiscInputData is not equal to 0. Otherwise the Crawl URL History Set MUST have the StartAddressID set to the value from the Crawl Queue Set. ▪ DeletePending equal to 0. Each new record in the Crawl Queue Set MUST have the following values: ▪ CrawlID set to input parameter @CrawlID. ▪ If @MiscInputData was passed in as a 0 then StartAddressID in the Crawl Queue MUST be set to value stored in the Crawl Queue Set. ▪ DocID, TransactionFlags, HostDepth, EnumerationDepth, ContentSourceID, ProjectID equal to the corresponding values of the record from the Crawl URL History record. ▪ TransactionType set to 1 (Delete) ▪ Scope set to 1 ▪ SourceDocID set to 0 ▪ ChangeLogBatchID set to 0 ▪ DeleteReason set to 2 (Delete Crawl) For every record in the Crawl URL History Set for which a record was added to the Crawl Queue Set, the stored procedure MUST set DeletePending to 1, which sets a delete request for the record. The following is executed for all values for of @CrawlType. The stored procedure MUST set @MiscOutputData to the count of records in Crawl URL History set with the value of the field CrawlID equal to @CrawlID.
109	Input: @CrawlID MUST be set to the unique identifier of crawl. @ContentSourceID MUST be set to the unique identifier of content source associated with the crawl. Other input parameters MUST be ignored. Actions: The stored procedure updates the Crawl Queue set (specified in Section

Value of @CrawlStage	Description

	3.1.1.4) and Crawl URL History set (specified in Section 3.1.1.2) at the beginning of an incremental crawl.
	This action may be called multiple times for the same @CrawlID, the stored procedure MUST not do anything in all the calls but the first one.
	The stored procedure MUST add a record to the Crawl Queue set for every record in the Crawl URL History set which satisfies all of the following conditions:
	▪ ContentSourceID equal to @ContentSourceID
	▪ DeletePending equal to 0 (there are no delete requests)
	▪ Any one of the following conditions is met:
	  ▪ ErrorLevel is 2 (the record was previously crawled with an error)
	  ▪ ChangeLogCookieType is not null (it was previously crawled and there is a cookie)
	  ▪ UseChangeLog equal to 0 (does not belong to a site that supports incremental crawling based on a change log) and either TransactionFlags as specified in section 2.2.2.3
	Each new record added to the Crawl Queue Set MUST have values:
	▪ StartAddressID, DocID, HostDepth, EnumerationDepth, ContentSourceID, ProjectID are set to the corresponding values of the Crawl URL History record.
	▪ TransactionType set to 2 (Modify Item).
	▪ Scope set to the value of Scope in the Crawl URL History record.
	▪ TransactionFlag set to the value of TransactionFlag of the Crawl URL History record along with the bit 0x00000004.
	▪ SourceDocID set to the ParentDocID of the Crawl URL History record.
	▪ ChangeLogBatchID set to 0.
	The stored procedure MUST set value of CrawlID to the input parameter @CrawlID for all records from the Crawl URL History for which a Crawl Queue Set record was added.
	The stored procedure MUST add a record to the Crawl Queue Set for every record in the Crawl URL History Set which satisfies all of the following conditions:
	▪ ContentSourceID equal to @ContentSourceID
	▪ DeletePending equal to 0 (there are no delete requests)
	▪ SecurityUpdateErrorID is not equal to 0
	Each new record added to the Crawl Queue Set MUST have values:
	▪ StartAddressID, DocID, HostDepth, EnumerationDepth, ContentSourceID, ProjectID are set to the corresponding values of the Crawl URL History record.
	▪ TransactionType set to 2 (Modify Item).
	▪ Scope set to 2

Value of @CrawlStage	Description
	▪ TransactionFlag set to the value of TransactionFlag of the Crawl URL History record additionally having the bit 0x2000000 set. ▪ SourceDocID set to the ParentDocID of the Crawl URL History record. ▪ ChangeLogBatchID set to 0. The stored procedure MUST set value of SecurityUpdateCrawlID to the input parameter @CrawlID for all records from the Crawl URL History for which a Crawl Queue Set record was added. The stored procedure MUST set @MiscOutputData to 1 if there any records in the Crawl Queue Set with the value of CrawlID equal to @CrawlID

170	This value of @CrawlStage is only allowed if the server is running Microsoft® SharePoint® 2010 Products and Technologies role
	Input:
	@MiscInputData MUST be set to the unique identifier of the crawl.
	@CrawlType MUST be set to the type of crawl and MUST be a Crawl Type data type as specified in Section 2.2.1.2.
	Other input parameters MUST be ignored.
	Actions: The stored procedure MUST perform the following sequence of actions for this crawl stage which indicates the start of an anchor crawl.
	For every record of the Pending Annotations set, as specified in Section 3.1.1.9, for which there is an entry in the Crawl URL History set, specified in Section 3.1.1.2, in which the:
	▪ TargetDisplayHash of the Pending Annotations record is equal to the DisplayHash of the Crawl URL History record.
	▪ TargetDisplayURL of the Pending Annotations Set record is equal to the DisplayURL of the Crawl URL History record.
	▪ IndexType of the Crawl URL History record is equal to "1".
	▪ DeletePending of the Crawl URL History record is "0", which means that there is no request to delete.
	The stored procedure MUST do the following:
	▪ Delete all records from the Annotations Set which have TargetDocID equal to TargetDocID of the record from Annotation Pending set which satisfied the following conditions:
	  ▪ There exists a record in the Crawl URL History Set with the DisplayHash and DisplayURL equal to the TargetDisplayHash and TargetDisplayURL of the record from the Annotation Pending set.
	  ▪ The record from the Crawl URL History set has an IndexType value equal to "1" and DeletePending equal to "0".
	▪ Insert a record into the Annotations Set with values
	  ▪ Pid, LCID, AnnotationText, AnnotationNumeric, UpdateTime set to the corresponding values of the record in the Pending Annotation Set.
	  ▪ TargetDocID set to DocID of the Crawl URL history record.

Value of @CrawlStage	Description

	▪ If such a record doesn't already exist, insert a record into the Pending Anchor Change set (specified in section 3.1.1.15) with values
	  ▪ CrawlID set to @MiscInputData.
	  ▪ TargetDocID set to TargetDocID of the Crawl URL History record.
	The stored procedure MUST remove all records from Pending Annotations set.
	If @CrawlType is equal to "1" (Full Crawl) and @CrawlID is equal to "0", the stored procedure MUST add a record to the Anchor Change set, as specified in Section 3.1.1.14, for every record in the Pending Anchor Change Set in which:
	▪ CrawlID is equal to @MiscInputData.
	▪ There are no records in the Anchor Text Info set, as specified in Section 3.1.1.8, with TargetDocId equal to the TargetDocId of the Pending Anchor Change record.
	▪ There are no entries in the Annotations set, as specified in section 3.1.1.37, with TargetDocId equal to the TargetDocId of the Pending Anchor Change record.
	▪ There are no entries in the Social Distance Property set, as specified in Section 3.1.1.11, with TargetDocId equal to the TargetDocId of the Pending Anchor Change record.
	Each new record added to the Anchor Change Set MUST have the following values:
	▪ CrawlID set to @MiscInputData.
	▪ TargetDocID set to the TargetDocID of the Pending Anchor Change record.
	▪ ChangeType set to "1".
	If @CrawlType is equal to 1 (Full Crawl) and @CrawlID is equal to 0, then the stored procedure MUST add a record to the Anchor Change Set for every record in the following source datasets:
	▪ Anchor Text Info Set
	▪ Annotations Set
	▪ Social Distance Property Set
	The TargetDocID MUST not be NULL and MUST not equal -1. For each new record added to the Anchor Change Set the row MUST have values:
	▪ CrawlID set to @MiscInputData.
	▪ TargetDocID set to TargetDocID from the source datasets.
	▪ ChangeType set to 2.
	If @CrawlType is "2" (Incremental Crawl) or @CrawlID is not equal to "0", the stored procedure MUST add a record to the Anchor Change Set for every record in the Pending Anchor Change Set in which the following is true:
	▪ CrawlID is equal to @MiscInputData.
	▪ Any one of the following conditions is met:

Value of @CrawlStage	Description
	▪ There is an entry in the Anchor Text Info Set in which TargetDocId is equal to the TargetDocId of the Pending Anchor Change record. ▪ There is an entry in the Annotations Set in which TargetDocId is equal to the TargetDocId of the Pending Anchor Change record. ▪ There is an entry in the Social Distance Property Set in which TargetDocId is equal to the TargetDocId of the Pending Anchor Change record. Each new record added to the Anchor Change Set MUST have the following values: ▪ CrawlID set to @MiscInputData. ▪ TargetDocID set to the TargetDocID of the Pending Anchor Change record. ▪ ChangeType set to "1" if the TargetDocID is NULL; otherwise set to "2".
141	Input: @CrawlID MUST be set to the unique identifier of the crawl. Other input parameters MUST be ignored. Actions: The stored procedure checks if the crawl is done and returns the status of the crawl. If this server is not running Microsoft® SharePoint® Foundation 2010 role and there are no records in the following datasets with a CrawlID equal to the @CrawlID passed in, then the stored procedure MUST set @CrawlStatus to 11 (Done). Otherwise it MUST set @CrawlStatus to 4 (Started): ▪ Links set (specified in Section 3.1.1.5) ▪ Colleague Links set (specified in Section 3.1.1.6) ▪ Crawl Queue set (specified in Section 3.1.1.4) If this server is running SharePoint Foundation 2010 role and there are no records in the following datasets with a CrawlID equal to the @CrawlID passed in, then the stored procedure MUST set @CrawlStatus to 11 (Done). Otherwise it MUST set @CrawlStatus to 4 (Started): ▪ Links set (specified in Section 3.1.1.5) ▪ Crawl Queue set (specified in Section 3.1.1.4)
151	Input: @CrawlID MUST be set to unique identifier of the crawl. Other input parameters MUST be ignored. Actions: The stored procedure MUST check if the crawl is partially done. If this server is not running SharePoint Foundation 2010 role and there are no records in the following datasets with a CrawlID equal to the @CrawlID passed in, then the stored procedure MUST set @MiscOutputData to 1, otherwise it MUST set @MiscOutputData to 0: ▪ Links set (specified in Section 3.1.1.5) ▪ Colleague Links set (specified in Section 3.1.1.6)

Value of @CrawlStage	Description
	If this server is running SharePoint Foundation 2010 role and there are no records in the following datasets with a CrawlID equal to the @CrawlID passed in, then the stored procedure MUST set @MiscOutputData to 1, otherwise it MUST set @MiscOutputData to 0: ▪ Links set (specified in Section 3.1.1.5)
145	Input: @CrawlID MUST be set to the unique identifier of the crawl. @ContentSourceID MUST be set to the unique identifier of the content source associated with the crawl. @CrawlType MUST be set to the type of crawl and MUST be a Crawl Type data type, as specified in Section 2.2.1.2. Other input parameters MUST be ignored. This action can be called multiple times for the same @CrawlID, but the stored procedure MUST NOT do anything in any call but the first one. Actions: If @CrawlType is "1" (Full Crawl), the stored procedure MUST add a record to the Crawl Queue set, as specified in Section 3.1.1.4, for every record in the Crawl URL History set, as specified in Section 3.1.1.2, that matches the following criteria: ▪ ContentSourceID is equal to the input parameter @ContentSourceID. ▪ DeletePending is "0", which means that there is no delete request. ▪ ParentUpdateCrawlID is equal to the input parameter @CrawlID. ▪ CommitCrawlID is less than the input parameter @CrawlID. Each new record added to the Crawl Queue set MUST have the following values: ▪ StartAddress, DocID, HostDepth, EnumerationDepth, ContentSourceID, and ProjectID set to the corresponding values in the Crawl URL History record. ▪ TransactionType set to "1" (Delete). ▪ Scope equals "1". ▪ TransactionFlags set to "0". ▪ ChangeLogBatchID set to "0". ▪ SourceDocID set to the ParentDocID of the Crawl URL History record. ▪ DeleteReason equals "3" (Unvisited Full). The set of DocIDs of the records added to the Crawl Queue set is referred to as DocsToDelete in this section. If @CrawlType is "1" (Full Crawl) and @MiscInputData is "1", the stored procedure MUST add a record to the Crawl Queue set for every record in the Crawl URL History set that matches the following criteria: ▪ ContentSourceID is equal to the input parameter @ContentSourceID. ▪ DeletePending is "0", which means that there is no delete request.

Value of @CrawlStage	Description
	▪ CommitCrawlID is less than the input parameter @CrawlID. ▪ TransactionFlags has the bit "0x00000008". ▪ DocID does not belong to the DocsToDelete set. ▪ HostID of the document equals the HostID of the start address for this document or the HostID of the document that belongs to the same start address and has TransactionFlags with a value of "0x0000400". Each new record added to the Crawl Queue Set MUST have the following values: ▪ StartAddress, DocID, TransactionFlags, HostDepth, EnumerationDepth, ContentSourceID, and ProjectID set to the corresponding values in the Crawl URL History record. ▪ TransactionType set to "2" (Modify). ▪ Scope set to "2". ▪ ChangeLogBatchID set to "0". ▪ SourceDocID set to the ParentDocID of the Crawl URL History record. The set of DocIDs of the records added to the Crawl Queue Set is referred to as DocsToRecrawl in this section. If @CrawlType is "1" (Full Crawl) and @MiscInputData is "2", the stored procedure MUST add a record to the Crawl Queue Set for every record in the Crawl URL History Set in which: ▪ ContentSourceID is equal to the input parameter @ContentSourceID. ▪ DeletePending is "0", which means that there is no delete request. ▪ CommitCrawlID is less than the input parameter @CrawlID. ▪ DocID does not belong to the DocsToDelete set. Each new record added to the Crawl Queue set MUST have the following values: ▪ StartAddress, DocID, TransactionFlags, HostDepth, EnumerationDepth, ContentSourceID, and ProjectID set to the corresponding values in the Crawl URL History record. ▪ TransactionType set to "2" (Modify). ▪ Scope set to "2". ▪ ChangeLogBatchID set to "0". ▪ SourceDocID set to the ParentDocID of the Crawl URL History record. The set of DocIDs of the records added to the Crawl Queue Set is added to the DocsToRecrawl Set. The stored procedure MUST set the value of the field CrawlID to the value of the input parameter @CrawlID for all records in the Crawl URL History set that have DocID contained within DocsToRecrawl. If @CrawlType is "1" (Full Crawl), the stored procedure MUST add a record to the Crawl

Value of @CrawlStage	Description
	Queue Set for every record in the Crawl URL History set that matches the following criteria: ▪ ContentSourceID is equal to the input parameter @ContentSourceID. ▪ DeletePending is equal to "0", which means that there is no delete request. ▪ CommitCrawlID is less than the input parameter @CrawlID. ▪ DocID does not belong to the DocsToDelete set. ▪ DocID does not belong to the DocsToRecrawl set. Each new record added to the Crawl Queue MUST have the following values: ▪ StartAddress, DocID, HostDepth, EnumerationDepth, ContentSourceID, and ProjectID set to the corresponding values in the Crawl URL History record. ▪ TransactionType set to "1" (Delete). ▪ Scope equals "1". ▪ TransactionFlags set to "0". ▪ ChangeLogBatchID set to "0". ▪ SourceDocID set to the ParentDocID of the Crawl URL History record. ▪ DeleteReason equal to "3" (Unvisited Full). If @CrawlType is "2" (Incremental Crawl), the stored procedure MUST add a record to the Crawl Queue set for every record in the Crawl URL History set that matches the following criteria: ▪ ContentSourceID is equal to the input parameter @ContentSourceID. ▪ DeletePending is equal to "0", which means that there is no delete request. ▪ CommitCrawlID is less than the input parameter @CrawlID. ▪ UseChangeLog is equal to "0". ▪ ParentUpdateCrawlID is equal to @CrawlID. Each new record added to the Crawl Queue set MUST have the following values: ▪ StartAddress, DocID, HostDepth, EnumerationDepth, ContentSourceID, and ProjectID set to the corresponding values in the Crawl URL History record. ▪ TransactionType set to "1" (Delete). ▪ Scope equals "1". ▪ TransactionFlags set to "0". ▪ ChangeLogBatchID set to "0". ▪ SourceDocID set to the ParentDocID of the Crawl URL History record.

Value of @CrawlStage	Description
	▪ DeleteReason equals "4" (Unvisited Incremental). The set of DocIDs of the records added to the Crawl Queue set is added to the DocsToDelete Set. If @CrawlType is "2" (Incremental Crawl), the stored procedure MUST add a record to the Crawl Queue set for every record in the Crawl URL History set that matches the following values: ▪ ContentSourceID is equal to the input parameter @ContentSourceID. ▪ DeletePending is equal to "0", which means that there is no delete request. ▪ CommitCrawlID is less than the input parameter @CrawlID. ▪ UseChangeLog is equal to "1". ▪ ParentUpdateCrawlID is equal to @CrawlID. Each new record added to the Crawl Queue Set MUST have the following values: ▪ StartAddress, DocID, HostDepth, EnumerationDepth, ContentSourceID, and ProjectID set to the corresponding values in the Crawl URL History record. ▪ TransactionType set to "1" (Delete). ▪ Scope equals "2". ▪ TransactionFlags set to "0". ▪ ChangeLogBatchID set to "0". ▪ SourceDocID set to the ParentDocID of the Crawl URL History record. ▪ DeleteReason equals "5" (Delete Unvisited Log). The set of DocIDs of the records added to the Crawl Queue set is added to the DocsToDelete Set. The stored procedure MUST set DeletePending to "1" for all records in the Crawl URL History set that have a DocID in the DocsToDelete Set. If a record is in the Crawl Queue with CrawlID equal to @CrawlID, the stored procedure MUST set @MiscOutputData to "1".
149	Input: @CrawlID MUST be set to the unique identifier of the crawl. @ProjectID MUST be set to the identifier of the project and MUST be a Project Identifier data type, as specified in Section 2.2.1.1. Other input parameters MUST be ignored. Actions: The stored procedure MUST complete the crawl with the identifier @CrawlID. The stored procedure is called multiple times for the same crawl. The stored procedure is called for the same crawl until @MiscOutputData is not set to "1". It can be called for the same crawl after it, but it MUST NOT perform any actions. The stored procedure MUST add a record to the Local Completed Crawls set, as specified in Section 3.1.1.27, with CrawlID set to the input parameter @CrawlID and Time set to the current time in UTC.

Value of @CrawlStage	Description
	The rest of the actions MUST NOT be done if this server is running the SharePoint Foundation 2010 role. If the input parameter @ProjectID is "1" (Portal Content), the stored procedure MUST perform the following sequence of actions: The stored procedure MUST update the statistics SuccessCount, WarningCount, ErrorCount, DeleteCount, and LevelHighErrorCount in the Crawled Hosts set, as specified in Section 3.1.1.7, for all hosts that were crawled during the specified crawl. The stored procedure MUST add a record to the Changed Anchor Target Document set, as specified in Section 3.1.1.20, for every record in the Anchor Text Info set, as specified in Section 3.1.1.8, in which there is an entry in the Changed Anchor Deleted Documents, as specified in Section 3.1.1.23, with CrawlID equal to @CrawlID and DocID equal to DocID of the Anchor Text Info record. Each new record added to the Changed Anchor Target Document set MUST have the following values: ▪ CrawlID set to @CrawlID. ▪ DocID set to TargetDocID of the Anchor Text Info record. The stored procedure MUST set TargetDocID to "-1" in the Anchor Text Info set for all records for which a Changed Anchor Target Document Set was added. The stored procedure MUST update all records in the Anchor Text Info set that match the following criteria: ▪ TargetDocID is either NULL or "-1". ▪ There is an entry in the Changed Anchor Source Documents set, as specified in Section 3.1.1.21, with CrawlID equal to @CrawlID and DocID equal to the SourceDocID of the Anchor Text Info record. The update MUST use a record from the Crawl URL History Set, as specified in Section 3.1.1.2, that matches all of the following criteria: ▪ DisplayURL is equal to the Link of the Anchor Text Info record. ▪ DisplayHash is equal to the LinkHash of the Anchor Text Info record. ▪ DeletePending is "0", which means there is no delete request. ▪ CrawlID is equal to CommitCrawlID. The update to the Anchor Text Info record MUST set the following values: ▪ TargetDocID set to the DocID of the Crawl URL History record. ▪ CrawlID set to @CrawlID. ▪ If the Anchor Text Info record has a value of "0" for InterSite and has a value for SourceDocSiteID that is different from the SiteID of the corresponding Crawl URL History record, InterSite in the Anchor Text Info record MUST be set to "1". Otherwise, it MUST be set it to "0". The stored procedure MUST insert a new record, with the format @CrawlID, DocID, into the Changed Anchor Target Documents Set for every updated record in the Anchor Text Info set. The stored procedure MUST update all records in the Anchor Text Info Set for which:

Value of @CrawlStage	Description

	▪ TargetDocID is either NULL or "-1".
	▪ There is a record in the Crawl URL History set, as specified in Section 3.1.1.2, that matches all of the following criteria:
	  ▪ DisplayURL is equal to the Link of the Anchor Text Info record.
	  ▪ DisplayHash is equal to the LinkHash of the Anchor Text Info record.
	  ▪ DeletePending is "0", which means that there is no delete request.
	  ▪ CrawlID is equal to CommitCrawlID.
	  ▪ IndexType is equal to "1".
	  ▪ There exists a record in the Changed Anchor Committed Documents Set, as specified in section 3.1.1.22, that has a DocID that is equal to the DocID of the Crawl URL History record and CrawlID is equal to @CrawlID.
	The update to the Anchor Text Info record MUST be done with the following values:
	▪ TargetDocID set to the DocID of the Crawl URL History record.
	▪ CrawlID set to @CrawlID.
	▪ If the Anchor Text Info record has a value of "0" for InterSite and has a value for SourceDocSiteID that is different from the SiteID of the corresponding Crawl URL History record, set InterSite in the Anchor Text Info record to "1". Otherwise, set it to "0".
	The stored procedure MUST insert a new record, with the format @CrawlID, DocID, into the Changed Anchor Target Documents Set for every updated record in the Anchor Text Info Set.
	The stored procedure MUST add additional records to the Anchor Text Info set if there are duplicate URLs. If the Anchor Text Info set has more than one record where the DisplayURL of the TargetDocID are the same, the stored procedure MUST add new records to the Anchor Text info set for every combination of two records where the DisplayURL of TargetDocID are the same.
	For example, if the records with duplicate DisplayURLs have the following attributes:
	▪ SourceDocID1, Link, TargetDocID1
	▪ SourceDocID2, Link, TargetDocID2
	▪ SourceDocID3, Link, TargetDocID3
	The new records added to the Anchor Text info Set have the following values:
	▪ SourceDocID1, Link, TargetDocID2
	▪ SourceDocID1, Link, TargetDocID3
	▪ SourceDocID2, Link, TargetDocID1
	▪ SourceDocID2, Link, TargetDocID3
	▪ SourceDocID3, Link, TargetDocID1

Value of @CrawlStage	Description
	▪ SourceDocID3, Link, TargetDocID2 For every record in the Social Distance Property set, as specified in Section 3.1.1.11, for which there is a record in the Changed Anchor Deleted Documents set, as specified in Section 3.1.1.23, that match the following criteria: ▪ DocID is equal to the SourceDocID of the Social Distance Property set. ▪ CrawlID is equal to @CrawlID. The stored procedure MUST copy the Social Distance Property set record into the Social Distance Deleted Property set, as specified in section 3.1.1.12. The stored procedure MUST set TargetDocID to "-1" for every record in the Social Distance Property set that matches the following criteria: ▪ There is a record in the Changed Anchor Deleted Documents set that has a CrawlID that is equal to @CrawlID and DocID equals the TargetDocID of the Social Distance set record. ▪ There are no records in the Social Distance Deleted Property set, as specified in Section 3.1.1.12, where the SourceDocID and TargetUserID are equal to the SourceDocID and TargetUserID of the Social Distance set record. The stored procedure MUST remove every entry from the UserID DocID Pair set, as specified in section 3.1.1.24, for which there is a record in the Changed Anchor Target Documents set in which the CrawlID of the record equals the input parameter @CrawlID and the DocID of the record equals the DocID of the entry. The stored procedure MUST update every record in the Social Distance Property set that matches the following criteria: ▪ There is a record in the UserID DocID Pair Set in which the UserID is equal to the TargetUserID of the Social Distance Property record. ▪ There is a record in the Social Distance New Property set that has a CrawlID that is equal to @CrawlID. The SourceDocID and the TargetUserID are equal to SourceDocID and TargetUserID of the Social Distance Property record. For each Social Distance record to be updated, the stored procedure MUST set TargetDocID to the DocID of the UserID DocID Pair. The stored procedure MUST update every record from the Social Distance Property set that matches the following: ▪ TargetDocID is either NULL or "-1". ▪ There is a record in the UserID DocID Pair set in which the UserID is equal to the TargetUserID of the Social Distance Property record. ▪ There is a record in the Changed Anchor Committed Documents set that has a CrawlID that is equal to @CrawlID, and DocID is equal to the DocID of the UserID DocID Pair record. For each Social Distance Property record to be updated, the stored procedure MUST set TargetDocID to the DocID of the UserID DocID Pair. For each Social Distance Property that was updated, the stored procedure MUST add a record in the Social Distance New Property set, as specified in section 3.1.1.13, with CrawlID set to @CrawlID, and SourceDocID and TargetUserID set to SourceDocID and TargetUserID of the updated Social Distance Property record. The stored procedure MUST add a record to the Pending Anchor Change set for every

Value of @CrawlStage	Description
	record in the Social Distance set that matches either of the following: ▪ There is a record in the Social Distance New Property set in which CrawlID is equal to @CrawlID and SourceDocID and TargetUserID are equal to SourceDocID and TargetUserID of the Social Distance record. ▪ There is a record in the Social Distance Deleted Properties set, as specified in section 3.1.1.12, that has a CrawlID that is equal to @CrawlID and SourceDocID and TargetUserID are equal to SourceDocID and TargetUserID of the Social Distance record Each new record added to the Pending Anchor Change set MUST have the following values: ▪ CrawlID set to @CrawlID. ▪ TargetDocID set to TargetDocID of the Social Distance Property record. The stored procedure MUST add a record to the Pending Anchor Change set for every record in the Social Distance Property set that matches the following: ▪ TargetDocID is equal to the SourceDocID. ▪ There is a record in the Social Distance New Property set that has a CrawlID that is equal to @CrawlID and SourceDocID and TargetUserID are equal to SourceDocID and TargetUserID of the Social Distance Property record. Or there is a record in the Social Distance Deleted Property set that has a CrawlID that is equal to @CrawlID and SourceDocID and TargetUserID are equal to SourceDocID and TargetUserID of the Social Distance Property record. Each new record added to the Pending Anchor Change Set MUST have the following values: ▪ CrawlID set to @CrawlID. ▪ TargetDocID set to the TargetDocID of the Social Distance Property record. If there is also a record in the Social Distance Property set that matches the following list, the stored procedure MUST insert a new record into the Pending Anchor Change set with CrawlID equal to @CrawlID and TargetDocID equal to the TargetDocID of the Social Distance Property record. ▪ TargetDocID of the record equals the SourceDocID of the Social Distance Property record. ▪ SourceDocID of the record does not equal the TargetDocID of the Social Distance Property record. ▪ This is a public relationship link. In other words, the value of Pid in the Social Distance Property record is "394". ▪ TargetDocID of the record is not "-1". ▪ There is no record in the Anchor Pending Change set that has a value of CrawlID that equals @CrawlID and the value of TargetDocID equals TargetDocID of the record. The stored procedure MUST delete every record in the Social Distance Property set for which there is a record in the Social Distance Delete Properties Set that matches the

Value of @CrawlStage	Description
	following: ▪ CrawlID is equal to @CrawlID. ▪ SourceDocID and TargetUserID of the Social Distance Delete Property record is equal to SourceDocID and TargetUserID of the Social Distance Property record. The stored procedure MUST delete all records from the Social Distance New Links Set, Social Distance Deleted Links Set, and Social Distance Unchanged Links Set. If the input parameter @ProjectID is "2" (Anchor Project), the stored procedure MUST delete all records from the following datasets: ▪ Anchor Change Set ▪ Pending Anchor Change set ▪ Changed Anchor Target Documents set ▪ Changed Anchor Source Documents set ▪ Changed Anchor Deleted Documents set ▪ Changed Anchor Committed Documents set
153	Input: @CrawlID MUST be set to the unique identifier of the crawl. Other input parameters MUST be ignored. The stored procedure MUST set @MiscOutputData to the maximum of: ▪ The maximum ChildrenCount among all the records from Reported Crawl Error set (specified in Section 3.1.1.16). ▪ The number of documents in the Reported Crawl Error record with CrawlID equal to @CrawlID.
155	Input: @CrawlID MUST be set to the unique identifier of the crawl. Other input parameters MUST be ignored. The stored procedure MUST set @MiscOutputData to the sum of the ChildrenCount value of all the records of the Reported Crawl Error Set, as described in section 3.1.1.16, in which the value of CrawlID is equal to the input parameter @CrawlID.
154	Input: @CrawlID MUST be set to the unique identifier of the crawl. Other input parameters MUST be ignored. The stored procedure MUST update statistics (SuccessCount, WarningCount, ErrorCount, DeleteCount, LevelHighErrorCount) in the Crawled Hosts set (specified in Section 3.1.1.7) for all hosts that were crawled during the specified crawl with identifier @CrawlID.
115	Input: @ContentSourceID MUST be set to the unique identifier of the content source associated with the crawl.

Value of @CrawlStage	Description
	@ApplicationType MUST be set to 0. Other input parameters MUST be ignored. Actions: The stored procedure MUST complete part of stopping of the specified crawl. The stored procedure is called multiple times for the same crawl until the output parameter @MiscOutputData is not set to 1. The stored procedure MUST perform some part of the following actions on each invocation. The stored procedure MUST set DeletePending to 0 (clear the delete request) for every record in Crawl URL History set (specified in Section 3.1.1.2) that match the following criterion: ▪ ContentSourceID is equal to @ContentSourceID ▪ DeletePending is not equal to 0 (there is a delete request) ▪ There is a record in the Crawl Queue set (specified in Section 3.1.1.4) with CrawlID equal to @CrawlID ▪ TransactionType is equal to 1 (Delete) ▪ DocID is equal to the DocID of the Crawl URL History record. The stored procedure MUST remove from the Crawl Queue set all records that have CrawlID equal to @CrawlID. The stored procedure MUST remove from the Links set (specified in Section 3.1.1.5) all records that have CrawlID equal to @CrawlID. If this server is not running SharePoint Foundation 2010 role the stored procedure MUST remove from the Colleague Links set (specified in Section 3.1.1.6) all records that have CrawlID equal to @CrawlID. The following is only done when @ProjectID is equal to 1 (Portal Content), the stored procedure MUST update statistics (SuccessCount, WarningCount, ErrorCount, DeleteCount, LevelHighErrorCount) in the Crawled Host set (specified in Section 3.1.1.7) for all hosts which were crawled by the crawl with identifier @CrawlID. If @ProjectId is 1 (Portal Content) and this server is not running SharePoint Foundation 2010 role, the stored procedure MUST remove all records from: ▪ Changed Anchor Target Documents set (specified in Section 3.1.1.20) ▪ Changed Anchor Source Documents set (specified in Section 3.1.1.21) ▪ Changed Anchor Deleted Documents set (specified in Section 3.1.1.23) ▪ Changed Anchor Committed Documents set (specified in Section 3.1.1.22) ▪ Social Distance Deleted Property set (specified in Section 3.1.1.12) that have CrawlID equal to @CrawlID. If @ProjectId is 1 (Portal Content) the stored procedure MUST add a record into the Local Completed Crawls set (specified in Section 3.1.1.27) CrawlID set to @CrawlID and Time set to the current time in UTC.
92	Input: All input parameters MUST be ignored. The stored procedure MUST perform cleanup of crawl-related sets. Action: The stored procedure MUST delete all records from the following sets:

Value of @CrawlStage	Description
	▪ Crawl URL History set, as specified in section 3.1.1.2. ▪ Links set, as specified in section 3.1.1.5. ▪ Crawl Queue set, as specified in section 3.1.1.4. ▪ Crawled Hosts set, as specified in section 3.1.1.7. ▪ Deleted URL set, as specified in section 3.1.1.3. ▪ Reported Crawl Error set, as specified in section 3.1.1.16. ▪ Local Completed Crawls, as specified in section 3.1.1.27. ▪ Pending Annotations Status, as described in section 3.1.1.10. If this server is not running the SharePoint Foundation 2010 role, the stored procedure MUST delete all records from the following sets: ▪ Anchor Text Info set, as specified in Section 3.1.1.8. ▪ Changed Anchor Target Documents set, as specified in Section 3.1.1.20. ▪ Changed Anchor Source Documents set, as specified in Section 3.1.1.21. ▪ Changed Anchor Deleted Documents set, as specified in Section 3.1.1.23. ▪ Changed Anchor Committed Documents set, as specified in Section 3.1.1.22. ▪ Pending Anchor Change set, as specified in Section 3.1.1.15. ▪ Anchor Change set, as specified in Section 3.1.1.14. ▪ Annotations set, as specified in Section 3.1.1.37. ▪ Pending Annotations set, as specified in Section 3.1.1.9. ▪ Colleague Links set, as specified in Section 3.1.1.6. ▪ Social Distance Deleted Property set, as specified in section 3.1.1.12.
163	Input: @MiscInputData MUST be set to the unique identifier of crawl component to be disabled. Other input parameters MUST be ignored Actions: The stored procedure MUST disable the specified crawl component. The stored procedure MUST set Status to 3 (Disabled) for the record in the Local Crawl Components set, as specified in Section 3.1.1.26, that has a ComponentID equal to input parameter @MiscInputData. The stored procedure MUST set value of BatchID in the Crawl Queue set (specified in Section 3.1.1.4) to 0 for all documents in the Crawl URL History set (specified in Section 3.1.1.2) which were being crawled by the specified crawl component in @MiscInputData. The stored procedure MUST increment the counter Retry of the Crawl URL History set by 1 for all documents that were updated.

Return Code Values: This stored procedure MUST return 1 upon completion.

Result Sets: SHOULD NOT<4> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.5.8 proc_MSS_CrawlAdmin

This stored procedure performs crawl administration operations.

```
PROCEDURE(proc_MSS_CrawlAdmin)(
    @ComponentID            int,
    @ProjectID              int,
    @CrawlStage             int,
    @CrawlType              int,
    @CrawlID                int,
    @ContentSourceID        int,
    @ApplicationType        int,
    @MiscInputData          int,
    @MiscOutputData         int OUTPUT,
    @CrawlStatus            int OUTPUT,
    @CrawlSubStatus         int OUTPUT
);
```

@ComponentID: A crawl component unique identifier

@ProjectID: A project identifier as specified in ProjectIdentifier (section 2.2.1.1).

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

@CrawlType: Type of crawl as specified in Crawl Type (section 2.2.1.2)

@CrawlID: Unique identifier of the crawl

@ContentSourceID: unique identifier of content source

@ApplicationType: MUST be set to 0.

@MiscInputData: Input data, depends on value of @CrawlStage. See the following table.

@MiscOutputData: On return, it MUST be set to @CrawlID unless it is specified otherwise in the following table.

@CrawlStatus: On return, it MUST be set to CrawlStatus from Crawl Status Set (section 2.2.1.4) unless other value is specified in the following table.

@CrawlSubStatus: On return, it MUST be set to CrawlSubstatus from Crawl Status Set (sections 2.2.1.5, 2.2.1.6, 2.2.1.7) unless other value is specified in the following table.

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

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

The following table specifies the input parameters for this stored procedure, as well as the final value of @MiscOutputData depending on @CrawlStage

Value of Crawl Stage	Description
96	Input: @MiscInputData: MUST be time duration in minutes. It represents the time after which the unresponsive crawl components are disabled. Actions: In this case the stored procedure MUST return the identifier of one of the crawl components that are currently unresponsive and need to be disabled. A crawl component is considered unresponsive if it did not report status for a given period of time passed in by @MiscInputData, which means that more minutes have elapsed after the RecordTime of Crawl Component Set, as specified in[MS-SRCHTP] section 3.1.1.3, than the number of minutes specified in @MiscInputData. The stored procedure MUST set @MiscOutputData to unique identifier of one of the crawl components which did not report status in the last @MiscInputData minutes. The component chosen is arbitrarily selected from the list of components which did not report status. If there is no such component then the stored procedure MUST set @MiscOutputData to 0x7fffffff. The stored procedure also checks if a **master crawl component** exists (Master field is set to 1 for any crawl component from Crawl Component Set). If the Master field is equal to 0 for all the records in the Crawl Components Set ([MS-SRCHTP] section 3.1.1.3), the stored procedure MUST set Master field to 1 for one arbitrary record which meets the following conditions, within the set: ▪ State field is Ready ▪ DesiredState is not equal to Inactive
97	Input: @ComponentID MUST be set to the unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: The stored procedure MUST set @MiscOutputData to state of crawl component from Crawl Component Set ([MS-SRCHTP] section 3.1.1.3)
98	Input: @ComponentID MUST be set to the unique identifier of a crawl component. @MiscInputData MUST contain a crawl component state value ([MS-SRCHTP] section 3.1.1.3). Other input parameters MUST be ignored. Actions: The stored procedure MUST set State to @MiscInputData for the crawl component from Crawl Component Set ([MS-SRCHTP] section 3.1.1.3). If the server is running the Microsoft® SharePoint® 2010 Products and Technologies role and @MiscInputData equals "5" (Inactive), the stored procedure MUST set Master to "0" for the crawl component from the Crawl Component Set.
94	Input: @ComponentID MUST be set to the unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: The stored procedure MUST set State to Ready for crawl component from Crawl Component Set ([MS-SRCHTP] section 3.1.1.3) identified by @ComponentID. For every record in the Crawl Status Set, as specified in section 3.1.1.1, in which the following is true: ▪ Status is not equal to "Failed", "Done" or "Stop".

Value of Crawl Stage	Description

	▪ There is a record in the Crawl Store Status Set, as specified in section 3.1.1.28, in which:
	  ▪ CrawlID is equal to the CrawlID of the Crawl Status record.
	  ▪ GthrDBID is equal to the ordinal of the crawl store to which the crawl component with the identifier @ComponentID belongs.
	The stored procedure MUST insert a new record into the Crawl Component Status Set, as specified in section 3.1.1.25, with:
	▪ ComponentID set to @ComponentID.
	▪ CrawlID set to the value of the field CrawlID of the record in the Crawl Status Set.
	▪ Status set to "0".

95	Input: @ComponentID MUST contain the unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: The stored procedure updates the status of crawl component as follows: If the State of the crawl component from Crawl Component Set ([MS-SRCHTP] section 3.1.1.3) with the CrawlComponentID equal to @ComponentID is not Disabled or DisableForRemove, then the stored procedure MUST set ReportTime for this component to UTC. The stored procedure MUST set @MiscOutputData to the State of crawl component with identifier @ComponentID.
100	Input: @ProjectID: A project identifier as specified in Project Identifier (section 2.2.1.1) @CrawlType: Type of crawl as specified in Crawl Type (section 2.2.1.2) @ContentSourceID: Unique identifier of a content source. @MiscInputData: If @ProjectID equals "2", this parameter MUST be set to the CrawlID of the crawl that triggered this anchor crawl. Otherwise, the parameter MUST be ignored. Other input parameters MUST be ignored. Actions: When called with these input parameters, the stored procedure MUST start the crawl. The stored procedure MUST create new record in Crawl Status Set (section 3.1.1.1) with values: ▪ ProjectID, CrawlType, ContentSourceID, MainCrawlID MUST be set to the input parameters @ProjectID, @CrawlType, @ContentSourceID and @MiscInputData; ▪ Request MUST be set to Start; ▪ RequestTime MUST be set to UTC; ▪ CrawlID MUST be set to an ever increasing number (there MUST not be two structures with the same CrawlID in Crawl Status Set (section 3.1.1.1)) The stored procedure MUST set @MiscOutputData to @CrawlID of newly created record.
108	Input:

Value of Crawl Stage	Description
	@MiscInputData: The unique identifier of a crawl store ([MS-SRCHTP] section 3.1.1.3). @CrawlID: The unique identifier of an ongoing crawl. Other input parameters MUST be ignored. Actions: In this case the stored procedure initiates a request for all the active crawl components associated with the crawl store specified in @MiscInputData to join crawl specified by @CrawlID. The stored procedure MUST perform the following actions: If the crawl store specified by @MiscInputData is not participating in the crawl @CrawlID (there is not structure in Crawl Store State Set with GthrDBID equal to @MiscInputData and CrawlID and equal to @CrawlID, as specified in section 3.1.1.28), then the crawl store with identifier @MiscInputData MUST join the crawl. In this case the stored procedure MUST add a new record to Crawl Store Status Set with GthrDBID set to @MiscInputData, CrawlID set to @CrawlID, JoinTime set to UTC. All crawl components from this crawl with State not equal to Disabled, DisableForRemove or InactiveStore, MUST join the crawl. For each of them the stored procedure MUST add a new record to the Crawl Component Status Set with ComponentID set to the identifier of the crawl component, CrawlID set to @CrawlID, and Status set to 0.
102	Input: @CrawlType: Type of crawl as specified in Crawl Type, as specified in section 2.2.1.2. @CrawlID: The unique identifier of a crawl. @ContentSourceID: The unique identifier of a content source. Other input parameters MUST be ignored. Actions: The stored procedure MUST set CrawlType to @CrawlType for crawl with unique identifier @CrawlID in Crawl Status Set, as specified in section 3.1.1.1. If all the following conditions are true: ▪ @CrawlType is not "Delete Crawl". ▪ There exists another record in the Crawl Status Set in which: ▪ ContentSourceID equals @ContentSourceID. ▪ CrawlType is not "Delete Crawl". ▪ The value of CrawlID does not equal the value of @CrawlID. ▪ The status does not equal "Forbid", "Done" or "Stopped". The stored procedure MUST set the following: ▪ Status to "Forbid". ▪ Substatus to "0". ▪ Request to "0". for record in Crawl Status Set with the value CrawlID equals @CrawlID—that is, it MUST fail the crawl specified by @CrawlID If @CrawlType is Delete Crawl, the stored procedure MUST update the previously created Delete Crawl. In this case it MUST set CrawlID to @CrawlID for the started Delete Crawl associated with the content source with identifier @ContentSourceID in the Requested Delete Crawl Set (section 3.1.1.19)

Value of Crawl Stage	Description
	The stored procedure MUST set ▪ Status to Initializing ▪ SubStatus to Adding Start Address ▪ Request to 0 ▪ StartTime to UTC in Crawl Status Set (section 3.1.1.1) for crawl with identifier @CrawlID.
103	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: The stored procedure MUST set the following fields of the record in Crawl Status Set (section 3.1.1.1) where CrawlID is equal to @CrawlID: ▪ Status MUST be set to Initializing; ▪ SubStatus MUST be set to Wait For Crawl Components; ▪ Request MUST be set to 0; ▪ StartTime MUST be set to UTC.
104	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: If there are 0 components which are participating in the crawl with identifier @CrawlID but which are not crawling (that is, State not equals Crawling in Crawl Components Status (section 3.1.1.25)), then the stored procedure MUST set Status of crawl with identifier @CrawlID to Started in the Crawl Status Set (section 3.1.1.1).
105	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Action: The stored procedure MUST set Status to Forbid, SubStatus to 0, Request to 0 in Crawl Status Set (section 3.1.1.1) for the crawl specified in @CrawlID, that is, it MUST fail the crawl @CrawlID.
140	Input: @CrawlID: The unique identifier of a crawl. All other input parameters MUST be ignored. Actions: The stored procedure MUST not perform any action if the crawl in Crawl Status Set (section 3.1.1.1) specified by @CrawlID has Status not equal to Started or if there is a crawl store which participates in the crawl with identifier @CrawlID but there are no crawl components belonging to this crawl store. If all of the of following conditions are true:

Value of Crawl Stage	Description
	▪ The record in the Crawl Status Set, as specified in section 3.1.1.1, with CrawlID equal to @CrawlID has SubStatus equal to Delete Unvisited. ▪ All the crawl components that participate in the crawl with the identifier @CrawlID reported that they are not doing anything. The stored procedure MUST set SubStatus to Wait All Crawl Stores, as specified in section 2.2.1.4, in the Crawl History Set for the crawl with the identifier @CrawlID and stop execution. If the record in the Crawl Status Set with CrawlID equal to @CrawlID has SubStatus equal to "Started", the stored procedure MUST identify if the corresponding crawl has completed the current stage. This crawl is considered to have completed the current stage if all the crawl components are not processing items for a period of time that is defined by a specific implementation. If the crawl completed the current stage, and is not an anchor crawl and the crawl type is not Delete Crawl, the stored procedure MUST set SubStatus to Move Unvisited Items to Crawl Queue, as specified in section 3.1.1.4, for the crawl in the Crawl Status Set, as specified in section 3.1.1.1, with CrawlID equal to @CrawlID. If the crawl completed the current stage and this is an anchor crawl or the crawl type is a Delete Crawl, the stored procedure MUST set SubStatus to Wait All Crawl Stores.
144	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: The stored procedure MUST mark all components participating in crawl specified by @CrawlID as being used. (Activity Info structure in Crawling in Crawl Components Status (section 3.1.1.25)) The stored procedure MUST change SubStatus of crawl with identifier @CrawlID to Delete Unvisited in the Crawl Status Set (section 3.1.1.1)
150	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: This stored procedure MUST change Status of the crawl specified by @CrawlID to Completing and SubStatus to Wait For Crawl Components (section 2.2.1.6)
146	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: If there are no crawl components participating in crawl with identifier @CrawlID with Status not equal to Done in Crawl Components Status (section 3.1.1.25) then the stored procedure MUST change SubStatus to Completing in Crawl Status Set (section 3.1.1.1) for records with CrawlID equal to @CrawlID.
152	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: The stored procedure MUST change SubStatus to Deletes Pending in Crawl Status Set (section

Value of Crawl Stage	Description
	3.1.1.1) for records with CrawlID equal to @CrawlID.
148	Input: @CrawlID: The unique identifier of a crawl. @ProjectID: A project identifier as specified in Project Identifier (2.2.1.1) @ComponentID: The unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: The stored procedure MUST change Status to Done, SubStatus to 0 and EndTime to UTC in Crawl Status Set (section 3.1.1.1) for crawl with identifier @CrawlID. If @ProjectID equals Portal Content, the stored procedure MUST initiate an anchor crawl with @CrawlID as the main crawl. The stored procedure MUST add a new record to the Crawl Status Set with ProjectID set to Anchor Project, Request set to "Request to Start", CrawlType set to "Incremental Crawl", ContentSourceID set to "1", RequestTime set to UTC, and MainCrawlID set to @CrawlID. If @ProjectID is equal to the Anchor Project and there is a failed crawl, or Status equals Forbid in the Crawl Status Set, the stored procedure MUST request the anchor text crawl to start one more time. The stored procedure MUST add a new record to the Crawl Status Set with ProjectID set to Anchor Project, Request set to "Request to Start", CrawlType set to "Incremental Crawl", ContentSourceID set to "1", RequestTime set to UTC, and MainCrawlID set to the Main Crawl identifier of the failed anchor crawl.
110	Input: @ContentSourceID: The unique identifier of content source. Other input parameters MUST be ignored. Actions: The stored procedure MUST initiate stopping of the crawl with the content source specified by @ContentSourceID. If there is crawl which is actively crawling (Status not Forbid, Done, Stop, Stopping) associated with the content source with identifier @ContentSourceID and this crawl type is not Delete Crawl, then the stored procedure MUST set Request to Request to Stop in Crawl Status Set (section 3.1.1.1) for crawl with identifier @CrawlID.
111	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: The stored procedure MUST set Status to Stopping and SubStatus to Wait for Crawl Components to Stop in Crawl Status Set (section 3.1.1.1) for crawl specified by @CrawlID.
113	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: The stored procedure MUST perform the following actions only if there are no crawl components specified by @CrawlID have status that is not equal to Stopped in Crawl Components Status (section 3.1.1.25) In this case, the stored procedure MUST change SubStatus to Complete Stop in Crawl Status Set (section 3.1.1.1) for the crawl with identifier @CrawlID.

Value of Crawl Stage	Description
114	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: The stored procedure MUST change Status to Stopped in Crawl Status Set (section 3.1.1.1) for crawl with identifier @CrawlID.
120	Input: @ContentSourceID: The unique identifier of a content source. Other input parameters MUST be ignored. Actions: The stored procedure MUST initiate pausing of the crawl for the content source specified by @ContentSourceID. The stored procedure MUST set Request to "Request to Pause" in Crawl Status Set (section 3.1.1.1) for the crawl which meets the following conditions: ▪ The crawl which is associated with content source with identifier @ContentSourceID; ▪ Status field for this crawl in Crawl Status Set (section 3.1.1.1) is set to Started; ▪ Request field for this crawl in Crawl Status Set (section 3.1.1.1) is set to 0.
121	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: The stored procedure MUST set Status to Pausing and Request to 0 in Crawl Status Set (section 3.1.1.1) for crawl specified by @CrawlID.
123	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: If there are no components specified by @CrawlID having a status other than Paused in Crawl Components Status (section 3.1.1.25), then the stored procedure MUST set Status to Paused in Crawl Status Set (section 3.1.1.1) for crawl with identifier @CrawlID.
130	Input: @ContentSourceID: The unique identifier of a content source. Other input parameters MUST be ignored. Actions: The stored procedure MUST initiate the operation of resuming the crawl for the content source specified by @ContentSourceID. If there is crawl associated with the content source specified by @ContentSourceID and the crawl status is Paused, then the stored procedure MUST set Request to Request to Resume in Crawl Status Set (section 3.1.1.1) for this crawl.
131	Input: @CrawlID: The unique identifier of crawl.

Value of Crawl Stage	Description
	Other input parameters MUST be ignored. Actions: The stored procedure MUST set Status to Resuming and Request to 0 in Crawl Status Set (section 3.1.1.1) for crawl specified by @CrawlID.
133	Input: @CrawlID: The unique identifier of a crawl. Other input parameters MUST be ignored. Actions: If there are no crawl components which participate in the crawl with identifier @CrawlID with Status not equal to Started in Crawl Component Status Set (section 3.1.1.25), the stored procedure MUST change status to Started in Crawl Status (section 3.1.1.1) for crawl with identifier @CrawlID.
91	Input: All input parameters MUST be ignored. Actions: The stored procedure MUST delete all records from the following crawl-related sets: Crawl Status Set Crawl Component Status Set Requested Delete Crawl Set The stored procedure MUST set DocCount to 0 in Crawl Stores Set ([MS-SRCHTP] section 3.1.1.3).
106	Input: @CrawlID: The unique identifier of a crawl. @ComponentID: The unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: For the record in Crawl Component Status Set (section 3.1.1.25) specified by @ComponentID and @CrawlID, the stored procedure MUST set Status to Started. That is, the crawl component with identifier @ComponentID is actively participating in the crawl with identifier @CrawlID.
142	Input: @CrawlID: The unique identifier of a crawl. @ComponentID: The unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: For the record in Crawl Component Status Set (section 3.1.1.25) specified by @ComponentID and @CrawlID the stored procedure MUST update its Activity Info to indicate that the crawl component identified by @ComponentID is not processing any items.
143	Input: @CrawlID: The unique identifier of a crawl. @ComponentID: The unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: For the record in Crawl Component Status specified by @ComponentID and @CrawlID the

Value of Crawl Stage	Description
	stored procedure MUST update its Activity Info to indicate that the crawl component identified by @ComponentID is currently processing items.
147	Input: @CrawlID: The unique identifier of a crawl. @ComponentID: The unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: For the record in Crawl Component Status Set (section 3.1.1.25) specified by @ComponentID and @CrawlID the stored procedure MUST set its Status to Done.
112	Input: @CrawlID: The unique identifier of a crawl. @ComponentID: The unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: For the record in Crawl Component Status Set (section 3.1.1.25) specified by @ComponentID and @CrawlID the stored procedure MUST set its Status to Stopped.
122	Input: @CrawlID: The unique identifier of crawl. @ComponentID: The unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: For the record in Crawl Component Status Set (section 3.1.1.25) specified by @ComponentID and @CrawlID the stored procedure MUST set its Status to Paused.
132	Input: @CrawlID: The unique identifier of a crawl. @ComponentID: The unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: For the record in Crawl Component Status Set (section 3.1.1.25) specified by @ComponentID and @CrawlID the stored procedure MUST set its Status to Started.
162	Input: @MiscInputData: The unique identifier of a crawl component. Other input parameters MUST be ignored. Actions: The stored procedure MUST set State to Disabled in Crawl Components Set ([MS-SRCHTP] section 3.1.1.3) for crawl component specified by @MiscInputData. The stored procedure MUST remove all records for crawl component specified by @MiscInputData from Crawl Components Status Set (section 3.1.1.25).
165	Input: @MiscInputData: The unique identifier of a crawl component. Other input parameters MUST be ignored. Actions:

Value of Crawl Stage	Description
	The stored procedure MUST set State to DisableForRemove in Crawl Components Set ([MS-SRCHTP] section 3.1.1.3) for crawl component specified by @MiscInputData. The stored procedure MUST remove all records for crawl component specified by @MiscInputData from Crawl Components Status Set (section 3.1.1.25).
160	Input: @ContentSourceID: The unique identifier of a content source. Other input parameters MUST be ignored. Actions: The stored procedure MUST initiate the deletion of the content source specified by @ContentSourceID. The stored procedure MUST add a new record to Requested Delete Crawl Set (section 3.1.1.19) with ContentSourceId set to @ContentSourceID and StartAddressId set to 0
161	Input: @ContentSourceID: The unique identifier of a content source. @MiscInputData: The unique identifier of a start address. Other input parameters MUST be ignored. Actions: The stored procedure MUST initiate the deletion of the start address specified by @MiscInputData from the content source specified by @ContentSourceID. The stored procedure MUST add new record to Requested Delete Crawl Set (section 3.1.1.19) with ContentSourceId set to @ContentSourceID and StartAddressId set to @MiscInputData.
164	Input: @ComponentID: The unique identifier of a crawl component. @ProjectID: A project identifier. @ContentSourceID: The unique identifier of a content source. @ApplicationType: Type of the application, . Actions: The stored procedure MUST create new record in Crawl Status Set with ProjectID set to @ProjectID, Request set to Start, CrawlType set to Delete, ContentSourceID set to @ContentSourceID, RequestTime set to UTC and MainCrawlID set to 0.

3.1.5.9 proc_MSS_CrawlStoreToJoinAllOngoingCrawls

The **proc_MSS_CrawlStoreToJoinAllOngoingCrawls** stored procedure is called to make the specified crawl store join all crawls that it is not already part of. For every crawl that the crawl store needs to join, the stored procedure MUST add an entry to the crawl store status for the specified crawl store, and to the crawl component status for each crawl component of the specified crawl store. For the specification of Crawl Store Status see Section 3.1.1.28. For the specification of Crawl Component Status see Section 3.1.1.25.

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

```
PROCEDURE proc_MSS_CrawlStoreToJoinAllOngoingCrawls (
    @GthrDBID                int
);
```

@GthrDBID: The ordinal of the crawl store from Crawl Stores Set ([MS-SRCHTP] section 3.1.1.3).

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

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

3.1.5.10 proc_MSS_CreateNewComponentRecord

If the Component Activity Set, as specified in section 3.1.1.35, contains a record with ComponentName equal to @ComponentName and ProjectName equal to ProjectName, the stored procedure MUST set CPR_Crawls to DPR_Crawls and CPR_MasterMerge to DPR_MasterMerge for this record.

If the preceding record does not exist this stored procedure MUST add a record to the Component Activity Set using the following values:

- ComponentName, ProjectName, ComponentType and ComponentID of the created record in the Component Activity Set MUST match the values for @ComponentName, @ProjectName, @ComponentType and @ComponentID that were passed in.
- DPR_MasterMerge and DPR_Crawls as specified in section 2.2.2.1 MUST be set to 0 if there is no other record in Component Activity Set with the same ProjectName, otherwise DPR_MasterMerge and DPR_Crawls MUST be set to the same value as the other records in this set which have the same ProjectName.
- CPR_MasterMerge and CPR_Crawls MUST be set to DPR_MasterMerge and DPR_Crawls correspondingly.

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

```
PROCEDURE proc_MSS_CreateNewComponentRecord(
    @ComponentName          nvarchar(100),
    @ProjectName            nvarchar(100),
    @ComponentType          int,
    @ComponentId            int
);
```

@ComponentName: Name of the crawl component or query component (2) that MUST have up to a maximum of 100 Unicode characters

@ProjectName: Name of the project. As specified in section 2.2.1.19.

@ComponentType: Type of the component as specified in [MS-SRCHTP] section 3.1.1.3

@ComponentID: If @ComponentType is not equal to 0 or 3 this value MUST contain identifier of the component otherwise this value MUST be ignored.

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

Result Sets: SHOULD NOT<5> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.5.11 proc_MSS_CreateConfigTransaction

Search components call this stored procedure to create and obtain a unique TransactionID.

This stored procedure creates new record in Uncommitted Transactions of Configuration Properties set (section 3.1.1.36). The value of TransactionID of the newly created record MUST be set to an ever increasing integer.

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

```
PROCEDURE proc_MSS_CreateConfigTransaction ();
```

Return code values: MUST return TransactionID of the created record

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

3.1.5.12 proc_MSS_DeleteConfigurationProperties

The **proc_MSS_DeleteConfigurationProperties** stored procedure is called to delete all configuration properties which names match the specified pattern. The list of property names MUST be used to avoid deleting some properties even if they match the pattern.

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

```
PROCEDURE proc_MSS_DeleteConfigurationProperties (
    @Pattern                nvarchar(300),
    @PropertiesToKeep       nvarchar(1024)
);
```

@Pattern: The pattern that MUST be matched against all configuration property names according to the configuration properties wildcard rules defined by:

- The "%" character is used as a wildcard
- The wildcard MUST only be used in either or both of the following positions:
  - the beginning of the pattern string
  - the end of the pattern string

@PropertiesToKeep: The list of the configuration property names that MUST NOT be deleted by the given pattern. This list MUST be in the following format:

,Column1[,Column2...],

This list MUST begin with a comma and MUST also end with a comma. If there are no columns to keep, this parameter MUST be passed as NULL.

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

Result Sets: SHOULD NOT<7> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.5.13 proc_MSS_DeleteConfigurationProperty

The **proc_MSS_DeleteConfigurationProperty** stored procedure is called to delete the specified configuration property by its name.

```
PROCEDURE proc_MSS_DeleteConfigurationProperty (
    @Name                   nvarchar(300)
);
```

@Name: The name of the configuration property.

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

Result Sets: SHOULD NOT<8> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.5.14 proc_MSS_FlushTemp0

The **proc_MSS_FlushTemp0** stored procedure is called to process the next set of items within a given **crawl.** This stored procedure called during the Process Links step in the Crawl Data Flow Diagram (section 3.2.5.1)

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

```
PROCEDURE proc_MSS_FlushTemp0(
    @ComponentID            int,
    @FlushProjectID         int,
    @FlushCrawlID           int,
    @LogDiscoveredLinks     int,
    @ApplicationType        int,
    @MoreLinks              BIT OUTPUT,
    @NextDocID              int,
    @MaxDocID               int OUTPUT,
    @LinksProcessed         int OUTPUT
);
```

@ComponentID: The unique identifier of the crawl component. The crawl component MUST have a Crawl Component State (section 2.2.1.12) of 1 or 4.

@FlushProjectID: The project identifier as specified in Project Identifier (section 2.2.1.1). Based on the value of this parameter the store procedure performs specific actions which are specified in the following table.

@FlushCrawlID: The unique identifier of the crawl.

@LogDiscoveredLinks: MUST be ignored

@ApplicationType: MUST be 1

@MoreLinks:

- If value of @FlushProjectID is 2 (Anchor Project), @MoreLinks MUST be set to 0
- If value of @FlushProjectID is 1 (Portal Content), upon return from the stored procedure, this flag MUST be set if 1000 links were processed. The flag MUST NOT be set if less than 1000 links were processed, which indicates that no more links remain to be processed.

@NextDocID: If the previous call to the stored procedure failed with insufficient document identifiers (return value 2) then this parameter is the beginning of the new interval of document identifiers. Otherwise this value MUST be 0.

@MaxDocID: If the previous call to the stored procedure failed with insufficient document identifiers (return value 2) then this parameter is the end of the new interval of document identifiers. Otherwise this value MUST be 0.

On output, **@MaxDocID** must be set only if the stored procedure failed because number of identifiers was insufficient to process batch of links (return value 2):

- If there are documents in the current crawl store then the stored procedure MUST set **@MaxDocID** to maximum identifier of the document in this crawl store
- If there are no documents in the current crawl store the stored procedure MUST set **@MaxDocID** to 0

If the stored procedure succeeds then value of the output parameter MUST be ignored.

@LinksProcessed: Upon return from this stored procedure, this parameter must be set to the number of links processed in the current batch.

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

Value	Description
0	The stored procedure failed. No changes were persisted.
1	Successful execution.
2	The stored procedure failed because the number of item identifiers in the range defined by @NextDocID and @MaxDocID was insufficient for processing all the new items included in the current batch. No changes were persisted.

Result Set: MUST not return any result set.

If value of the input parameters @NextDocID and @MaxDocID is not equal 0 the stored procedure MUST update Current DocID Chunk (section 3.1.1.18): it MUST set value of the field FirstDocID to @NextDocID, LastDocID to @MaxDocID.

The following table describes the behavior of the stored procedure for each possible value of @FlushProjectID:

Value of @FlushProjectID	Description
2 (Anchor Project)	The stored procedure MUST process all links from Links Set (3.1.1.5). For every record in this set the stored procedure MUST perform the following actions: ▪ The stored procedure MUST assign a new DocID to the link. The stored procedure MUST first use the item identifiers of the previously deleted documents. When all the item identifiers previously used were exhausted the stored procedure MUST use new item identifiers from the range defined by FirstDocID, LastDocID from Current DocID Chunk (section 3.1.1.18) ▪ If the stored procedure used DocIDs from the range preceding, FirstDocID in Current DocID Chunk (section 3.1.1.18) MUST be updated with the maximal value of used DocID plus 1. ▪ The stored procedure MUST insert a new entry into Crawl Queue with values

Value of @FlushProjectID	Description
	▪ CrawlID, StartAddressID, ContentSourceID, ProjectID, DocID, Scope, TransactionFlags, HostDepth, EnumerationDepth, ChangeLogBatchID, and CachedBlob equal to the corresponding fields of the link. ▪ TransactionType equal to Modify. ▪ SourceDocID equal to the ParentDocID of the record from the URL History Set, with the values CompactHash and CompactURL equal to corresponding values of the link and with the value DeletePending equal to "0". ▪ The stored procedure MUST insert a new entry into the Crawl URL History with the following values: ▪ StartAddressID, ContentSourceID, ProjectID, DocID, CrawlID, AccessURL, AccessHash, CompactURL, CompactHash, DisplayURL, DisplayHash, TransactionFlags, HostDepth, EnumerationDepth, UseChangeLog, IndexType, LCID, PropMD5, EndPathFlag, HostID, and SiteID equal to corresponding fields of the link. ▪ ParentDocID equal to SourceDocID of the link. ▪ ParentHostID equal to SourceHostID of the link. ▪ ErrorID equal to "0". ▪ ErrorLevel equal to "0". ▪ CrawlScope equal to "2" (Deep). ▪ PHFlags equal to "0". ▪ SecurityID equal to NULL.
1 (Portal Project)	The stored procedure MUST set @MoreLinks to 1 if there are more than 1000 elements in Links Set (3.1.1.5) with field CrawlID equal to input parameter @FlushCrawlID. The stored procedure MUST select up to 1000 elements from the Links Set in which the field CrawlID is equal to the input parameter @FlushCrawlID. This set is referred as LinksToProcess. For every link from LinksToProcess that has no record in the Crawled Host Set, as specified in section 3.1.1.7, with the value of the field HostID equal to the value of the field HostID of the link, the stored procedure MUST insert a new record into the Crawled Host Set with the following values: ▪ HostID set to HostID of the link. ▪ HostName set to Host of the link. ▪ SuccessCount, ErrorCount, WarningCount, DeleteCount, and LevelHighErrorCount set to "0". For each link in LinksToProcess, the stored procedure MUST select a record, if one exists, from the Crawl URL History set, referred to as a processed record, in which the value of the field AccessURL, AccessHash of the entry is equal to the corresponding fields of the processed record and the processed record has no delete requests, which means that the value of DeletePending is "0". The stored procedure MUST mark records from LinksToProcess as processed if they

Value of @FlushProjectID	Description
	satisfy the flowing condition: ▪ value of field ItemType is Link ▪ corresponding processed record exists and values of field DocID of processed record is not -1 ▪ value of field CrawlID of processed record is greater than value of field @FlushCrawlID For every link in LinksToProcess which satisfies the following conditions: ▪ value of field ItemType is not Anchor Link ▪ value of field ItemType is not Last Link ▪ corresponding processed record exists and value of the field DocID of the processed record is not -1 ▪ value of field CrawlType is Full Crawl ▪ the link has no errors (value of hrResult is 0) ▪ The link is not a duplicate of another link from LinksToProcess. In other words, there does not exist another link in LinksToProcess (referred to as another link) in which: ▪ The value of AccessHash of the link equals AccessHash of another link. ▪ The value of AccessURL of the link equals AccessURL of another link. ▪ The value of LinkID of the link does not equal LinkID of another link. ▪ The field StartAddressID of the processed record is not "0" and equals the corresponding field of the link. the stored procedure MUST insert a new entry into Crawl Queue (3.1.1.4) with values ▪ CrawlID, StartAddressID, ContentSourceID, ProjectID, DocID, Scope, TransactionFlags, HostDepth, EnumerationDepth, SourceDocID, ChangeLogBatchID, CachedBlob, UserSecurityInfo equal to corresponding fields of the link ▪ TransactionType equals 2 For each link matching the condition preceding, the stored procedure MUST set CrawlID to @FlushCrawl in Crawl URL History for all records which have the value of DocID equal to the value of field DocID of the link. The stored procedure MUST mark all links matching the condition preceding as processed. The stored procedure MUST set field DocID for all records which satisfy the following conditions to the new value. The stored procedure MUST use DocIDs of the deleted documents as well as DocIDs from the range defined by FirstDocID, LastDocID from Current DocID Chunk (section 3.1.1.18). If the stored procedure used DocIDs from the range preceding, FirstDocID in

Value of @FlushProjectID	Description

	Current DocID Chunk (section 3.1.1.18) MUST be updated with the maximal value of used DocID plus 1.
	If there are not enough DocIDs the stored procedure MUST return 2 (which means need more DocIDs).
	For all unprocessed records in LinksToProcess, in which the following conditions are satisfied:
	▪ The value of the field ItemType is "Link".
	▪ The link has no errors, which means that the value of hrResult is "0".
	▪ TransactionType is either "Add item" or "Modify item".
	▪ The value of ParentProcessChangeLog is "0".
	▪ The corresponding processed record does not exist.
	▪ The link is not a duplicate of another link from LinksToProcess, which means that it does not exist as another link in LinksToProcess, in which:
	  ▪ The value of AccessHash of the link is equal to the AccessHash of another link.
	  ▪ The value of AccessURL of the link is equal to the AccessURL of another link.
	  ▪ The value of LinkID of the link does not equal the LinkID of another link.
	The stored procedure MUST
	▪ set field DocID to the new value. The stored procedure MUST use DocIDs of the deleted documents as well as DocIDs from the range defined by input parameters @NextDocID, @MaxDocID. If there are not enough DocIDs the stored procedure MUST return 2 (which means need more DocIDs)
	▪ insert a new entry into Crawl URL History Set with values
	  ▪ StartAddressID, ContentSourceID, ProjectID, DocID, CrawlID, AccessURL, AccessHash, CompactURL, CompactHash, DisplayURL, DisplayHash, TransactionFlags, HostDepth, EnumerationDepth, UseChangeLog, IndexType, LCID, PropMD5, EndPathFlag, HostID, SiteID equal to corresponding fields of the record
	  ▪ ParentDocID equal to the SourceDocID of the record.
	  ▪ ErrorID, ErrorLevel equal to "0".
	  ▪ ParentHostId equal to the SourceHostID of the record
	▪ insert a new entry into Crawl Queue with values
	  ▪ CrawlID, StartAddressID, ContentSourceID, ProjectID, DocID, TransactionFlags, HostDepth, EnumerationDepth, SourceDocID, ChangeLogBatchID, and CachedBlob equal to corresponding fields of the record.

Value of @FlushProjectID	Description
	▪ TransactionType equal to Modify. ▪ Scope equal to "2" (Deep). ▪ mark record from LinksToProcess as processed For each unprocessed link in LinksToProcess the stored procedure MUST do the following: If value of ItemType of the link is Link, and value of TransactionType of the link is Rename, and processed record does not exist, the stored procedure MUST mark the link as processed If value of ItemType of the link is Link, and value of TransactionType of the link is Rename, and processed record exists, and values AccessHash and AccessURL of processed record equal to the corresponding values of the link, the stored procedure MUST set TransactionType to Modify and Scope to 2 (Deep) If the following conditions hold: ▪ value of ItemType of the link is Link ▪ value of TransactionType of the link is Rename ▪ processed record exists ▪ either value AccessHash of processed record does not equal to AccessHash of the link or AccessURL of processed record does not equal to AccessURL of the link the stored procedure MUST perform the following 12 steps first with @TransactionType equals Delete and then with @TransactionType equals Add up to the second stage defined later. The following steps MUST be performed: 1. If the following conditions are satisfied, the stored procedure MUST mark the link as processed. 2. value ItemType of the link is Link 3. value of TransactionType of the link is Delete 4. processed record does not exist 5. If the following conditions are satisfied, the stored procedure MUST mark the link as processed. ▪ value of @ProjectID is 1 (Portal Content) ▪ processed record exists ▪ value of ItemType of the link is Start Address ▪ value of CrawlType of the link is Incremental Crawl 6. If the following conditions are satisfied, the stored procedure MUST mark the link as processed.

Value of @FlushProjectID	Description
	▪ processed record exists ▪ CrawlID field from processed record is greater than @FlushCrawl ▪ value of ChangeLogBatchID of the link is 0 7. If the following conditions are satisfied, the stored procedure MUST mark the link as processed. 8. processed record exists 9. value of ChangeLogBatchID of the link is not 0 10. Item is Link or Shallow Link 11. TransactionType is either Modify or Add 12. value of CommitCrawlID of processed record is not 0 13. If the following conditions are satisfied, the stored procedure MUST mark the link as processed. 14. processed record exists 15. value of ChangeLogBatchID of the link is not 0 16. CrawlID field from processed record is greater that @FlushCrawl 17. value of CommitCrawlID of processed record is not 0 18. value of CrawlScope of processed record is 2 (Deep) 19. If the following conditions are satisfied, the stored procedure MUST mark the link as processed. 20. processed record exists 21. CrawlID field from processed record is greater than @FlushCrawl 22. value of ChangeLogBatchID of the link is not 0 23. value of TransactionFlags of the processed record has flag 0x0200 set (this link is a file inside a smart folder) 24. value of TransactionFlags of processed record does not have the flag 0x0004 set (this link is a folder) 25. If processed record exists and field StartAddress of the link does not equal to corresponding field from processed record and either of the following conditions is satisfied, the stored procedure MUST mark the link as processed. 26. field ParentDocID from processed record is -1 and value ItemType of the link is Start Address and value of ChangeBatchID of the link is 0 27. value of SourceDocID of the link is not -1 and value of HostID of the

Value of @FlushProjectID	Description
	link not equal to value of SourceHostID of the link 28. value of SourceDocID of the link is not -1 and value of ProtocolSwitch of the link is 1 and value of SourceIsStartAddress of the link is not 1 and value of SourceHostHop of the link is 1 29. If the link satisfied the following conditions, the stored procedure MUST set CrawlID and CommitCrawlID to @FlushCrawl, ErrorID to Not Modified, and ErrorLevel to Not Modified in the Crawl URL History for the record with the DocID equal to the DocID of the link. Then the stored procedure MUST mark the link as processed. ▪ value of CrawlType of the link is Incremental Crawl ▪ processed record exists ▪ value of UseChangeLog of the link is 0 ▪ @FlushCrawl is greater that the value of CrawlID of processed record ▪ TransactionFlags has flag 0x0200 set (this link is a file inside smart folder) ▪ TransactionFlags does not have flag 0x2000000 set (this link is not for refreshing security info) ▪ The value of LastModifiedTime of the link equals the value of corresponding field in the processed record. 30. If the link satisfied the following conditions, the stored procedure MUST insert a new record into the Crawl Queue with the following values: ▪ CrawlID, StartAddressID, DocID, ChangeLogBatchID, ContentSourceID, and ProjectID equal to the corresponding fields of the link. ▪ HostDepth, EnumerationDepth, ParentDocID set to corresponding fields of the processed record. ▪ TransactionType equal to Modify. ▪ Scope equal to "2" (Deep). ▪ TransactionFlags equal to the corresponding value of the processed record with the bit corresponding to the mask 0x2000000 also set. Conditions: ▪ value ChangeLogBatchID of the link is greater than 0 ▪ value TransactionFlags of the link has the flag 0x2000000 set ▪ value of SecurityUpdateCrawlID of processed record is less than @FlushCrawl ▪ either value TransactionFlags of the link does not have the flag 0x200000 set or value of CachedSecurityUpdateCrawlID from processed record is less than @FlushCrawl If additionally value of TransactionFlags of the link does not have the flag 0x200000 set, the stored procedure MUST set

Value of @FlushProjectID	Description
	▪ SecurityUpdateCrawlID to @FlushCrawl ▪ SecurityID to NULL in the record of the Crawl URL History Set for which the value of DocID equals the value of the DocID of the link; otherwise the stored procedure MUST set CachedSecurityUpdateCrawlID to @FlushCrawl in the record of Crawl URL History Set for which the value of DocID equals the value of DocID of the link. The stored procedure MUST mark the link as processed. 1. If all of the following conditions hold: ▪ value of UseChangeLog of the link is greater than 0 ▪ value of CommitCrawlID of processed record is greater than 0 ▪ value of PropMD5 of the link does not equal to the one from processed record the stored procedure MUST change the value of the filed @Scope of the link to 2 (Deep) 2. If TransactionType is not Delete and there are no errors for this link (hrResult is 0) the stored procedure MUST: 3. if value of UseChangeLog of the link is 1 and value of ParentProcessChangeLog of the link is 1 the stored procedure MUST mark link as processed and go to the next link 4. if value of UseChangeLog of the link is 1 and processed record exists and ParentDocID of processed record is not NULL the stored procedure MUST set ParentDocID of the link to ParentDocID of processed record 5. if value of DocID of the link is NULL the stored procedure MUST assign new document identifier to the link using first the document identifiers of the previously deleted documents. When all the item identifiers previously used were exhausted the stored procedure MUST use values from the range defined by FirstDocID, LastDocID from Current DocID Chunk (section 3.1.1.18). If the store procedure used DocIDs from the range preceding, FirstDocID in Current DocID Chunk (section 3.1.1.18) MUST be updated with the maximal value of used DocID plus 1. 6. @NextDocID, @MaxDocID. If the stored procedure is unable to assign item identifier it MUST roll back all the changes and return 2. Otherwise, the stored procedure MUST insert new record in Crawl URL History Set 7. if value of DocID is not NULL the stored procedure MUST set 8. CrawlID to @FlushCrawl; 9. ParentDocID, StartAddressID, ContentSourceID to values of corresponding fields of the link; 10.ParentHostID to the value of the field SourceHostID of the link in the

Value of @FlushProjectID	Description
	Crawl URL History Set for the record with DocID equal to the DocID of the link 11.the stored procedure MUST insert a new record in Crawl URL History Set with: 12.CrawlID, StartAddressID, ContentSourceID, ProjectID, DocID, TransactionFlags, HostDepth, EnumerationDepth, ChangeLogBatchID and CachedBlob, equal to corresponding values of the link 13.Scope equal to "2" (Deep). 14.TransactionType equal to Modify. 15.SourceDocID equal to the ParentDocID of the link. 16.If the following conditions are satisfied 17.There are errors associated with the link (hrResult is not 0) 18.either value DocID of the link is NULL or value of the field StartAddress of the link does not equal to corresponding value of processed record 19.there is no record in the Crawl Deleted URL Set with AccessURL and AccessURL equal to the corresponding values of the link. the stored procedure MUST insert a new record into Crawl Deleted URL Set The stored procedure MUST increase the DeleteCount counter by 1 in the Crawled Host List Set for the record where the value of HostID equals the one of the link. The stored procedure MUST insert a record into the Crawl URL Changes Set if the value of AccessURL of the link does not contain 'anchor:' 20.If the following conditions are satisfied 21.There are errors associated with the link (hrResult is not 0) 22.The value of DocID of the link is not NULL. 23.The value of the field StartAddress of the link equals the corresponding value of the processed record. The stored procedure MUST insert a record into the Crawl Queue with TransactionType set to Delete, Scope set to "2" (Deep), and DeleteReason set to Delete Excluded. 24.If the following conditions are satisfied 25.value of ChangeLogBatchId is greater than 0 26.value of CrawlID is greater than @FlushCrawl

Value of @FlushProjectID	Description
	27.TransactionType is Delete the stored procedure MUST insert record into Crawl Queue with TransactionType set to Delete; Scope set to 2 (Deep); DeleteReason set to Delete Change Log The stored procedure starts second stage of processing all links from LinksToProcess. The stored procedure MUST mark all of links in the LinksToProcess as not processed. The stored procedure MUST mark all records from LinksToProcess as processed if any of the following is true: ▪ value of ParentProcessChangeLog of the link is 1 ▪ value of ItemType of link is Last Link ▪ value of ItemType of link is Start Address ▪ value of DisplayURL of the link starts with 'bdc' (the data represented by this link was crawled by the **Business Data Connectivity (BDC)**) ▪ hrResult of the link equals 0x80040D07 (URL Excluded because of crawl rule). The stored procedure MUST set CrawlID to @FlushCrawlID, AnchorText to the value of AnchorText of the link, AnchorHash to the value of AnchorHash of the link in Anchor Text Info Set for every record which satisfies the following conditions: there is a link in LinksToProcess in which the following are true: ▪ The link is not yet processed. ▪ The value of LinkID and SourceDocID of the link equal the corresponding values of the record. ▪ The value of FirstLink of the link is equal to the LinkOrdinal of the record. ▪ The value of the DisplayHash of the link equals the value of the LinkHash of the record. ▪ The value of the DisplayURL of the link equals the value of the Link of the record. ▪ The value of the AnchorHash of the link equals the corresponding value of the record. ▪ Either the value of the AnchorHash of the link is "0" or the value of AnchorText of the link equals the corresponding value of the record. the stored procedure MUST mark all links corresponding to the updated records as processed. The stored procedure MUST set CrawlID to @FlushCrawlID in the Anchor Text Info Set for every record which satisfies the following conditions: there is a link in LinksToProcess in which

Value of @FlushProjectID	Description
	▪ the link is not yet processed ▪ value of SourceDocID of link equal to corresponding values of the record ▪ value of FirstLink of link equal to LinkOrdinal of the record ▪ value of DisplayHash of the link equals the value of the LinkHash of the record. ▪ value of DisplayURL of link equals the value of the Link of the record. the stored procedure MUST mark all links corresponding to updated records as processed the stored procedure MUST insert a new record into Changed Anchor Target Documents Set with value CrawlID equals @FlushCrawlID and value DocID equals TargetDocID of the record for all updated links from Anchor Text Info Set, for which TargetDocId is not -1 The stored procedure MUST insert into Anchor Text Info Set a new record with ▪ values SourceDocID, Link, LinkHash, LCID, AnchorText, AnchorHash, LinkOrdinal, Pid, SourceDocSiteID, InterSite set to SourceDocID, DisplayURL, DisplayHash, LCID, AnchorText, AnchorHash, FirstLink, Pid, SiteID, InterSite of global link ▪ value CrawlID set to @FlushCrawlID ▪ TargetDocID set to value of DocID of the link for every not processed link from LinksToProcess The stored procedure MUST insert into Change Anchor Target Documents a record with CrawlID equals @FlushCrawlID, DocID equals DocID of the link for every not processed link from LinksToProcess with value DocID not equal -1 The stored procedure deletes from the Anchor Text Info Set every record that has a corresponding link in LinksToProcess in which: ▪ The values of SourceDocID in the records equal the one of the link. ▪ The values of ItemType of the link equal Last Link. ▪ The value of ParentProcessChangeLog of the link equals "0". and the value of CrawlID of the records is less than @FlushCrawlID. The stored procedure MUST insert a record into Changed Anchor Target Documents Set with values CrawlID equals @FlushCrawlID, DocID equals DocID TargetDocID of the record for every deleted record. After all the links are processed the stored procedure MSUT perform the following actions: The stored procedure MUST remove from Links Set all links included in LinksToProcess. The stored procedure MUST copy into a set up to 5,000 links from the Colleague Links Set, as specified in section 3.1.1.6, in which the value of CrawlID equals @FlushCrawlID. This set is referred to as ColleagueLinksToProcess. If the stored procedure was able to copy 5,000 links, it MUST set @MoreLinks to "1".

Value of @FlushProjectID	Description
	If a link from ColleagueLinksToProcess has a link from the Social Distance Property Set specified in section 3.1.1.11, which is referred to as social link, in which values of SourceDocID and TargetDocID of the link equal to corresponding values of the social link, this link is referred to as a known link. For every link in ColleagueLinksToProcess in which: ▪ The link is a known link. ▪ The value of Pid of the link equals the value of Pid of the social link. The stored procedure MUST insert a new entry into the Social Distance Unchanged Property Set, with values set as follows: ▪ CrawlID equals @FlushCrawl. ▪ SourceDocID and TargetDocID equal the corresponding values of the social link For every link in ColleagueLinksToProcess in which: ▪ The value of Pid of the link is not equal to "-1". ▪ The value of TargetDocID is null. The stored procedure MUST insert a new entry into the Social Distance New Property Set, as specified in 3.1.1.13, with the following values: ▪ CrawlID equals @FlushCrawlID. ▪ SourceDocID and TargetDocID equal the corresponding values of the social link. For every record in ColleagueLinksToProcess in which: ▪ The link is a known link. ▪ The value of Pid of the record does not equal the value of Pid of the social link ▪ The value of Pid of the record does not equal "-1". The stored procedure MUST set the Pid of the social link to the value in the Pid of the record. The stored procedure MUST insert a record into Social Distance New Properties Set a record with CrawlID equals @FlushCrawlID, SourceDocID, TargetDocID to the corresponding values of the record for every updated record. The stored procedure MUST remove every record from Social Distance Set which satisfied the following conditions: ▪ there is a link in ColleagueLinksToProcess with SourceDocID of the record equal to the SourceDocID of the link and Pid of the link is -1 and ▪ there are no records in the Social Distance Unchanged Property Set (referred to as an unchanged record) in which the value of CrawlID of the unchanged record equals the @FlushCrawlID and the value of SourceDocID and TargetDocID of unchanged record equals the

Value of @FlushProjectID	Description
	corresponding values of the record ▪ there are no records in the Social Distance Deleted Properties Set, referred to as a new record, in which the value of CrawlID of the new record equals @FlushCrawlID and the value of the SourceDocID and the TargetDocID of the new record equals the corresponding values of the record. For every record in ColleagueLinksToProcess in which the: ▪ corresponding social link exists ▪ value of TargetDocID of social link is not NULL ▪ value of Pid of the record is not -1 the stored procedure MUST insert a new entry into the Social Distance Set with the values SourceDocID, SourceUserID, TargetUserID, and Pid equal to the corresponding values of the record. The stored procedure MUST insert into the UserID to DocID Pair Set, as specified in section 3.1.1.24, a new pair with the UserID equal to the SourceUserID of the record and DocID equal to the SourceDocID of the record for every record from the ColleagueLinksToProcess in which the value Pid of the record is "-1" and there is a pair in the UserID to DocID Pair Set in which the UserID of the pair equals the SourceUserID of the record. It is possible that there are multiple records with the same SourceUserID and with the Pid equal to "-1". In that case, the stored procedure MUST use only the record with the minimal SourceDocID. The stored procedure MUST remove from the Colleague Links Set all links corresponding to the ColleagueLinksToProcess.

3.1.5.15 proc_MSS_GetAnchorDocIDs

The **proc_MSS_GetAnchorDocIDs** stored procedure MUST return all document identifiers (1) from Anchor Change Set (section 3.1.1.14)

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

```
PROCEDURE proc_MSS_GetAnchorDocIDs();
```

Return Code Values: An integer which MUST be ignored.

Result Set: MUST return the following result set:

3.1.5.15.1 GetAnchorDocIDs Result Set

The result set MUST contain zero or more rows which are ordered by TargetDocID.

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

```
TargetDocID                int;
```

TargetDocID: A unique identifier of the document.

3.1.5.16 proc_MSS_GetCompiledScopeRules

The **proc_MSS_GetCompiledScopeRules** stored procedure is called to retrieve all the search scope rules of search scopes that are compiled. The stored procedure MUST return all data in the Compiled Search Scope Rules Set defined in Section 3.1.1.34.

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

```
PROCEDURE proc_MSS_GetCompiledScopeRules();
```

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

Result Sets: MUST return the following result set:

3.1.5.16.1 Compiled Scopes Result Set

See Scope Compilation Result Set (section 2.2.4.1).

3.1.5.17 proc_MSS_GetCompiledScopes

The **proc_MSS_GetCompiledScopes** stored procedure is called to retrieve the information about all search scopes that have undergone search scope compilation. The stored procedure MUST return all data in the Compiled Search Scopes Set defined in Section 3.1.1.33.

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

```
PROCEDURE proc_MSS_GetCompiledScopes();
```

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

Result Sets: MUST return the following result set:

3.1.5.17.1 Scopes Result Set

See Scope Compilation Result Set (Section 2.2.4.1).

3.1.5.18 proc_MSS_GetCompletedScopesCompilationId

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

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

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

@ComponentRole: Type of the component. The value MUST be a valid Component Type defined in [MS-SRCHTP] section 2.2.1.12.

@CompilationID: Upon return from this stored procedure, this parameter MUST be set to the CompilationChangeID of the Scopes System Set if the component role is a crawl component of type 2, otherwise it MUST be set to the LastCompilationID of the Scopes System Set. For the specification of the Scopes System Set see [MS-SQLPADM2] Section 3.1.1.4.

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

Result Sets: MUST NOT return any result set.

3.1.5.19 proc_MSS_GetComponentStatusUpToDate

The **proc_MSS_GetComponentStatusUpToDate** stored procedure is called to retrieve a judgment about whether or not the given component has failed to complete any component activity (section 3.1.1.35).

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

```
PROCEDURE proc_MSS_GetComponentStatusUpToDate(
    @ComponentType                int,
    @ComponentNumber              int,
    @ProjectName                  nvarchar(100),
    @UpToDate                     int OUTPUT
);
```

@ComponentType: The component type ([MS-SRCHTP] section 2.2.1.12) of the component for which the judgment is being requested. The value MUST be 1 or 2.

@ComponentNumber: If *@ComponentType* is 1, the CrawlComponentNumber ([MS-SRCHTP] section 3.1.1.3) of the crawl component for which the judgment is being requested. If *@ComponentType* is 2, the QueryComponentNumber ([MS-SRCHTP] section 3.1.1.3) of the query component (2) for which the judgment is being requested.

@ProjectName: The name of the full-text index catalog for which the judgment is requested. The value MUST be "Portal_Content" or "AnchorProject".

@UpToDate: Upon return from this stored procedure, this parameter MUST be set to 0 if the component activity set (section 3.1.1.35) contains any component activity for which:

1. the value of ComponentType (section 3.1.1.35) is *@ComponentType*, the value of ComponentNumber (section 3.1.1.35) is *@ComponentNumber*,
2. the value of ProjectName (section 3.1.1.35) is *@ProjectName*, and
3. any of the following are true:
    1. the value of HaveResetRequest (section 3.1.1.35) is not 0,
    2. the value of CPR_MasterMerge (section 3.1.1.35) is not the value DPR_MasterMerge (section 3.1.1.35), or
    3. the value of CPR_Crawls (section 3.1.1.35) is not the value of DPR_Crawls (section 3.1.1.35).

Otherwise, this parameter MUST be set to 1.

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

Result Sets: MUST NOT return any result set.

3.1.5.20 proc_MSS_GetConfigurationProperties

The **proc_MSS_GetConfigurationProperties** stored procedure is called to retrieve a list of all configuration properties which names match specified pattern.

```
PROCEDURE proc_MSS_GetConfigurationProperties (
    @Pattern                nvarchar(300)
);
```

@Pattern: The pattern that all configuration property names MUST be matched against according to the configuration properties wildcard rules defined by:

- The "%" character is used as a wildcard
- The wildcard MUST only be used in either or both of the following positions:
  - at the beginning of the pattern string
  - at the end of the pattern string

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

Result Sets: This procedure MUST return the following result set:

3.1.5.20.1 Configuration Properties Result Set

The Configuration Properties result set MUST contain zero or more rows, each corresponding to a single configuration property.

The first two columns of the result set MUST be as follows:

```
Name                    nvarchar(300),
Value                   sql_variant;
```

All other columns MUST be ignored.

Name: The name of the configuration property.

Value: The value of the configuration property.

3.1.5.21 proc_MSS_GetConfigurationProperty

The **proc_MSS_GetConfigurationProperty** stored procedure is called to retrieve the value of the specified configuration property by its name.

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

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

@Name: The name of the configuration property.

@Value: Upon return from this stored procedure, this parameter MUST be set to the value of the configuration property. If the property with the specified name doesn't exist then the value of this parameter MUST be set as NULL.

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

Result Sets: SHOULD NOT<9> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.5.22 proc_MSS_GetContentSourceDocCount

The proc_MSS_GetContentSourceDocCount stored procedure is called to retrieve crawl statistics for correspondent content source. These values are contained within the Crawl Component Status as specified in section 3.1.1.25.

```
The T-SQL syntax for the stored procedure is as follows:
PROCEDURE proc_MSS_GetContentSourceDocCount (
    @ContentSourceID          int,
    @SuccessCount             int OUTPUT,
    @WarningCount             int OUTPUT,
    @ErrorCount               int OUTPUT,
    @DeleteCount              int OUTPUT,
    @LevelHighErrorCount      int OUTPUT
);
```

@ContentSourceID: An integer that identifies the content source.

@SuccessCount: Upon return from this stored procedure, this parameter MUST be set to number of successes for the content source.

@WarningCount: Upon return from this stored procedure, this parameter MUST be set to number of warnings for the content source

@ErrorCount: Upon return from this stored procedure, this parameter MUST be set to number of errors for the content source.

@DeleteCount: Upon return from this stored procedure, this parameter MUST be set to number of deletes for the content source.

@LevelHighErrorCount: Upon return from this stored procedure, this parameter MUST be set to number of level high errors for the content source.

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

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

3.1.5.23 proc_MSS_GetCrawledProperties

The **proc_MSS_GetCrawledProperties** stored procedure is called to retrieve the list of crawled properties from the **Crawled Property Set**, as specified in [MS-SQLPADM2] section 3.1.1.1, that meet the constraints imposed by the input parameters.

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

```
PROCEDURE proc_MSS_GetCrawledProperties {
```

```
    @OOBOnly                        bit,
    @InterestingPropertiesOnly      bit,
    @UpDate                         datetime,
    @LimitRows                      int,
    @Propset                        uniqueidentifier,
    @PropertyName                   nvarchar(440)
};
```

@OOBOnly: This value MUST be 1 if only the out-of-box crawled properties are to be retrieved. Otherwise, it MUST be 0.

@InterestingPropertiesOnly: This parameter MUST be ignored by the client

@UpDate: If this value is NOT NULL, the stored procedure MUST only return crawled properties that have their LastModifiedTime, as specified in [MS-SQLPADM2] section 3.1.1.1, equal to or greater than this value.

@LimitRows: If this value is NOT 0, it limits the maximum count of rows returned by the stored procedure. This value MUST NOT be NULL.

@Propset: The crawled property set identifier associated with the crawled property category. If @UpDate is NULL, then the stored procedure MUST return crawled properties with their crawled property set identifier greater than or equal to this value.

@PropertyName: The name of the crawled property. If @UpDate is NULL, then the stored procedure MUST return crawled properties with their property name greater than this value.

Return Value: This stored procedure returns an integer value that MUST be ignored.

Result Sets: MUST return the result set:

3.1.5.23.1 GetCrawledProperties Result Set

The **GetCrawledProperties** result set contains information about crawled properties. 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),
LastModified                datetime;
```

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

IsMappedToContent: This value MUST be set to 1 if the variant type (2) 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: This value MUST be set to 1 if the sample crawled properties set is complete. Otherwise, it MUST be set to 0.

VariantType: The variant type (2) for the crawled property. This value MUST NOT be NULL.

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

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

LastModified: The date and time of the last change to the crawled property

3.1.5.24 proc_MSS_GetCrawledPropMappingUpdates

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

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

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

@UpDate: The earliest last modified datetime a **search property mapping** MUST have to be included in the result set.

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

Result Sets: MUST return the following result set:

3.1.5.24.1 GetCrawledPropertyMappingUpdates Result Set

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

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

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

pid: The unique identifier for the managed property.

mappingorder: An integer representing the mapping order of crawled property to a managed property.

crawledPropertyid: The unique identifier for the crawled property.

3.1.5.25 proc_MSS_GetCrawls

The **proc_MSS_GetCrawls** stored procedure retrieves the list of the current crawls for an crawl component based on the Crawl Status structure see section 3.1.1.1 and the Crawl Component Status structure see section 3.1.1.25.

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

```
PROCEDURE proc_MSS_GetCrawls (
    @ComponentID int,
    @CatalogID int,
    @MasterRole bit
);
```

@ComponentID: If MasterRole is 0, ComponentID is the identifier of the crawl component found in the Crawl Component Status section 3.1.1.25, if MasterRole is 1, ComponentID MUST be ignored.

@CatalogID: Project Identifier as specified in section 2.2.1.1 for which crawls are returned.

@MasterRole: When MasterRole is 1, MUST include active crawls, where crawl status (2.2.1.4) is not done, stopped or forbid, for all crawl components from the list of crawls in the crawl status set section 3.1.1.1. When MasterRole is 0, the list of crawls is filtered by ComponentID passed in.

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

Result Sets: MUST return the following result set:

3.1.5.25.1 Get Crawls Result Set

The result set MUST contain zero or more rows, each corresponding to a crawl from the Crawl Status structure see section 3.1.1.1. The result set MUST contain zero rows when there is a HaveResetRequest attribute of the Component Activity structure section 3.1.1.35 is equal to 1. When MasterRole is 1, MUST include active crawls, where crawl status (2.2.1.4) is not done, stopped or forbid, for all crawl components from the list of crawls in the crawl status set section 3.1.1.1.

When the MasterRole is 0, the result set is filtered by the crawls that MUST match all of the following criteria:

- The crawl identifier is also in the Crawl Component Status set section 3.1.1.25
- The @ComponentID matches the Crawl Component Status ComponentID
- The crawl status (3.1.1.1) and Component Status (3.1.1.25) MUST match any one of the following conditions:
  - Crawl status is started.
  - Crawl status is starting with the substatus equal to Wait For Crawl Components.
  - Crawl status is paused.
  - Crawl status is pausing and the crawl component status is not already paused.
  - Crawl status is resuming and the crawl component status is not already started.
  - Crawl status is stopping and the substatus is equal to Wait For Crawl Components To Stop and the crawl component status is not already stopped.

- Crawl status is completing and the substatus is Wait For Crawl Components and the crawl component status is not already done.

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

```
CrawlID              int,
CrawlType            int,
Status               int,
SubStatus            int,
Request              int,
ContentSourceID      int,
MainCrawlID          int;
```

CrawlID: CrawlID as specified in section 3.1.1.1.

CrawlType: CrawlType as specified in section 3.1.1.1.

Status: Status as specified in section 3.1.1.1.

SubStatus: SubStatus as specified in section 3.1.1.1.

Request: Request as specified in section 3.1.1.1.

ContentSourceID: ContentSourceID as specified in section 3.1.1.1.

MainCrawlID: MainCrawlID as specified in section 3.1.1.1.

3.1.5.26 proc_MSS_GetCurrentRegistryVersion

The proc_MSS_GetCurrentRegistryVersion stored procedure is called to retrieve the **component registry version** from the configuration properties (specified in section 3.1.1.36).

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

```
PROCEDURE proc_MSS_GetCurrentRegistryVersion (
    @LikeKey nvarchar(100),
    @CurrentRegistryVersion int OUTPUT
);
```

@LikeKey: A string containing the name of the component for which to return the component registry version. The string MUST begin with search component name (defined in section 3.1.1.39) which MUST be followed by "\CurrentRegistryVersion"

@CurrentRegistryVersion: Upon return from this stored procedure, this parameter MUST be set to current component registry version if it exists in the configuration properties. This value is undefined if the component registry version is not found.

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

Value	Description
1	If the component registry version can't be found
0	If the component registry version is successfully found

Result Sets: SHOULD NOT<11> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.5.27 proc_MSS_GetDatabaseID

The **proc_MSS_GetDatabaseID** stored procedure is called to retrieve the crawl store identifier associated with a crawl component.

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

```
PROCEDURE proc_MSS_GetDatabaseID (
@ComponentID int,
@GthrDBID int OUTPUT
);
```

@ComponentID: The identifier of the crawl component for which to retrieve the crawl store identifier.

@GthrDBID: Upon return from this stored procedure, this parameter MUST be set to crawl store identifier associated with a given crawl component. If the crawl store identifier association does not exist for the given crawl component then this value MUST be set to -1.

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

Result Sets: SHOULD NOT<12> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.5.28 proc_MSS_GetAllGathererDatabaseIDs

The proc_MSS_GetAllGathererDatabaseIDs stored procedure is called to retrieve the list of crawl stores.

```
PROCEDURE proc_MSS_GetAllGathererDatabaseIDs();
```

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

Result Sets:

This procedure MUST return the following result set:

3.1.5.29 Gatherer DB IDs Result Set

The result set contains a list of crawl stores and MUST contain zero or more rows, each row corresponding to a single crawl store.

```
ID                    nvarchar(40) NOT NULL;
```

ID: A GUIDString which identifies the crawl store.

3.1.5.30 proc_MSS_GetAllPropertyStoreIDs

The proc_MSS_GetAllPropertyStoreIDs stored procedure is called to retrieve the list of property stores.

PROCEDURE proc_MSS_GetAllPropertyStoreIDs();

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

Result Sets:

This procedure MUST return the following result set:

3.1.5.31 Property Store IDs Result Set

The result set contains a list of property stores and MUST contain zero or more rows, each row corresponding to a single property store.

```
ID                    nvarchar(40) NOT NULL;
```

ID: A GUIDString which identifies the property store.

3.1.5.32 proc_MSS_GetDeleteCrawl

The proc_MSS_GetDeleteCrawl stored procedure is called to retrieve the list of content source and start address combinations that MUST be deleted in a delete crawl.

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

```
PROCEDURE proc_MSS_GetDeleteCrawl (
    @CrawlID int
);
```

@CrawlID: Delete crawl identifier

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

Result Sets:

This procedure MUST return the following result set.

3.1.5.32.1 Delete Crawl Result Set

The Delete Crawl result set contains information about content source and start address combinations that MUST be deleted in a delete crawl. The result set MUST contain zero or more rows.

```
ContentSourceID                  int NOT NULL,
StartAddressID                   int NOT NULL;
```

ContentSourceID: The content source identifier.

StartAddressID: The start address identifier. If the start address identifier is 0, then all the start addresses for the content source ContentSourceID MUST be deleted.

3.1.5.33 proc_MSS_GetDeletedScopesForCompilation

The **proc_MSS_GetDeletedScopesForCompilation** stored procedure is called to retrieve all search scopes to be deleted in the current scope compilation. The stored procedure MUST return all the data in the Deleted Scopes to Compile Result Set defined in Section 3.1.1.32.

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

```
PROCEDURE proc_MSS_GetDeletedScopesForCompilation();
```

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

Result Sets: MUST return the following result set:

3.1.5.33.1 Deleted Search Scopes Result Set

The Deleted Search Scopes Result Set MUST contain one row corresponding to each search scope to be deleted in the current search scope compilation.

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

```
ScopeID                     int;
```

ScopeID: The unique identifier of the search scope.

3.1.5.34 proc_MSS_GetDocCount

The proc_MSS_GetDocCount stored procedure is called to retrieve counts of items in the link set (section 3.1.1.5), crawl queue (section 3.1.1.4), URLs pending deletion from the **crawl URL history** (section 3.1.1.2) and optionally the number of documents in crawl URL history.

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

```
PROCEDURE proc_MSS_GetDocCount (
    @SkipDocCount                int,
    @DocCount                    int OUTPUT,
    @DocCountWithPendingDeletes  int OUTPUT,
    @LinksInQueue                int OUTPUT,
    @DocsInQueue                 int OUTPUT
);
```

@SkipDocCount: If DocCount needs to be calculated then this parameter MUST be set to 0 otherwise 1.

@DocCount: If SkipDocCount is 0 this parameter MUST be set to the number of ULRs in crawl URL history. This number MUST exclude URLs that are pending deletion. If SkipDocCount was set to 1 then this parameter MUST be set to 0.

@DocCountWithPendingDeletes: After this stored procedure completes, this parameter MUST be set to the number of URLs pending deletion from the crawl URL history.

@LinksInQueue: After this stored procedure completes, this parameter MUST be set to the number of links in the link set

@DocsInQueue: After this stored procedure completes, this parameter MUST be set to the number of items in **crawl queue**

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

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

3.1.5.35 proc_MSS_GetDocStatus

The **proc_MSS_GetDocStatus** stored procedure is called to retrieve the DocId, ErrorId and DisplayURL of the records from the Crawl URL History, as specified in section 3.1.1.2, with the value DeletePending equal to "0" and with the value DisplayHash equal to one of the items in the DisplayHashes list that was passed in.

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

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

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

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

Result Sets: MUST return the following result set:

3.1.5.35.1 Document Status Result Set

The Document Status result set returns the DocId, ErrorId and DisplayURL for items matching specified @DisplayHashes from the Crawl URL History as specified in section 3.1.1.2. The result set MUST contain zero or more rows, each corresponding to an item within Crawl URL History. The result is sorted by the DisplayURL (Ascending), IndexType (Descending) and DocID (Ascending).

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

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

DocId: The unique identifier for the item.

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

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

3.1.5.36 proc_MSS_GetError

The proc_MSS_GetError stored procedure is called to retrieve error data for correspondent **HRESULT**. The **error** MUST be created if it does not already exist in the **Crawl Error Set**, see [MS-SQLPADM2] section 3.1.1.3 for more information.

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

```
PROCEDURE proc_MSS_GetError(
    @hrResult                   int,
    @ErrorID                    int OUTPUT,
    @ErrorLevel                 int OUTPUT,
    @MarkDelete                 bit OUTPUT
);
```

@hrResult: An integer identifies the HRESULT of the **error.**

@ErrorID: Upon return from this stored procedure, this parameter MUST be set to a unique identifier for the **error**.

@ErrorLevel: Upon return from this stored procedure, this parameter MUST be set to an integer representing **Crawl Error Level** as described in section 2.2.1.10. It MUST be set to 2 if the **error** has been added.

@MarkDelete: Upon return from this stored procedure, this parameter MUST be set to 1 if the **error** was marked as deleted. Otherwise, it MUST be 0. It MUST be set to 0 if the **error** has been added.

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

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

3.1.5.37 proc_MSS_GetGathererDBs

The proc_MSS_GetGathererDBs stored procedure is called to retrieve the list of crawl stores.

```
PROCEDURE proc_MSS_GetGathererDBs();
```

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

Result Sets:

This procedure MUST return the following result set:

3.1.5.37.1 Gatherer DBs Result Set

The result set contains a list of crawl stores and MUST contain zero or more rows, each row corresponding to a single crawl store. The result set MUST be ordered by Ordinal (ascending).

```
ID                      nvarchar(40) NOT NULL,
Ordinal                 int NOT NULL,
Name                    nvarchar(256) NOT NULL;
```

ID: A **GUIDString** which identifies the crawl store.

Ordinal: An integer which identifies the crawl store.

Name: The name of the crawl store.

3.1.5.38 proc_MSS_GetHost

The **proc_MSS_GetHost** stored procedure is called to retrieve the integer identifier of a crawl store and the integer identifier of a **host name**. If the host name is not found, a crawl store is assigned and a unique identifier for the host name is created. The crawl topology state, see [MS-SRCHTP] section 2.2.1.6, MUST be active for the stored procedure to succeed. Values for GthrDBID and HostID MUST be returned. The proc_MSS_GetGathererDBs stored procedure is called to retrieve the list of crawl stores.

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

```
PROCEDURE proc_MSS_GetHost (
   @HostName                nvarchar(300),
   @GthrDBID                int OUTPUT,
   @HostID                  int OUTPUT
);
```

@HostName: Host name. The value MUST NOT be NULL.

@GthrDBID: Upon return from this stored procedure, this parameter MUST be set to the identifier of the crawl store associated with this host name.

@HostID: Upon return from this stored procedure, this parameter MUST be set to the identifier of the host name.

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

Result Sets: MUST NOT return any result sets.

3.1.5.39 proc_MSS_GetIncompleteCommands

The **proc_MSS_GetIncompleteCommands** stored procedure is called to retrieve all commands from the commands set (section 3.1.1.41) intended for processing by a given object but not yet processed by that object.

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

```
PROCEDURE proc_MSS_GetIncompleteCommands (
   @ObjectName              nvarchar(1024)
);
```

@ObjectName: The name of the object requesting commands intended for it.

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

Result Sets: MUST return a Commands Result Set as specified in section 3.1.5.39.1. The returned result set MUST only contain rows from the commands set (section 3.1.1.41) where the Completed value is set to 0 and MUST contain exactly one result for each command in the commands set (section 3.1.1.41), where ObjectName (section 3.1.5.39.1) is set to the ObjectName value of the command (section 3.1.1.41), Description (section 3.1.5.39.1) is set to the Description value of the command (section 3.1.1.41), and Completed (section 3.1.5.39.1) is set to the Completed value of the command (section 3.1.1.41), and MUST NOT contain other results.

3.1.5.39.1 Commands Result Set

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

```
CommandID              bigint,
ObjectName             nvarchar(1024),
Description            nvarchar(3072),
CreatedUtcTime         datetime,
Completed              int,
CompletedUtcTime       datetime
```

CommandID: A 64-bit integer identifying the command.

ObjectName: The name of the object that is expected to process the command.

Description: A string which the processing object will interpret and act upon.

CreatedUtcTime: The time of creation of this command. This value MUST be ignored.

Completed: 0 if the command has not been processed; 1 if it has.

CompletedUtcTime: The time of completion of this command, or NULL if it has not been completed. This value MUST be ignored.

3.1.5.40 proc_MSS_GetManagedProperties

The **proc_MSS_GetManagedProperties** stored procedure is called to retrieve the list of managed properties from the **Managed Property Set**, as specified in [MS-SQLPADM2] section 3.1.1.1, which were added or modified, on or after the *@ldate* time

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

```
PROCEDURE proc_MSS_GetManagedProperties {
    @ldate                    datetime,
    @GetAll                   bit
};
```

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

@GetAll: This value MUST be set to 1 if all the managed properties that were added or modified, on or after the @ldate are to be retrieved. It MUST be set to 0 if only the non-internal managed properties are to be retrieved,

Return Code Values: An integer value which MUST be 0

Result Sets: MUST return the following result set

3.1.5.40.1 GetManagedProperties Result Set

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

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

```
PID                                 int,
FriendlyName                        nvarchar(64),
PropertyDescription                 nvarchar(2048),
ManagedType                         int,
FullTextQueriable                   bit,
Retrievable                         bit,
Scoped                              bit,
RespectPriority                     bit,
RemoveDuplicates                    bit,
NoDelete                            bit,
NoMap                               bit,
Hidden                              bit,
HasMultipleValues                   bit,
NoWordBreaker                       bit,
NameNormalized                      bit,
IncludeInMD5                        bit,
Mapped                              bit,
QueryIndependentRank                bit,
UserFlags                           smallint,
WordBreakerOverride                 int,
Weight                              float,
LengthNormalization                 float,
LastModified                        datetime,
ForceChangeSchemaVersion            int,
MaxIndexedStringLength              int,
MaxNonIndexedStringLength           int,
MaxRetrievalLength                  int,
DecimalPlaces                       int,
IsInDocProps                        bit,
IsInPropertyBlobOptimizedResults    bit,
IsInFixedColumnOptimizedResults     bit,
OverrideHasMultipleValues           bit,
SuppressStringNormalizer            bit,
Pronunciation                       bit,
QueryPropertyBlob                   bit,
NicknameExpansion                   bit,
IsQIRCustomizationAllowed           bit,
DefaultForQIR                       int;
SplitStringCharacters               nvarchar(64);
```

PID: The unique identifier for the managed property. This value MUST NOT be NULL.

FriendlyName: A string that uniquely identifies the managed property. This value MUST NOT be NULL.

PropertyDescription: The description of the managed property.

ManagedType: The type, as specified in section 2.2.1.13, of the managed property.

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

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

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

RespectPriority: This value 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.

RemoveDuplicates: This value MUST be 1 if the duplicate entries are removed. Otherwise, MUST be 0.

NoDelete: This value MUST be 1 if this property can never be deleted from the metadata schema. Otherwise, it MUST be 0.

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

Hidden: This value MUST be set to 1 to specify that the property is an internal property. Otherwise, it MUST be set to 0.

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

NoWordBreaker: This parameter MUST be ignored by the client.

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

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

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

QueryIndependentRank: This value 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. This value MUST NOT be NULL.

WordBreakerOverride: This parameter MUST be ignored by the client.

Weight: A decimal value used to adjust **property oriented rank**. This value MUST NOT be NULL.

LengthNormalization: A decimal value used to adjust property oriented rank. This value MUST NOT be NULL.

LastModified: The date and time of the last change to the managed property. This value MUST NOT be NULL.

ForceChangeSchemaVersion: This parameter MUST be ignored by the client.

MaxIndexedStringLength: The maximum number of characters persisted in the string value column in the *MSSDocProps* table defined in [MS-SQLPQ2] section 2.2.5.2. If the string value length is greater than MaxIndexedStringLength, the string is truncated to MaxIndexedStringLength. This value MUST NOT be NULL.

MaxNonIndexedStringLength: If the string value length in the *MSSDocProps* table, defined in [MS-SQLPQ2] section 2.2.5.2, is greater than the size allowed by the MaxIndexedStringLength, the string is truncated to MaxIndexedStringLength and the string overflow is stored in the strVal2

column, in the *MSSDocProps* table defined in [MS-SQLPQ2] section 2.2.5.2, up to the size allowed by the MaxNonIndexedStringLength. This value MUST NOT be NULL.

MaxRetrievalLength: The maximum number of characters persisted for a fixed-length string property in the *MSSDocResults* table defined in [MS-SQLPQ2] section 2.2.5.2. This MUST NOT be NULL.

DecimalPlaces: The number of floating point decimal places that must be honored in the metadata index. This value MUST NOT be NULL.

IsInDocProps: This value MUST be 1 if the managed property is stored in the metadata index. Otherwise, MUST be 0.

IsInPropertyBlobOptimizedResults: This value MUST be 1 if the managed property is persisted in the *PropertyBlob* column of the *MSSDocResults* table defined in [MS-SQLPQ2] section 2.2.5.2. Otherwise, MUST be 0.

IsInFixedColumnOptimizedResults: This value MUST be 1 if the managed property is persisted in the *MSSDocResults* table defined in [MS-SQLPQ2] section 2.2.5.2. Otherwise, MUST be 0.

OverrideHasMultipleValues: This value MUST be 1 if the metadata schema object model MUST NOT change the *HasMultipleValues* parameter because it has been set explicitly. Otherwise, MUST be 0.

SuppressStringNormalizer: This value MUST be 1 if the string normalization for the managed property is to be skipped. Otherwise this value MUST be 0.

Pronunciation: This value MUST be 1 if the managed property needs a pronunciation string. Otherwise, MUST be 0.

QueryPropertyBlob: This value MUST be 1 if the managed property is persisted in the *PropertyBlob* column of the *MSSDocResults* table defined in [MS-SQLPQ2] section 2.2.5.2. Otherwise, MUST be 0.

NicknameExpansion: This value MUST be set to 1 if the managed property is compared to the nickname mappings at query time. Otherwise, it MUST be 0.

IsQIRCustomizationAllowed: This value MUST be 1 if the managed property supports query independent rank and customization of query independent rank is allowed. Otherwise, MUST be 0.

DefaultForQIR: The default value for query independent rank. If the managed property does not participate in query independent rank then this value MUST be 0.

SplitStringCharacters: This value MUST contain characters which are used to split the string data for the managed property into separate strings which do not contain the SplitStringCharacters. Otherwise, it MUST be NULL.

3.1.5.41 proc_MSS_GetMappingsForCrawledPropertiesById

Retrieves a list managed properties that are mapped to a crawled property utilizing the Metadata Schema as specified in [MS-SQLPADM] section 3.1.1.1.

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

```
PROCEDURE proc_MSS_GetMappingsForCrawledPropertiesById (
    @crawledPropertyId              int
```

```
);
```

@crawledPropertyId: Identifier for the crawled property.

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

Result Sets:

This procedure MUST return the result set described in section 3.1.5.42.

3.1.5.42 Get Mappings for Crawled Properties by Identifier Result Set

The result set MUST contain zero or more rows, each corresponding to a managed property that is mapped to the crawled property.

```
PID                    int NOT NULL,
mappingorder           int NOT NULL;
```

PID: Identifier for the managed property which is mapped to the crawled property.

mappingorder: mapping order for the managed property.

3.1.5.43 proc_MSS_GetMasterRole

This stored procedure is used by each crawl component to determine whether it is the master crawl component.

The stored procedure MUST set @MasterRole to the value of the Master field of the record of Crawl Component Set ([MS-SRCHTP] section 3.1.1.3) with CrawlComponentNumber equal to @ComponentID. If no such record exists, then the stored procedure MUST set @MasterRole to 1.

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

```
PROCEDURE proc_MSS_GetMasterRole(
    @ComponentID              int,
    @MasterRole               bin OUTPUT
);
```

@ComponentID: Unique identifier of a crawl component.

@MasterRole: MUST be set to 1 if the crawl component is the master crawl component of the search service application and 0 otherwise.

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

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

3.1.5.44 proc_MSS_GetMatrixPIAliases

This stored procedure extracts data from the MSSDocProps table (see [MS-SQLPQ2] section 2.2.5.2).

```
PROCEDURE proc_MSS_GetMatrixPIAliases();
```

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

Result Sets: This procedure MUST return the following result set:

3.1.5.44.1 MatrixPI Aliases Result Set

For each row in MSSDocProps (see [MS-SQLPQ2] section 2.2.5.2) that has a **Pid** value of 19, the result set MUST contain a row for each row in MSSDocProps that has the same **DocId** value as the first row, and a **Pid** value of 15, 21 or 27.

```
From NVARCHAR(450) NOT NULL,
To NVARCHAR(MAX) NOT NULL
```

From: Value of the strVal field of the row in MSSDocProps that has a Pid value of 19.

To: Value of the strVal2 field of the row in MSSDocProps that has a Pid value of 15, 21 or 27.

3.1.5.45 proc_MSS_GetNextCrawlBatch

The stored procedure is called to retrieve the next batch of documents to be crawled. The records this stored procedure returns are marked within the Crawl Queue (section 3.1.1.4) as being currently crawled by setting the BatchID field to the identifier of the newly created batch.

The stored procedure MUST generate a new unique identifier for the new crawl batch and on successful completion of this stored procedure the output parameter @BatchID MUST be set to this value.

The stored procedure MUST select a set of records from the Crawl Queue with the BatchID field equal to 0 and the CrawlID field equal to @CrawlID. The number of records in this set MUST be less than or equal to the value of @BatchSize. For each record in this set, the stored procedure MUST set the BatchID field to the new generated identifier which is to be returned in @BatchID. The stored procedure MUST return the selected set of records.

The stored procedure MUST raise an error if crawl component with identifier @CrawlComponentID has status Disable in Crawl Component Set ([MS-SRCHTP] section 3.1.1.3).

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

```
PROCEDURE proc_MSS_GetNextCrawlBatch(
    @ComponentID            int,
    @ProjectID              int,
    @CrawlID                int,
    @BatchSize              int,
    @BatchID                bigint OUTPUT
);
```

@ComponentID: Identifier of the crawl component.

@ProjectID: A project identifier as specified in Project Identifier (section 2.2.1.1).

@CrawlID: MUST be set to the unique identifier for the crawl.

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

@BatchID: The unique identifier of the batch returned by this stored procedure.

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

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

Result Sets: MUST return the following result set:

3.1.5.45.1 GetNextCrawlBatch Result Set

The GetNextCrawlBatch result set returns a list of items to be crawled. The result set MUST contain zero or more rows and MUST be equal to or less than *@BatchSize*.

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

```
CrawlID                 int,
SourceDocID             int,
DocID                   int,
DisplayURL              nvarchar(1500),
AccessURL               nvarchar(1500),
CompactURL              nvarchar(40),
EndPathFlag             int,
StartAddressID          int,
HostDepth               int,
EnumerationDepth        int,
TransactionFlags        int,
MD5                     int,
PropMD5                 int,
UseChangeLog            int,
IndexType               int,
LastModifiedTime        bigint,
FolderDelCount          int,
Reserved1               bigint,
Reserved2               bigint,
Reserved3               int,
Reserved4               int,
TransactionType         int,
LCID                    int,
SeqID                   bigint,
ChangeLogCookie         int,
ChangeLogCookieType     int,
ChangeLogBatchID        int,
Scope                   int,
DocPropsMD5             bigint,
Retry                   int,
RetryCount              int,
DocPropsBlob            varbinary(8000),
HostID                  int,
ParentHostID            int,
LinksBitmap             int,
CachedBlob              varbinary(8000),
SecurityID              nvarchar(40),
```

```
PHFlags                    int,
DelayRetryCount            int;
```

In the following field specifications, the selected record from the Crawl Queue, as specified in section 3.1.1.4, is referred to as the **record from queue**. Also, the record from the Crawl URL History, as specified in section 3.1.1.2, which has the value of the field DocID equal to the one of the record from the queue is referred to as the **record from history.**

CrawlID: Value of the field CrawlID of the record from queue.

SourceDocID: Value of the field SourceDocID of the record from queue.

DocID: Value of field DocID of the record from queue.

DisplayURL: Value of field DisplayURL of the record from history.

AccessURL: Value of the field AccessURL of the record from history.

CompactURL: Value of the field CompactURL of the record from history.

EndPathFlag: Value of the field EndPathFlag of the record from history.

StartAddressID: Value of the field StartAddressID of the record from queue.

HostDepth: Value of the field HostDepth of the record from queue.

EnumerationDepth: Value of the field EnumerationDepth of the record from queue.

TransactionFlags: Value of the field TransactionFlags of the record from queue.

MD5: Value of the field MD5 of the record from history.

PropMD5: Value of the field PropMD5 of the record from history.

UseChangeLog: Value of the field UseChangeLog of the record from history.

IndexType: Value of the field IndexType of the record from history.

LastModifiedTime: Value of the field LastModifiedTime of the record from history.

FolderDelCount: This parameter MUST be ignored by the client.

Reserved1: This parameter MUST be ignored by the client**.**

Reserved2: This parameter MUST be ignored by the client.

Reserved3: This parameter MUST be ignored by the client.

Reserved4: This parameter MUST be ignored by the client**.**

TransactionType: Value of the field TransactionType of the record from queue.

LCID: Value of the field LCID of the record from history.

ItemType: Value of the field ItemType of the record from history.

SeqID: Value of the field SeqID of the record from queue.

ChangeLogCookie: Value of the field ChangeLogCookie of the record from history.

ChangeLogCookieType: type of ChangeLogCookie. The values of the ChangeLogCookieType is implementation detail of protocol client

ChangeLogBatchID: Value of the field ChangeLogBatchID of the record from queue.

Scope: Value of the field Scope of the record from queue.

DocPropsMD5: Value of the field DocPropsMD5 of the record from history if @ProjectID equals Portal Content, 0 if @ProjectID equals Anchor Project.

Retry: Value of the field Retry of the record from history.

RetryCount: Value of the field RetryCount of the record from history.

DocPropsBlob: Value of the field DocPropsBlob of the record from history if @ProjectID equals Portal Content, NULL if @ProjectID equals Anchor Project.

HostID: Value of the field HostID of the record from history if @ProjectID equals Portal Content, -1 if @ProjectID equals Anchor Project.

ParentHostID: Value of the field ParentHost of the record from history if @ProjectID equals Portal Content, -1 if @ProjectID equals Anchor Project.

LinksBitmap: Value of the field LinksBitmap of the record from history if @ProjectID equals Portal Content, 0 if @ProjectID equals Anchor Project.

CachedBlob: Value of the field CachedBlob of the record from history if @ProjectID equals Portal Content, NULL if @ProjectID equals Anchor Project.

SecurityID: Value of the field SecurityID of the record from history if @ProjectID equals Portal Content, NULL if @ProjectID equals Anchor Project.

PHFlags: Value of the field PHFlags of the record from history if @ProjectID equals Portal Content, 0 if @ProjectID equals Anchor Project.

DelayRetryCount: Value of the field DelayRetryCount of the record from history.

ChangeLogCookieEnd: Value of the field ChangeLogCookieEnd of the record from history if @ProjectID equals Portal Content, NULL if @ProjectID equals Anchor Project.

LogLevel: Value of the field LogLevel of the record from history if @ProjectID equals Portal Content, 0 if @ProjectID equals Anchor Project.

IndexType: Value of the field IndexType of the record from history if @ProjectID equals Portal Content, 0 if @ProjectID equals Anchor Project

3.1.5.46 proc_MSS_InvalidateAdminDocIDChunks

The **proc_MSS_InvalidateAdminDocIDChunks** stored procedure is called to mark all groups from Current DocID Groups Set (section 3.1.1.17) as invalid.

The stored procedure MUST set InvalidatedByRefactoring to 1 for all record in Current DocID Chunks Set.

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

```
PROCEDURE proc_MSS_InvalidateAdminDocIDChunks();
```

Return Value: This stored procedure returns an integer that MUST be ignored

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

3.1.5.47 proc_MSS_GetNextDocIDChunk

The **proc_MSS_GetNextDocIDChunk** stored procedure returns a range of document identifier values which are not currently assigned to crawled documents.

This stored procedure MUST return @NewNextDocID and @NewMaxDocID from the Current DocID Chunks Set (section 3.1.1.17), where @NewNextDocID is assigned with FirstDocId and @NewMaxDocId is assigned with LastDocId. This record is selected based on the following conditions:

- DBID equal to @GthrDBID
- @CurrentMaxDocID is greater or equal to FirstDocId
- @CurrentMaxDocID is smaller than LastDocID
- InvalidatedByRefactoring is equal to 0

If there is no record in Current DocID Chunks Set which meets the preceding conditions specified, then the stored procedure MUST create and add a record to Current DocID Chunks Set (section 3.1.1.17) with the value DBID equal to @GthrDBID. The range defined by FirstDocId and LastDocId MUST not overlap any range of document identifiers defined by other existing records within the set. In this case the stored procedure MUST return the range of document identifiers defined by the newly added record.

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

```
PROCEDURE proc_MSS_GetNextDocIDChunk(
    @GthrDBID               int,
    @CurrentMaxDocID        int,
    @NewNextDocID           int OUTPUT,
    @NewMaxDocID            int OUTPUT
);
```

@GthrDBID: unique identifier of crawl store

@CurrentMaxDocID: identifier which the group should contain

@NewNextDocID: the first DocID of the result group

@NewMaxDocID: the last DocID of the result group

Return Value: This stored procedure returns an integer that MUST be ignored

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

3.1.5.48 proc_MSS_GetPauseCrawlsReasons

The proc_MSS_GetPauseCrawlsReasons stored procedure is used to get the Pause Crawls Reasons bit mask as specified in section 2.2.2.1 from the maximum value of DPR_Crawls attribute within the Component Activity set as specified in section 3.1.1.35.

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

```
PROCEDURE proc_MSS_GetPauseCrawlsReasons ();
```

Return Values: An integer which MUST be a Pause Crawls Reasons as specified in section 2.2.2.1.

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

3.1.5.49 proc_MSS_GetPauseStatus

The proc_MSS_GetPauseStatus stored procedure is called to identify whether the crawl and **master merge** are paused or reset request is posted for a given component and project combination in Component Activity (defined in section 3.1.1.35). The proc_MSS_GetGathererDBs stored procedure is called to retrieve the list of crawl stores.

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

```
PROCEDURE proc_MSS_GetPauseStatus (
    @ComponentName          nvarchar(100),
    @ProjectName            nvarchar(100),
    @lAreCrawlsPaused       int,
    @lIsMasterMergePaused   int
);
```

@ComponentName: The search component name as defined in section 3.1.1.39.

@ProjectName: The Project Name as defined in section 2.2.1.19

@lAreCrawlsPaused: Upon return from this stored procedure this parameter MUST be set to 1 if crawl is paused, otherwise it MUST be set to 0.

@lIsMasterMergePaused: Upon return from this stored procedure this parameter MUST be set to a value that satisfies the bit-mask 0x8003. Bit 0x8000 indicates that this is the version of this stored procedure that corresponds to SP1. If bit 0x1 is set then the master merge is paused, otherwise this bit MUST NOT be set. If the bit 0x2 is set then reset request is posted, otherwise this bit MUST NOT be set.

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

Result Sets: SHOULD NOT<19> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.5.50 proc_MSS_GetPreferredNameList

This stored procedure extracts data from the MSSDocProps table (see [MS-SQLPQ2] section 2.2.5.2).

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

```
PROCEDURE proc_MSS_GetPreferredNameList();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.5.50.1 Preferred Name List Result Set

The Preferred Name List result set MUST contain all the distinct values of strVal field from every row in MSSDocProps (see [MS-SQLPQ2] section 2.2.5.2) that has a **Pid** value of 19.

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

```
NAME                        nvarchar(450);
```

NAME: Value of the strVal field of the row in MSSDocProps that has a **Pid** value of 19.

3.1.5.51 proc_MSS_GetRankingModels

See [MS-SQLPADM2] section 3.1.5.105

3.1.5.52 proc_MSS_GetSampleExtremes

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

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

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

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

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

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

Result Sets: MUST return the following result set:

3.1.5.52.1 Sample Extremes Result Set

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

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

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

CrawledPropertyId: A unique identifier of the crawled property.

CPCount: The number of samples for the CrawledPropertyId.

3.1.5.53 proc_MSS_GetSchemaHighLevelInfo

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

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

```
PROCEDURE proc_MSS_GetSchemaHighLevelInfo();
```

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

Result Sets: MUST return the following result set:

3.1.5.53.1 Schema High Level Info Result Set

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

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

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

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

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

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

LastURIAdds: This parameter MUST be ignored by the client.

LastURIModifiedDeleted: This parameter MUST be ignored by the client.

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

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

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

LastSmpDelete: This parameter MUST be ignored by the client.

LastAliasAdd: This parameter MUST be ignored by the client.

LastAliasOther: This parameter MUST be ignored by the client.

ForceChangeSchemaVersion: This parameter MUST be ignored by the client.

3.1.5.54 proc_MSS_GetSchemaMappings

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

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

```
PROCEDURE proc_MSS_GetSchemaMappings();
```

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

Result Sets: MUST return the following result set:

3.1.5.54.1 GetSchemaMappings Result Set

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

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

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

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

MappingOrder: An integer representing the mapping order of the crawled property to a managed property.

CrawledPropertyId: A 32-bit integer that uniquely identifies the crawled property.

3.1.5.55 proc_MSS_GetSchemaParameters

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

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

```
PROCEDURE proc_MSS_GetSchemaParameters();
```

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

Result Sets: MUST return the following result set:

3.1.5.55.1 GetSchemaParameters Result Set

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

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

```
ParamName                    nvarchar(40),
IsString                     bit,
strValue                     nvarchar(256),
fltValue                     float;
```

ParamName: The name of the schema parameter.

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

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

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

3.1.5.56 proc_MSS_GetScopeRulesForCompilation

The **proc_MSS_GetScopeRulesForCompilation** stored procedure is called to retrieve the search scope rules that correspond to search scopes which are involved in the current search scope compilation. The stored procedure MUST return all the data in the Scope Rules to Compile Set defined in Section 3.1.1.30.

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

```
PROCEDURE proc_MSS_GetScopeRulesForCompilation();
```

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

Result Sets: This stored procedure MUST return the Scope Rules Result Set as specified in [MS-SQLPADM2] section 2.2.4.5. The result set MUST contain one row for each search scope rule in the current search scope compilation.

3.1.5.57 proc_MSS_GetScopesForCompilation

The **proc_MSS_GetScopesForCompilation** stored procedure is called to retrieve the search scopes that are involved in the current search scope compilation. The stored procedure MUST return all the data in the Scopes to Compile Set defined in Section 3.1.1.29.

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

```
PROCEDURE proc_MSS_GetScopesForCompilation();
```

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

Result Sets: MUST return the Scope Compilation Result Set defined in Section 2.2.4.1. The result set MUST contain one row for every search scope that is involved in the current search scope compilation.

3.1.5.58 proc_MSS_GetSDID

The **proc_MSS_GetSDID** stored procedure is called to retrieve a **search security descriptor (2)** from the metadata schema.

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

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

@sdCheckSum: An identifier of a search security descriptor (2).

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

Result Sets: MUST return the following result set:

3.1.5.58.1 GetSDID Result Set

The GetSDID result set returns the list of search security descriptors (2) that are associated with @sdCheckSum. The result set MUST contain zero or more rows which are not ordered.

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

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

SDID: A unique identifier of the search security descriptor (2).

SD: The search security descriptor (2).

3.1.5.59 proc_MSS_GetStaticRankingFeatures

The proc_MSS_GetStaticRankingFeatures stored procedure is specified in [MS-SQLPADM2] section 3.1.5.127.

3.1.5.60 proc_MSS_GetStatus

The proc_MSS_GetStatus stored procedure is called to return either a crawl status as specified in section 2.2.1.4 or a content source status as specified in section 2.2.1.17 from the crawl status structure as specified in section 3.1.1.1.

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

```
PROCEDURE proc_MSS_GetStatus (
    @ProjectID              int,
    @ContentSourceID        int
```

```
);
```

@ProjectID: Project Identifier, see section 2.2.1.1. If the ProjectID is not found in the crawl status structure the stored procedure will return 0.

@ContentSourceID: Identifier of the content source associated with the crawl or -1.

Return Values: A value which MUST be listed in the crawl status as specified in section 2.2.1.4 when the ContentSourceID parameter is -1. When the ContentSourceID parameter is not equal to -1, MUST be a value listed in the content source status simple data type. If the ProjectID is not found the return code MUST be 0. When the content source status is requested, the return code of 0 MUST be returned if either the ProjectID or ContentSourceID are not found.

Result Sets: SHOULD NOT<20> return any result set. The protocol client MUST ignore any result sets returned by this stored procedure.

3.1.5.61 proc_MSS_GetStatusChangeRequest

The proc_MSS_GetStatusChangeRequest stored procedure indicates that a **catalog reset**, pause, resume of a crawl or a merge activity needs to take place for a given search component. The ComponentName and ProjectName parameters are used to lookup the search component's status in the component activity structure see section 3.1.1.35. If the HaveResetRequest attribute is 0 and all of the following conditions are met:

- DPR_Master_Merge is not equal to CPR_Master_Merge
- Both DPR_Master_Merge and CPR_Master_Merge are nonzero

Then the component activity record MUST be updated to make CPR_Master_Merge equal DPR_Master_Merge. Also while the HaveResetRequest attribute is 0 and all of the following conditions are met:

- DPR_Crawls is not equal to CPR_Crawls
- Both DPR_Crawls and CPR_Crawls are nonzero

Then the component activity record MUST be updated to make CPR_Crawls equal to DPR_Crawls.

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

```
PROCEDURE proc_MSS_GetStatusChangeRequest (
    @ComponentName nvarchar(100),
    @ProjectName nvarchar(100),
    @IsAdmin bit,
    @HaveResetRequest int OUTPUT,
    @HavePauseRequest int OUTPUT,
    @PauseLevel int OUTPUT
);
```

@ComponentName: A Search Component Name as specified in section 3.1.1.39.

@ProjectName: The Project Name as specified in section 2.2.1.19

@IsAdmin: Must be 1 when the search component is an **administration component**. Must be 0 when the search component is not an administration component.

@HaveResetRequest: Upon return from this stored procedure, this parameter MUST be set to 1 when the HaveResetRequest in the component activity structure is set to 1. MUST be 0 if HaveResetRequest attribute for this search component in the component activity structure is 0.

@HavePauseRequest: Upon return from this stored procedure, this parameter MUST be 0 when ANY of the following conditions are met using the attributes of the component activity structure:

- HaveResetRequest is set to 1
- DPR_Master_Merge is not equal to CPR_Master_Merge and Both DPR_Master_Merge and CPR_Master_Merge are nonzero
- DPR_Crawls is not equal to CPR_Crawls and Both DPR_Crawls and CPR_Crawls are nonzero
- DPR_Crawls is not equal to CPR_Crawls and DPR_Crawls is equal to 0
- DPR_Master_Merge is equal to CPR_Master_Merge and DPR_Crawls is equal to CPR_Crawls

MUST be 1 when ANY of the following conditions are met using the attributes of the component activity structure:

- DPR_Master_Merge is not equal to CPR_Master_Merge and CPR_Master_Merge is equal to zero.
- DPR_Crawls is not equal to CPR_Crawls and CPR_Crawls is equal to 0

@PauseLevel: Must be an integer in the following table.

Value	Description
0	MUST be 0 when none of the following conditions are met: ▪ the DPR_Master_Merge is not equal to CPR_Master_Merge and either DPR_Master_Merge is equal to zero or CPR_Master_Merge is equal to zero. ▪ the DPR_Crawls is not equal to CPR_Crawls and either DPR_Crawls is equal to zero or CPR_Crawls is equal to zero.
1	MUST be 1 when the DPR_Master_Merge is not equal to CPR_Master_Merge and either DPR_Master_Merge is equal to zero or CPR_Master_Merge is equal to zero.
2	MUST be 2 when the DPR_Crawls is not equal to CPR_Crawls and either DPR_Crawls is equal to zero or CPR_Crawls is equal to zero.

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

Value	Description
0	The search component does not need a status change and MUST ignore HaveResetRequest, HavePauseRequest and PauseLevel. MUST be 0 when the search component is an administration component and there is a crawl component in the component activity structure section 3.1.1.35 with HaveResetRequest equal to 1 and that components status in the crawl component status structure section 3.1.1.25 is not disabled.
1	The search component needs to perform a catalog reset, pause or resume activity based on the HaveResetRequest, HavePauseRequest and PauseLevel output parameters.

Result Sets: MUST NOT return any result sets.

3.1.5.62 proc_MSS_GetTopAnchorLinks

The proc_MSS_GetTopAnchorLinks stored procedure is called to list anchor text, social distance property as specified in section 3.1.1.11, annotations as specified in section 3.1.1.37, document title, and document path and anchor document properties BLOB as specified in section 3.1.1.38 information which was collected in the crawl indicated by the CrawlID. This information is returned for a range of documents greater than or equal to the minimum document identifier and less than the maximum document identifier.

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

```
PROCEDURE proc_MSS_GetTopAnchorLinks (
    @CrawlID int,
    @MinTargetDocID int,
    @MaxTargetDocID int
);
```

@CrawlID: The unique identifier for the crawl.

@MinTargetDocID: Document identifiers whose information is returned must be greater than or equal to @MinTargetDocID. .

@MaxTargetDocID: Document identifiers whose information is returned must be less than MaxTargetDocID.

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

Result Sets: MUST return the following result set:

3.1.5.62.1 Top Anchor Links Result Set

The GetTopAnchorLinks result set returns anchor text, social distance property as specified in section 3.1.1.11, annotations as specified in section 3.1.1.37, document title, and document path and anchor document properties BLOB as specified in section 3.1.1.38 information which has been collected during the crawl indicated by the crawl identifier. The result set MUST contain zero or more rows which MUST be ordered by TargetDocID (ascending) and then LinkId (ascending).

```
TargetDocID                 int NOT NULL,
SourceDocID                 int NOT NULL,
Pid                         int NOT NULL,
AnchorNumeric               int NOT NULL,
AnchorFrequency             int NOT NULL,
LCID                        int NOT NULL,
LinkId                      bigint NOT NULL,
AnchorText                  nvarchar(1500) NOT NULL,
AnchorBinary                varbinary(2048) NOT NULL,
ChangeType                  int NOT NULL,
InterSite                   int NOT NULL;
```

TargetDocID: The document identifier (1) of the item which the data is applied to.

SourceDocID: When Pid equals 10, MUST be the document identifier (1) of the item where the data originated. When the Pid is not equal to 10, the value for SourceDocID MUST be ignored.

Pid: The unique identifier of a managed property which MUST be one of the following: 10, 101, 102, 306, 307 2147418050, 2147418051, 2147418052.

AnchorNumeric: When Pid equals 306 or 307 and when AnchorText is null, MUST be the value of AnnotationNumeric column from the Annotations set as specified in section 3.1.1.37, for the target document. When Pid is not equal to 306 and 307 or AnchorText is NOT null, the value of AnchorNumeric MUST be ignored.

AnchorFrequency: When Pid equals 306 or 307 and when AnchorText is NOT null, MUST be the value of AnnotationNumeric column from the Annotations set as specified in section 3.1.1.37, for the target document. When Pid is not equal to 306 and 307 or AnchorText is null, the value of AnchorNumeric MUST be ignored.

LCID: LCID for the AnchorText value. When AnchorText is NULL, LCID MUST be NULL.

LinkId: A unique identifier for the link within the target document.

AnchorText: When pid is 10, AnchorText is the AnchorText attribute of the Anchor Text Info set as specified in section 3.1.1.8 for the TargetDocID item. When pid is 101 or 102, AnchorText is the UserId of the Social Distance Property set as specified in section 3.1.1.11 for the TargetDocID item. When pid is 306 or 307, AnchorText is the Annotation Text of the Annotations set as specified in section 3.1.1.9 for the TargetDocID item. When pid is 2147418050, AnchorText is the title of the TargetDocID item. When pid is 2147418051, AnchorText is the display url of the TargetDocID item. When pid is not 10, 101, 102, 306, 307, 2147418050, and 2147418051, AnchorText must be ignored. This string's length MUST be less than or equal to 1500 Unicode characters.

AnchorBinary: When Pid equals 2147418052, MUST hold data which is from anchor document properties BLOB as specified in section 3.1.1.38. When Pid is not equal to 2147418052, MUST be ignored by the client.

ChangeType: MUST be 1 when all anchor information for the TargetDocID's item has been deleted. MUST be 2 when this row contains anchor information.

InterSite: MUST be 0 when the host name of the SourceDocID is equal to the host name of the TargetDocID. When the host names for SourceDocID and TargetDocID are different, MUST be 1.

3.1.5.63 proc_MSS_IsPauseCrawlsPortalContentDone

The proc_MSS_IsPauseCrawlsPortalContentDone stored procedure is called to identify if all the **portal content** projects are paused. The stored procedure MUST ignore records from Component Activity Set (sections 3.1.1.35) which correspond to crawl components from Crawl Components set with State equal to Disabled and to query components from Query Components set with State equal to Offline.

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

```
PROCEDURE proc_MSS_IsPauseCrawlsPortalContentDone();
```

Return values: An integer which MUST be a value listed in the following table.

Value	Description
1	MUST return this value if all the portal content projects are paused
0	MUST return this value if one or more portal content projects are not paused

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

3.1.5.64 proc_MSS_PrepareForAnnotationsUpdate

The proc_MSS_PrepareForAnnotationsUpdate stored procedure is called at the beginning of the rank calculation portion of a crawl. This stored procedure returns the LastUpdateTime in UTC, which is the maximum UpdateTime in the Annotations set, as specified in section 3.1.1.37, the LastFullUpdateTime in UTC, which is the time when all the records in the Annotations set have been updated, and IsReadyForUpdate, which is a bit that specifies if annotations can be updated without data corruption (1 – annotations can be updated). In addition, when IsReadyForUpdate equals "1", this stored procedure clears the MSSAnnotationsPending table, as specified in [MS-SRCHTP] section 2.2.5.16.

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

```
PROCEDURE proc_MSS_PrepareForAnnotationsUpdate (

    @LastUpdateTime          datetime OUTPUT,
    @LastFullUpdateTime      datetime OUTPUT,
    @IsReadyForUpdate        bit OUTPUT
    );
```

@LastUpdateTime: Upon return of the stored procedure, MUST be set to the value LastUpdateTime from Pending Annotations Status Set (section 3.1.1.10) or MUST be set to NULL if this set is empty

@LastFullUpdateTime: MUST be set to the value LastFullUpdateTime from Pending Annotations Status Set or MUST be set to NULL if this set is empty

@IsReadyForUpdate: MUST be set to "1" if the value Status from the Pending Annotations Status Set equals "1" or "2" if this set is empty and to "0" otherwise

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

Result Sets: MUST NOT return any result sets.

3.1.5.65 proc_MSS_CommitTransactions

The stored procedure **proc_MSS_CommitTransactions** is called to commit a set of transactions (1). If the crawl component from the Crawl Components Status set, as specified in 3.1.1.25, with ComponentID equal to @ComponentID has Status equal to Disabled, the stored procedure MUST stop execution. This stored procedure MUST unpack the @DocRows transaction records from @DocFixedBlob, @DocStrings, and @DocBlobs according to section 2.2.3.19. For each transaction record that is unpacked, the stored procedure MUST call the stored procedure proc_MSS_ProcessCommitted, as specified in section 3.1.5.66, utilizing the results as input parameters.

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

```
PROCEDURE proc_MSS_CommitTransactions(
                  @ComponentID  int,
                  @ApplicationType int,
                  @DocRows int,
                  @DocFixedSize int,
```

```
                    @DocFixedBlob varbinary(max),
                    @DocStrings nvarchar(max),
                    @DocBlobs varbinary(max)
);
```

@ComponentID: Unique identifier of the crawl component.

@ApplicationType: The value of this parameter MUST be 0 and MUST be ignored.

@DocRows: Number of transactions to be committed.

@DocFixedSize: Size, in bytes, of each transaction record, MUST be 245.

@DocFixedBlob: Transaction records as specified in section 2.2.3.19.

@DocStrings: Transaction string as specified in section 2.2.3.22.

@TransactionBlobs: Transaction Data BLOB as specified in section 2.2.3.21.

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

Result Sets: MUST NOT return any result sets.

3.1.5.66 proc_MSS_ProcessCommitted

The **proc_MSS_ProcessCommitted** stored procedure is called to commit a crawl transaction (1).

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

```
PROCEDURE proc_MSS_ProcessCommitted(
    @RecrawlErrorCount              int,
    @ErrorCountAllowed              int,
    @ErrorDeleteCountAllowed        int,
    @DocID                          int,
    @CrawlID                        int,
    @TransactionType                int,
    @Scope                          int,
    @TransactionFlags               int,
    @CompactURL                     nvarchar(40),
    @CompactHash                    int,
    @ParentCompactURL               nvarchar(40),
    @ParentCompactHash              int,
    @DisplayURL                     nvarchar(4000),
    @DisplayHash                    int,
    @LastModifiedTime               bigint,
    @EndPathFlag                    int,
    @PropMD5                        int,
    @MD5                            int,
    @FolderDelCount                 int,
    @HostDepth                      int,
    @EnumerationDepth               int,
    @RetryCount                     int,
    @IndexType                      int,
    @SeqID                          bigint,

    @UseChangeLog                   int,
    @ChangeLogBatchID               int,
```

```
    @ChangeLogCookie                varbinary(8000),
    @ChangeLogCookieType            int,
    @ErrorDesc                      nvarchar(1024),
    @hrResult                       int,
    @DocPropsMD5                    bigint,
    @TransactionStatus              int,
    @CrawlType                      int,
    @DocPropsBlob                   varbinary(2048),
    @ErrorID                        int,
    @ErrorLevel                     int,
    @MarkDelete                     bit,
    @LinksBitmap                    int,
    @Title                          nvarchar(1500),
    @TitleLCID                      int,
    @SecurityId                     nvarchar(40) ,
    @PHFlags                        int ,
    @ProtocolLength                 int,
    @ApplicationType                int,
    @DelayRetryCount                int,
    @LogLevel                       int,
    @ErrorSource                    int,
    @RecrawlErrorInterval           int,
    @ErrorIntervalAllowed           int,
    @ErrorDeleteIntervalAllowed     int,
    @ChangeLogCookieEnd             varbinary(8000),
    @ChildSecurityBlob              varbinary(8000)
);
```

@RecrawlErrorCount: number of times the item SHOULD try to be crawled before the change log cookie MUST be set to NULL.

@ErrorCountAllowed: The value which specifies after what number of errors the document is deleted unless this error is either Access Denied Error or Object Not Found Error and the value of @ErrorDeleteCountAllowed is greater than @ErrorCountAllowed.

@ErrorDeleteCountAllowed: The value which specifies after what number of errors which are either Access Denied Error or Object Not Found Error the document is deleted unless the value of @ErrorCountAllowed is greater than @ErrorDeleteCountAllowed.

@DocID: The identifier of the document.

@CrawlID: The unique identifier of the crawl.

@TransactionType: The type of transaction. The value of this parameter MUST be a Transaction Type data type as specified in Section 2.2.1.14.

@Scope: The transaction scope. The value of this parameter MUST be a Transaction Scope Type as specified in Section 2.2.1.15.

@TransactionFlags: The bit mask of flags for the crawl transaction. The value of this parameter MUST be a valid Transaction Flags type as specified in Section 2.2.2.3.

@CompactURL: The **compact URL** of the item.

@CompactHash: The signature of the CompactURL of the item, as specified in section 3.1.1.5.

@ParentCompactURL: The value of CompactURL of the a item which has a link to the current item

@ParentCompactHash: The signature of ParentCompactURL, The algorithm for computing the signature is implementation details of the protocol client. The signature MUST have the same value given the same URL.

@DisplayURL: The display URL of the item.

@DisplayHash: The hash of the display URL of the item.

@LastModifiedTime: A UTC that specifies when the item was modified.

@EndPathFlag: The value of this parameter MUST be a valid End Path Flag as specified in Section 2.2.2.4.

@PropMD5: The signature of the item metadata, as specified in section 3.1.1.5.

@MD5: The signature of the item content, as specified in section 3.1.1.5.

@FolderDelCount: in case the item is a folder MAY specifies number of deletes which happened inside this folder. When this counter changes all items contained in the folder MUST be checked whether they were deleted or not.

@HostDepth: The value that specifies how many hosts the crawl should traverse when discovering links.

@EnumerationDepth: The value that shows how many links are followed during the crawl.

@RetryCount: Count of times the search service instance tried to process the document.

@IndexType: The value of this parameter MUST be a valid Index Type as specified in Section 2.2.1.20.

@SeqID: The identifier of the record in Crawl Queue as specified in section 3.1.1.4.

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

@ChangeLogBatchID: This parameter MUST be ignored if @UseChangeLog is equal to 0. The identifier of the subset of the change log that the current item belongs to.

@ChangeLogCookie: This parameter MUST be ignored when @UseChangeLog equals 0. When @UseChangeLog equals 1 ChangeLogCookie MUST contain a cookie which was obtained during the latest crawl which processed the item, NULL otherwise, as specified in section 3.1.1.5.

@ChangeLogCookieType: This parameter MUST be ignored when @UseChangeLog equals 0. When @UseChangeLog equals 1 this parameter contains the type of ChangeLogCookie, as specified in section 3.1.1.5.

@ErrorDesc: Description of an error which occurred during document processing, MUST be NULL if there were no errors.

@hrResult: Code of an error which occurred during document processing as specified in section 3.1.1.5. If there were no errors then this parameter will be 0.

@DocPropsMD5: The identifier of the item metadata.

@CrawlType: The type of the crawl. The value of this parameter MUST be a Crawl Type data type as specified in Section 2.2.1.2.

@DocPropsBlob: BLOB containing properties of the document, as specified in section 3.1.1.5.

@ErrorID: Identifier of the error from Crawl Error Set, as specified in [MS-SQLPADM2] section 3.1.1.3, which occurred in process of crawling of this document, 0 if there were no errors.

@ErrorLevel: Level of the error which occurred while crawling this document, as specified in section 3.1.1.5, 0 if there were no errors.

@MarkDelete: The value of this parameter MUST be set to 1 if there is a delete request, and it MUST be set to 0 otherwise.

@LinksBitmap: the crawl stores which have links to this item as specified in section 3.1.1.5.

@Title: Title of the document discovered during the crawl.

@TitleLCID: The language code identifier (LCID) of the @Title.

@SecurityId: Security identifier for the item, as specified in [MS-SQLPQ2] Section 2.2.5.4.

@PHFlags: Flags used by protocol handler. The value of this field is implementation details of the protocol client.

@ProtocolLength: Numbers of characters from the beginning of DisplayURL till the first colon excluding the colon itself.

@ApplicationType: The value of this parameter MUST be 0 and MUST be ignored.

@DelayRetryCount: The count of retries that are caused by errors within components of the crawler. These errors are retried until the document succeeded.

@LogLevel: Error logging level as specified in 2.2.1.9.

@RecrawlErrorInterval: The time, in hours, after which the first error in the ChangeLogCookie MUST be set to NULL.

@ErrorIntervalAllowed: The time, in hours, after which the first error in the document is deleted unless this error is either an Access Denied Error or an Object Not Found Error and the value of **@ErrorDeleteIntervalAllowed** is greater than the value of **@ErrorIntervalAllowed**.

@ErrorDeleteIntervalAllowed: The time, in hours, after which the first error that is either Access Denied or Object Not Found in the document that is marked for deletion because of one of these errors SHOULD be deleted, unless the value of **@ErrorIntervalAllowed** is greater than **@ErrorDeleteIntervalAllowed**.

@TransactionStatus: Contains the status of the transaction.

@ErrorSource: Unique identifier of the internal search component that has reported the error for the item.

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

@ChildSecurityBlob: If the item is a folder, this MAY specify properties of the items in this folder.

Value of @TransactionStatus	Description
3	If @TransactionStatus is equal to 3 (retry) the stored procedure MUST perform

Value of @TransactionStatus	Description

	the following actions:
	▪ The stored procedure MUST update all records in the Crawl URL History (specified in Section 3.1.1.2) that have DocID equal to @DocID and ProjectID equal to 1 (Portal Content). Each updated record MUST have the following values set:
	  ▪ RetryCount set to @RetryCount
	  ▪ DelayRetryCount set to @DelayRetryCount
	▪ If there exists a record in the Crawl Queue in which:
	▪ The values of DocID, CrawlID, TransactionType, Scope, and TransactionFlags equal the corresponding input parameters @DocID, @CrawlID, @TransactionType, @Scope, and @TransactionFlags.
	▪ The value of SeqID not equal to @SecID.
	▪ The stored procedure MUST delete all records from the Crawl Queue in which the values of DocID, CrawlID, and SeqID equal the values of the corresponding input parameters @DocID, @CrawlID, and @SeqID. Otherwise the stored procedure marks this document as ready to crawl by performing the following actions:
	▪ The BatchID in Crawl Queue MUST be set to "0".
	▪ TransactionFlags in the Crawl Queue MUST be set to @TransactionFlags.
	for all records of the Crawl Queue, as specified in Section 3.1.1.4, with DocID, CrawlID, and SeqID equal to the corresponding input parameters @DocID, @CrawlID, and @SeqID.

90	If the following conditions hold:
	▪ @CrawlType is Incremental Crawl
	▪ @ChangeLogCookie is not NULL
	▪ @ErrorLevel is now equal to 2 (Error)
	▪ @ChangeLogCookieEnd is not NULL
	▪ @TransactionFlags does not have flag 0x2000000 (security only transaction) set
	the system MUST perform the following actions:
	▪ for all records in Crawl Url History (3.1.1.2) with the value DocID equal to @DocID, the stored procedure MUST
	  ▪ set values TransactionFlags, ChangeLogCookie, ChangeLogCookieType, ChangeLogCookieEnd to the values @TransactionFlags, @ChangeLogCookie, @ChangeLogCookieType, @ChangeLogCookieEnd
	  ▪ set values RetryCount, DelayRetryCount, Retry to 0
	  ▪ LastTouchStart to UTC

Value of @TransactionStatus	Description

	▪ For all records in the Crawl Queue, as specified in section 3.1.1.4, in which DocID, CrawlID, and SeqID equal the corresponding values @DocID, @CrawlID, and @SeqID, the stored procedure MUST set TransactionFlags to @TransactionFlags and BatchID to "0".
	▪ Upon completion of the preceding, the stored procedure MUST return.
	If there is no record in the Crawl URL History in which:
	▪ DocID is equal to @DocID and either of the following conditions are met:
	  ▪ DeletePending is "0" and @TransactionType is not "1" (Delete).
	  ▪ DeletePending is "1" and @TransactionType is "1" (Delete).
	the stored procedure MUST delete all records from the Crawl Queue Set with DocID, CrawlID, and SeqID equal to @DocID, @CrawlID, and @SeqID and the stored procedure MUST return.
	Otherwise the stored procedure MUST continue with the following sequence of actions. The following text uses the term ProcessingRecord to refer to the records in the Crawl URL History that matched the following criteria:
	▪ DocID is equal to @DocID and either of the following conditions are met:
	  ▪ DeletePending is 0 and @TransactionType is not 1 (Delete)
	  ▪ DeletePending is 1 and @TransactionType is 1 (Delete)
	If the input parameter @MarkDelete is equal to "1", which means that there is a delete request, that is caused by either an Access Denied Error or an Object Not Found Error and there are less than input parameter @ErrorDeleteCountAllowed such errors for the document already, the stored procedure MUST not delete the document. Therefore, the stored procedure MUST set @MarkDelete to "0" and @ResetMarkDelete to "1" if the following conditions are satisfied:
	▪ @MarkDelete is "1".
	▪ CommitCrawlID of the ProcessingRecord is not "0" or @TransactionFlags has flag 0x00000008 set, or @LogLevel equals "2".
	▪ @hrResult is equal to either "0x80041205" (Access Denied) or "0x80041201" (Object Not Found).
	▪ @ErrorDeleteCountAllowed is greater than 0.
	▪ Any of the following conditions are met:
	  ▪ ErrorDeleteCount of the ProcessingRecord is less than @ErrorDeleteCountAllowed.
	  ▪ FirstErrorDeleteTime of the ProcessingRecord is NULL.
	  ▪ Fewer hours elapsed after the FirstErrorDeleteTime of the ProcessingRecord than specified by @ErrorDeleteIntervalAllowed.
	If the input parameter @MarkDelete is not equal to 1, which means that there is no delete request, and @ResetMarkDelete is equal to 0, but there are already more than @ErrorAllowedCount errors for this document and more time has elapsed after the first error than the number of hours specified by

Value of @TransactionStatus	Description
	@ErrorIntervalAllowed, the stored procedure MUST delete the document. Therefore, the stored procedure MUST set @MarkDelete to 1 if all of the following conditions are satisfied: ▪ @MarkDelete is equal to 0. ▪ @ResetMarkDelete is equal to 0. ▪ @ErrorLevel is equal to 2 (Error). ▪ ErrorCount of ProcessingRecord is greater than @ErrorAllowedCount. ▪ FirstErrorTime of ProcessingRecord is not NULL. ▪ More hours elapsed after the FirstErrorTime of the ProcessingRecord than specified by @ErrorIntervalAllowed. In this stored procedure the concept of child objects is used: all documents from the Crawl URL History set with ParentDocID equal to the DocID of the given document are considered to be of the document. All child objects of every child MUST be considered to be child objects of the given document. If the input parameter @MarkDelete is not equal to 1 (there is no delete request) and @ResetMarkDelete is 0 but there are already more than half of @ErrorAllowedCount errors for this document but less than @ErrorAllowedCount the stored procedure reports an error. Therefore, the stored procedure MUST add a record to the Reported Crawl Error set (as specified in Section 3.1.1.16) if the following conditions are satisfied: ▪ @MarkDelete is equal to 0. ▪ @ResetMarkDelete is equal to 0 . ▪ @ErrorLevel is equal to 2 (Error). ▪ ErrorCount of ProcessingRecord is greater than half of @ErrorAllowedCount. ▪ FirstErrorTime of ProcessingRecord is not NULL. ▪ Fewer hours elapsed after the FirstErrorTime of the ProcessingRecord than specified by @ErrorIntervalAllowed. The record added to the Reported Error set MUST have the following values: ▪ DocID set to @DocID. ▪ CrawlID set to @CrawlID. ▪ ErrorID set to @ErrorID. ▪ MarkDelete set to 0. ▪ ChildrenCount set to the number of child objects of the documents with DocID equal to @DocId in Crawl URL History set. ▪ ErrorCount set to ErrorCount of ProcessingRecord. ▪ FirstErrorTime set to FirstErrorTime of ProcessingRecord. If the value of the field AccessURL of ProcessingRecord contains the string

Value of @TransactionStatus	Description
	"anchor:" the stored procedure MUST set @MarkDelete to 1. If @TransactionFlags has flag 0x2000000 (security only transaction) set the stored procedure MUST set @MarkDelete to 0. The stored procedure MUST delete all records from the Deleted URL set as specified in section 3.1.1.3 for which AccessHash and AccessURL are equal to AccessHash and AccessURL of ProcessingRecord respectively. If the following conditions are satisfied: ▪ @CrawlType is 1 (Full Crawl) ▪ @hrResult is equal to 0 (there are no errors) or @hrResult is 0x40D90 (No Index) or @hrResult is 0x0004123A. ▪ @TransactionType is 2 (Modify). ▪ @TransactionFlags has flag 0x00000004 set (this document is a directory). ▪ @TransactionFlags does not have flag 0x2000000 (security only transaction) set. the stored procedure MUST updates records in Crawl URL History set which satisfy the following conditions: ▪ ParentDocID equal to @DocID ▪ TransactionFlags with flag 0x00000200 set and flag 0x00000004 not set ▪ DeletePending equal to 0 (not marked for delete) the stored procedure MUST set ParentCrawlID to @CrawlID for all these records. If the following conditions are satisfied: ▪ @CrawlType is 2 (Incremental Crawl) ▪ @hrResult is equal to 0 (there are no errors) or @hrResult is 0x40D90 (No Index) or @hrResult is 0x0004123A. ▪ @TransactionType is 2 (Modify). ▪ @TransactionFlags has flag 0x00000004 set (this document is a directory). ▪ @TransactionFlags does not have flag 0x2000000 (security only transaction) set. ▪ @ChangeLogCookieis is equal to NULL. the stored procedure updates the records in Crawl URL History based on the value of @UseChangeLog: ▪ If @UseChangeLog is 0 (the transaction does not use a change log), the stored procedure MUST set ParentUpdateCrawlID to @CrawlID for all records in the Crawl URL History set with the following values: ▪ ParentDocID equal to @DocID ▪ TransactionFlags with flag 0x00000200 set and flag 0x00000004 not set

Value of @TransactionStatus	Description
	▪ DeletePending equal to 0 (not marked for delete) ▪ If @UseChangeLog is 1 (the transaction uses a change log), the stored procedure MUST set value ParentDocID to @DocID for all records in the Crawl URL History set with the following values: ▪ ParentDocID equal to @DocID. ▪ TransactionFlags with flag 0x00000004 not set. ▪ DeletePending equal to 0 (not marked for delete) If the following conditions are satisfied: ▪ @CrawlType is equal to 2 (Incremental Crawl). ▪ @hrResult is equal to 0 (there are no errors). ▪ @TransactionType is equal to 2 (Modify). ▪ @UseChangeLog is equal to 1 or @ChangeLogCookieType is greater than 0. ▪ @TransactionFlags has flag 0x00000004 set (the link is a directory). ▪ @TransactionFlags has flag 0x02000000 set (this is a security only transaction) Then the stored procedure MUST add new a record to the Crawl Queue set (specified in Section 3.1.1.4) with flag 0x02000000 set on TransactionFlags, for every record from the Crawl History set that matches the following criterion: ▪ ParentDocID is equal to @DocID . ▪ UseChangeLog is equal to 1. ▪ DeletePending is equal to 0. ▪ SecurityUpdateCrawlID is less then @CrawlID ▪ @TransactionFlags does not have flag 0x0200000 set or CachedSecurityUpdateCrawlID is less that @CrawlID ▪ At least one of the following criterion MUST be met: ▪ TransactionFlags has flags 0x00000200 and 0x00000004 set (directory inside a folder which returns timestamps along with items) or ▪ TransactionFlags has flag 0x00000200 set and flag 0x00000004 not set (the file is inside a folder which returns timestamps along with items and the file was only crawled with an earlier crawl) and CrawlID is less than @CrawlID Each new record added to the Crawl Queue set MUST have the following values: ▪ CrawlID, ChangeLogBatchID, CachedBlob set to @CrawlID, @ChangeLogBatchID, @CachedBlob, respectively. ▪ StartAddressID, ContentSourceID, ProjectID set to the corresponding fields

Value of @TransactionStatus	Description
	of ProcessingRecord. ▪ DocID, HostDepth, EnumerationDepth, ParentDocID set to the corresponding fields of the Crawl URL History Record. ▪ TransactionType set to 2 (Modify). ▪ Scope set to 2 (Deep) ▪ TransactionFlags set to TransactionFlags of Crawl URL History record with flag 0x02000000 added For all records in the Crawl URL History which have value of the field DocID equal to the value of the field DocID of one of the added entries in the Crawl Queue set the stored procedure MUST perform the following update: ▪ if @TransactionFlags does not have flag 0x0200000 set, the stored procedure MUST set SecurityUpdateCrawlID to @CrawlID and SecurityId to NULL for the record from Crawl URL History set. ▪ if @TransactionFlags has flag 0x0200000 set, the stored procedure MUST set CachedSecurityUpdateCrawlID to @CrawlID for the record from Crawl URL History set. If @MarkDelete is equal to 1, the stored procedure MUST add a record to the Crawl Queue set (specified in Section 3.1.1.4) with the following values : ▪ CrawlID, DocID, TransactionFlags, ChangeLogBatchID, ErrorID, ErrorLevel set to the corresponding input parameters @CrawlID, @DocID, @TransactionFlags, @ChangeLogBatchID, @ErrorID, @ErrorLevel. ▪ StartAddressID, ContentSourceID, ProjectID set to values of corresponding fields of ProcessedRecord. ▪ TransactionType set to 1 (Delete). ▪ Scope set to 2 (Deep). ▪ HostDepth, EnumerationDepth and SourceDocID set to 0. If @MarkDelete is equal to 1, the stored procedure MUST set a delete request in Crawl URL History Set (specified in section 3.1.1.2) for all records with DocID equal to @DocID, the stored procedure MUST also set ErrorID, ErrorLevel, ErrorDesc, LogLevel to @ErrorID, @ErrorLevel, @ErrorDesc, @LogLevel for all these records If @TransactionType is "1" (Delete), @Scope equals "2" (Deep), and this document is a folder (@TransactionFlags has 0x00000004 set), for every record from the Crawl URL History set that matches the following: ▪ ParentDocID is equal @DocID. ▪ DeletePending is equal to 0 (there is no delete request). ▪ At least one of the following criterion MUST be met: ▪ This is a file in a folder which returns timestamps along with items (TransactionFlags has flag 0x00000200 set) but not a folder itself (TransactionFlags does not have flag 0x00000004 set)

Value of @TransactionStatus	Description
	▪ This is a file in a folder which returns timestamps along with items (TransactionFlags has flag 0x00000200 set) and it is a folder itself (TransactionFlags has flag 0x00000004 set) and the value UseChangeLog equals 1 the stored procedure MUST add a record to the Crawl Queue set with the following values: ▪ CrawlID, StartAddress, ChangeLogBatchID, ContentSourceID, ProjectID set to @CrawlID, @StartAddress, @ChangeLogBatchID, @ContentSourceID, @ProjectID, respectively ▪ DocID, TransactionFlags, HostDepth, EnumerationDepth set to corresponding fields of the Crawl URL History record ▪ SourceDocID set to ParentDocID of the Crawl URL History record ▪ DeleteReason set to 6 (Delete Recursive) For every Crawl URL History record for which a Crawl Queue record was added, DeletePending of the Crawl URL History record MUST be set to 1. If @TransactionType is 1 (Delete) the stored procedure MUST perform the following sequence of actions: ▪ Add a new record into the Deleted URL set (specified in section 3.1.1.3). ▪ If @AccessUrl does not begin with the string "anchor:", the stored procedure MUST perform all of the following actions: ▪ Add a new record into the Anchor Changed Target Documents set (specified in section 3.1.1.20) with values CrawlID, DocID set to the input parameters @CrawlID, @DocID respectively. ▪ Delete all records from the Link Annotation set (specified in section 3.1.1.9) with TargetDocID equal to @DocID. ▪ Add a new record into the Anchor Changed Deleted Documents set (specified in section 3.1.1.23) with values CrawlID, DocID set to the input parameters @CrawlID, @DocID respectively. ▪ Delete all records from the Anchor Document Property BLOB set (specified in section 3.1.1.38) with DocID equal to @DocID. ▪ If the ParentDocID of the history element equals "-1", the stored procedure MUST delete all records from the MssUserHosts table, as specified in [MS-SRCHTP] Section 2.2.5.19, with the value HostID equal to @HostID. ▪ Delete all records from the Crawl URL History Set with DocID equal to @DocID ▪ Delete all records from the Crawl Queue Set with DocID equal to @DocID and BatchID equal to "0" (not being crawled). ▪ For all records from the Crawl Url History set with ParentDocID equal to @DocID, the stored procedure MUST set ParentDocID to "0". If the following conditions are satisfied:

Value of @TransactionStatus	Description
	▪ @TransactionType is not 1 (Delete) ▪ @MarkDelete is equal to 0 ▪ @TransactionFlags does not have flag 0x02000000 set (this is not security only transaction) ▪ @ErrorLevel is equal to 2 (Error). The stored procedure MUST perform the following actions for all records from the Crawl Queue with the value of the field DocID equal to @DocID: ▪ DocID MUST be set to @DocID. ▪ CrawlID, CommitCrawlID, HostDepth, EnumerationDepth, ChangeLogCookie, ChangeLogCookieType, ErrorID, ErrorLevel, ErrorDesc, DocPropsMD5, CrawlScope, DocPropsBlob, ProtocolLength, ErrorSource MUST be set to the corresponding input parameters @CrawlID, @CommitCrawlID, @HostDepth, @EnumerationDepth, @ChangeLogCookie, @ChangeLogCookieType, @ErrorID, @ErrorLevel, @ErrorDesc, @DocPropsMD5, @CrawlScope, @DocPropsBlob, @ProtocolLength, @ErrorSource respectively. ▪ RetryCount, DelayRetryCount, Retry and LastModifiedTime set to 0. ▪ LastTouchStart MUST be set to the current time. ▪ ChangeLogCookieEnd MUST be set to NULL ▪ LogLevel to LogLevel of history record in case LogLevel of history record is greater than current LogLevel, and should be unchanged otherwise ▪ ErrorCount MUST be set to one more than the ErrorCount of ProcessingRecord. ▪ SecurityUpdateErrorID MUST be set to 0 if @Scope is equal to 2 and @TransactionFlags has flag 0x00000200 set and flag 0x00000004 not set, otherwise SecurityUpdateErrorID MUST be set to SecurityUpdateErrorID of ProcessingRecord. ▪ ErrorDeleteCount set to ErrorDeleteCount of ProcessingRecord if @ResetMarkDelete is equal to 1, otherwise set to 0. ▪ FirstErrorTime set to UTC if FirstErrorTime from ProcessingRecord is NULL, otherwise set to FirstErrorTime of ProcessingRecord. ▪ FirstErrorDeleteTime set to NULL if @ResetMarkDelete is equal to 0, set to UTC if FirstErrorDeleteTime of ProcessingRecord is NULL, otherwise it is set to FirstErrorDeleteTime of ProcessingRecord. If the following conditions are satisfied: ▪ @TransactionType is not 1 (Delete) ▪ @MarkDelete is equal to 0 ▪ @TransactionFlags does not have flag 0x02000000 set (this is not security only transaction)

Value of @TransactionStatus	Description
	▪ @ErrorLevel is not equal to 2 (Error). The stored procedure marks the document as crawled with the identifier @CrawlID. The stored procedure MUST update records from the Crawl Url History, as specified in section 3.1.1.2, with the value of the field DocID equal to @DocID with the following values: ▪ CrawlID, CommitCrawlID, CompactURL, CompactHash, DisplayURL, DisplayHash, TransactionFlags, HostDepth, EnumerationDepth, UseChangeLog, ChangeLogCookie, ChangeLogCookieType, IndexType, MD5, PropMD5, LastModifiedTime, FolderDelCount, EndPathFlag, ErrorID, ErrorLevel, ErrorDesc, DpcPropsMD5, DocPropsBlob, LinksBitmap, TitleLCID, SecurityID, PHFlags, ProtocolLength, LogLevel, CrawlScope, ErrorSource set to the corresponding input parameters @CrawlID, @CommitCrawlID, @CompactURL, @CompactHash, @DisplayURL, @DisplayHash, @TransactionFlags, @HostDepth, @EnumerationDepth, @UseChangeLog, @ChangeLogCookie, @ChangeLogCookieType, @IndexType, @MD5, @PropMD5, @LastModifiedTime, @FolderDelCount, @EndPathFlag, @ErrorID, @ErrorLevel, @ErrorDesc, @DpcPropsMD5, @DocPropsBlob, @LinksBitmap, @TitleLCID, @SecurityID, @PHFlags, @ProtocolLength, @LogLevel, @CrawlScope, and @ ErrorSource respectively. ▪ RetryCount, DelayRetryCount, Retry, ErrorCount, ErrorDeleteCount set to "0". ▪ If ErrorID of ProcessingRecord is equal to NotModifiedErrorID and ErrorLevel of the ProcessingRecord equals "1", the stored procedure MUST set ErrorID to the ErrorID of the ProcessingRecord and ErrorLevel to the ErrorLevel of the ProcessingRecord. ▪ ChangeLogCookieEnd MUST be set to NULL. ▪ LastTouchStart set to the UTC. ▪ Title to the input parameter @Title if it is not an empty string; otherwise set to NULL. ▪ SecurityUpdateErrorID set to "0" when Scope is "2"; otherwise set to the SecurityUpdateErrorID of the ProcessingRecord. ▪ FirstErrorTime and FirstErrorDeleteTime set to NULL. If the following conditions are satisfied: ▪ @ErrorLevel is not equal to "2" (Error). ▪ @hrResult not equal to "0x41203" (Path Not Modified). ▪ AccessURL of ProcessingRecord does not begin with the string "anchor:". The stored procedure MUST insert a record into the Changed Anchor Committed Documents set, as specified in section 3.1.1.22, with the values CrawlID and DocID equal to the input parameters @CrawlID, @DocID respectively. If the following conditions are satisfied: ▪ @CrawlType is 2 (incremental). ▪ @ChangeLogCookie is not NULL.

Value of @TransactionStatus	Description

	▪ ChangeLogCookie of ProcessingRecord is not NULL.
	▪ @ErrorLevel is 2 (Error).
	▪ ErrorCount of ProcessingRecord is greater than @RecrawlErrorCount.
	▪ Either one of the following conditions are met:
	  ▪ More hours elapsed after the FirstErrorTime of the ProcessingRecord than specified by @RecrawlErrorInterval.
	  ▪ @TransactionType is 2 (Modify) and @hrResult is 0x810200BC (Token Too Early)
	Then the stored procedure starts a deep crawl to refresh the document cookie. The stored procedure MUST perform the following actions:
	▪ Set ChangeLogCookie and ChangeLogCookieType to 0 for records in the Crawl URL History set (specified in section 3.1.1.2) with DocID equal to @DocID.
	▪ Add a record to the Crawl Queue set (specified in Section 3.1.1.4) with the following values:
	  ▪ CrawlID, DocID, TransactionFlags, HostDepth, EnumerationDepth, ChangeLogBatchID set to the input parameters @CrawlID, @DocID, @TransactionFlags, @HostDepth, @EnumerationDepth, @ChangeLogBatchID respectively.
	  ▪ StartAddressID, SourceDocID, ContentSourceID, ProjectID set to the values StartAddressID, ParentDocID, ContentSourceID respectively of ProcessingRecord
	  ▪ TransactionType set to 2 (Modify).
	  ▪ Scope set to 2.
	The stored procedure MUST delete all records from the Crawl Queue set (specified in Section 3.1.1.4) with DocID, CrawlID, SeqID equal to the input parameters @DocID, @CrawlID, @SeqID.

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

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

3.1.5.67 proc_MSS_PushSD

The **proc_MSS_PushSD** stored procedure is called to store a new search security descriptor (2).

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

```
PROCEDURE proc_MSS_PushSD(
    @sd                     image,
    @sdCheckSum             int,
    @type                   int,
    @sdid                   int OUTPUT
```

```
);
```

@sd: The search security descriptor (2).

@sdCheckSum: An identifier of a search security descriptor (2).

@type: An integer which MUST be 1 when the search security descriptor (2) is in the format defined in [MS-DTYP] section 2.4.6. Otherwise, it MUST be 0.

@sdid: Upon return from this stored procedure, this parameter MUST be set to a unique identifier of the new search security descriptor (2).

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

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

3.1.5.68 proc_MSS_QLog_GetRelevanceUpdateData

This stored procedure returns **clickthrough** information for the crawl items aggregated over the specified time interval. The structure of the clickthrough data is specified in [MS-SQLPQ2] section 3.1.1.4 (Raw Query Log Set). The protocol server MUST include an item in the result sets if any clicks or skips for that item have been recorded after cut-of time. Number of clicks for a particular URL MUST correspond to the total number of times a user has clicked on this URL in the results page during the time window between startDate and endUpdateDateExcl. Number of skips MUST correspond to the number of times user has clicked on a different URL with the lower rank in the results page, skipping this one during the same time interval. Otherwise the protocol server MUST NOT include the item in the result sets.

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

```
PROCEDURE proc_MSS_QLog_GetRelevanceUpdateData (
    @startDate              smalldatetime,
    @startUpdateDate        smalldatetime,
    @endUpdateDateExcl      smalldatetime
);
```

startDate: The beginning of the time interval in UTC format, included in the time interval

startUpdateDate: Cut-off time in UTC format, included in the time interval

endUpdateDateExcl: The end of time interval in UTC format, excluded from the time interval

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

Result Sets: MUST return the following two result sets:

3.1.5.68.1 proc_MSS_QLog_GetRelevanceUpdateData Result Set 1

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

```
URL:            nvarchar (1024),
URLHash:        int,
QueryString:    nvarchar (1024),
QueryLCID:      int,
```

```
NumClicks:        int;
```

The result set MUST only include records with NumClicks greater than 0.

URL: URL of the item.

URLHash: Unused. MUST be set to 0.

QueryString: Query string, see [MS-SQLPQ2] section 3.1.1.4.

QueryLCID: LCID of the query string.

NumClicks: Number of times the item was clicked on in a search. MUST be greater than or equal to 0.

3.1.5.68.2 proc_MSS_QLog_GetRelevanceUpdateData Result Set 2

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

```
URL:              nvarchar (1024),
URLHash:          int,
NumClicks:        int,
NumSkips:         int,
NumLastClicks:    int;
```

URL: URL of the item.

URLHash: Unused. MUST be set to 0.

NumClicks: Number of times the item was clicked on in a search results view during the specified time interval. MUST be greater than or equal to 0.

NumSkips: Number of times the item was not clicked on in a search results view during the specified time interval. MUST be greater than or equal to 0.

NumLastClicks: Number of times the item was the last item clicked on in a search results view during the specified time interval. For example, result view contained item1, item2 and item3 in this order. If the items clicked on were item3 and item2 in this order, then item2 was the last item clicked on. MUST be greater than or equal to 0.

3.1.5.69 proc_MSS_Recompile

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

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

```
PROCEDURE proc_MSS_Recompile();
```

Return Code Values: An integer which MUST be 0.

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

3.1.5.70 proc_MSS_DefragGathererIndexes

The **proc_MSS_DefragGathererIndexes** stored procedure is called periodically to update statistics and rebuild indexes on the search tables.

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

```
PROCEDURE proc_MSS_DefragGathererIndexes();
```

Return Code Values: An integer which MUST be 0.

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

3.1.5.71 proc_MSS_RemoveConfigTransactions

Search components call this stored procedure on system startup to remove previously uncommitted configuration transactions.

This stored procedure MUST remove all records from Configuration Properties Transactions (section 3.1.1.36) and Uncommitted Transactions of Configuration Properties (section 3.1.1.36).

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

```
PROCEDURE proc_MSS_RemoveConfigTransactions ();
```

Return Value: This stored procedure returns an integer that MUST be ignored.

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

3.1.5.72 proc_MSS_RemoveCrawlStoreFromAllCrawls

The **proc_MSS_RemoveCrawlStoreFromAllCrawls** stored procedure is called to remove all crawls associated with the specified crawl store from the crawl store status. For the specification of Crawl Store Status see Section 3.1.1.28.

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

```
PROCEDURE proc_MSS_RemoveCrawlStoreFromAllCrawls (
    @GthrDBID               int
);
```

@GthrDBID: The ordinal of the crawl store.

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

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

3.1.5.73 proc_MSS_ReportCommandCompleted

The **proc_MSS_ReportCommandCompleted** stored procedure is called to set the Completed value for a command from the commands set (section 3.1.1.41) to true, after the intended object processed the command.

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

```
PROCEDURE proc_MSS_ReportCommandComplete (
   @CommandID              bigint
);
```

@CommandID: A 64-bit integer that uniquely identifies the command (section 3.1.1.41).

When the protocol server processes this stored procedure, it MUST set the Completed value (section 3.1.1.41) of any command in the commands set (section 3.1.1.41), whose CommandID value (section 3.1.1.41) is equal to *@CommandID*, to true.

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

Result Sets: MUST NOT return any result sets.

3.1.5.74 proc_MSS_ResetCatalog

The **proc_MSS_ResetCatalog** stored procedure is called to clear all data from the metadata index. All data must be removed from the Definitions Set as defined in Section 3.1.1.40, MSSDocProps as defined in [MS-SQLPQ2] section 2.2.5.2, MSSDocResults as defined in [MS-SQLPQ2] Section 2.2.5.2 and MSSDocSdids as defined in [MS-SQLPQ2] Section 2.2.5.3.

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

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

@nCatalogId: An integer which MUST be 1.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result set.

3.1.5.75 proc_MSS_SetConfigPropertyWithTransaction

Search components call this stored procedure to add configuration property records.

This stored procedure MUST add a record into Configuration Properties Transactions (section 3.1.1.36) with the values passed in as parameters.

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

```
PROCEDURE proc_MSS_SetConfigPropertyWithTransaction(
    @Name                          nvarchar(300),
    @Value                         sql_variant,
    @TransactionID                 int
```

```
);
```

@Name: Name of the configuration property.

@Value: Value of the configuration property.

@TransactionID: Unique identifier of the uncommitted transactions of configuration properties update (section 3.1.1.36).

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

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

3.1.5.76 proc_MSS_SetConfigurationProperty

The **proc_MSS_SetConfigurationProperty** stored procedure is called to set the value of the specified configuration property or add it if it doesn't exist (see section 3.1.1.36).

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

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

@Name: The name of the configuration property.

@Value: The value of the configuration property.

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

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

3.1.5.77 proc_MSSSetCrawlComponentStatus

The **proc_MSSSetCrawlComponentStatus** stored procedure MUST update the Metadata Store Crawl Components Set defined in section 3.1.1.42. If the @ComponentID parameter matches an existing ComponentID the stored procedure MUST set values Status and ModifiedByComponentID of this record to the corresponding input parameters @Status and @ModifiedByComponentID otherwise it MUST insert a row into the Metadata Store Crawl Components Set corresponding to the @ComponentID, @ModifiedByComponentID and @Status.

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

```
PROCEDURE proc_MSS_SetCrawlComponentStatus (
    @ComponentID                int,
    @ModifiedByComponentID      int,
    @Status                 int
);
```

@ComponentID: Unique identifier of the crawl component.

@ModifiedByComponentID: Unique Identifier of the crawl component that calls this stored procedure.

@Status: Status of the crawl component indicated by **ComponentID**.

Return Code Values: An integer which MUST be 0.

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

3.1.5.78 proc_MSS_SetCrawledPropertyIsSampleCacheFull

The **proc_MSS_SetCrawledPropertyIsSampleCacheFull** stored procedure MUST update the IsSampleCacheFull flag in the sample crawled properties set (defined in [MS-SQLPADM2] section 3.1.1.1) for the specified crawled property.

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

```
PROCEDURE proc_MSS_SetCrawledPropertyIsSampleCacheFull (
    @CrawledPropId                  int,
    @IsSampleCacheFull              bit,
    @UseDateTrigger                 bit
);
```

@CrawledPropId: A unique identifier of a crawled property.

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

@UseDateTrigger: A bit which MUST be 0.

Return Code Values: An integer which MUST be 0.

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

3.1.5.79 proc_MSS_SetRequestToPauseCrawlsPortalContent

The proc_MSS_SetRequestToPauseCrawlsPortalContent stored procedure is called to register a pause request for portal content projects. It MUST set the pause reason on all portal content projects to be 0x00000020, "pause for database refactoring," as specified in section 2.2.2.1

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

```
PROCEDURE proc_MSS_SetRequestToPauseCrawlsPortalContent (
    @Reason int
);
```

@Reason: This parameter MUST be set to the value of 0x00000020, "pause for database refactoring", as specified in section 2.2.2.1

Return values: This stored procedure returns an integer value that MUST be ignored.

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

3.1.5.80 proc_MSS_SetStatusChangeRequest

The proc_MSS_SetStatusChangeRequest stored procedure is called to register a status change for all components in the components activity structure. The status change can be either reset, pause or resume.

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

```
PROCEDURE proc_MSS_SetStatusChangeRequest (
    @HaveResetRequest               int,
    @HavePauseRequest               int,
    @Level                          int,
    @Reason                         int
);
```

@HaveResetRequest: If all components in component activity structure needs to be reset then this parameter MUST be set to 1 otherwise this MUST be set to 0. When this parameter is set to 1 all the other parameters MUST be ignored.

@HavePauseRequest: If master merge or crawl needs to be paused then this parameter MUST be set to 1. If master merge or crawl needs to be resumed then this parameter MUST be set to 0. Master merge or crawl is decided based on the value of the parameter Level 1 or 2 respectively.

@Level: If the status of master merge needs to be changed then this parameter MUST be set to 1. If the status of Crawl needs to be changed then this parameter MUST be set to 2.

@Reason: If the parameter HavePauseRequest is set to 1, then this parameter MUST represent a bitmap of value of type pause reason as specified in [MS-SRCHTP] section 2.2.1.12 that MUST be added to the current pause reason of all the components. If the parameter HavePauseRequest is set to 0, this parameter MUST represent a bitmap of value of type pause reasons as specified in [MS-SRCHTP] section 2.2.1.12 that all the components SHOULD move to.

Return values: An integer which MUST be a value listed in the following table.

Value	Description
1	Status change request was registered successfully
0	Status change request failed to register

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

3.1.5.81 proc_MSS_ShareScopesCompilationInfo

The **proc_MSS_ShareScopesCompilationInfo** stored procedure is used to perform state changes in search scope compilation. The stored procedure MUST perform the following sequence of actions:

- If the specified component is a crawl component, and the ScopeCompilationID of the component in the Crawl Component Set (defined in see [MS-SRCHTP] section 3.1.1.3) is not updated to the specified completed search scope compilation identifier, then the stored procedure MUST set the value of ScopeCompilationID to the specified completed scope compilation identifier. If the specified component is a query component (2), and the ScopeCompilationID of the component in the Query Component Set (defined in [MS-SRCHTP] Section 3.1.1.2) is not updated to the

specified completed search scope compilation identifier, then the stored procedure MUST set the value of ScopeCompilationID to the specified completed scope compilation identifier.

- The stored procedure MUST start search scope compilation if there is no current search scope compilation in progress, the LastChangeID of the search scope in the Scope Set is greater than the LastCompilationID in the Scopes System Set, and the current time is past the NextCompilationTime in the Scopes System Set. For the specification of the Scope Set and the Scope System Set see [MS-SQLPADM2] section 3.1.1.4.

- If the search scope compilation is started, all search scopes that changed after the last search scope compilation, along with their associated search scope rules, and all deleted scopes are chosen for compilation. The stored procedure MUST add a record to the Scopes to Compile Set, as defined in Section 3.1.1.29, for every scope in the Scope Set that has LastChangeID greater than the LastCompilationID of the Scopes System Set. The stored procedure MUST add a record to the Scope Rules to Compile Set, as defined in section 3.1.1.30, for every search scope rule that corresponds to a search scope in the Scopes to Compile Set. The stored procedure MUST add a record to the Deleted Scopes to Compile Set, as defined in Section 3.1.1.32, for every search scope in the Deleted Scopes Set, as defined in Section 3.1.1.31. The stored procedure MUST update the CompilationChangeID to the LastChangeID of the Scopes System Set.

- If there is no current search scope compilation in progress, and if the LastChangeID if less than or equal to the LastCompilationID in the Scopes System Set, the stored procedure MUST set the LastCompilationTime to the current time and update the NextCompilationTime to the next time a scope compilation must be started.

- The stored procedure MUST complete search scope compilation for a standalone component if there is a compilation in progress and the specified completed compilation identifier is greater than or equal to the CompilationChangeID of the Scopes System Set. The stored procedure MUST complete search scope compilation for all query components and crawl components in a Ready State if their ScopeCompilationID is greater than or equal to the CompilationChangeID of the Scopes System Set.

- If the scope compilation is to be completed, the stored procedure MUST set the Compilation State to Compiled (Compilation State type 4 in [MS-SQLPADM2] Section 2.2.1.4) for all the search scopes in the Scopes Set that were compiled. The stored procedure MUST remove all deleted scopes of the completed compilation from the Deleted Scopes Set and the Compiled Scopes Set (defined in Section 3.1.1.33), and the corresponding search scope rules from the Compiled Scope Rules Set (defined in Section 3.1.1.34). The stored procedure MUST update the search scopes in the Scopes Set to the corresponding scopes in the Scopes to Compile Set. The stored procedure MUST update the search scope rules in the Scope Rules Set to the corresponding scope rules in the Scope Rules to Compile Set. The stored procedure MUST update the LastConsumerChangeID in the Scopes System Set by 1, and set the LastUpdate of the consumer to that value, for all consumers in Consumer Set (defined in see [MS-SQLPADM2] Section 3.1.1.4) that correspond to search scopes that were compiled. The stored procedure MUST update the NextCompilationTime of the Scopes System Set and set the LastCompilationTime to the current time. The stored procedure MUST finish the completion of compilation by clearing all data from the Scopes to Compile Set, the Scope Rules to Compile Set and the Deleted Scopes to Compile Set.

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

```
PROCEDURE proc_MSS_ShareScopeCompilationInfo (
    @ComponentRole                int,
    @ComponentID                  int,
    @CompletedCompilationID       int,
    @RequestedCompilationID       int OUTPUT
```

```
);
```

ComponentRole: Type of the component. The value MUST be in the component type table in [MS-SRCHTP] section 2.2.1.12.

ComponentID: The identifier of the search component.

CompletedCompilationID: The search scope compilation identifier of the last completed search scope compilation for the specified search component.

RequestedCompilationID: Upon return from this stored procedure, this parameter MUST be set to the CompilationChangeID of the Scopes System Set if there is a scope compilation in progress which is currently compiling for crawl components and the specified component is not a query component, or if there is a scope compilation in progress which is currently compiling for query components. In all other cases, this parameter MUST be set to the LastCompilationID of the Scopes System Set.

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

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

3.1.5.82 proc_MSS_TruncateCleanupTable

The **proc_MSS_TruncateCleanupTable** stored procedure is called to reset counts of crawled property samples associated with crawled property IDs which are used to adjust the IsSampleCacheFull property in the crawled property set (as specified in [MS-SQLPADM2] section 3.1.1.1) of the metadata index.

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

```
PROCEDURE proc_MSS_ TruncateCleanupTable();
```

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

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

3.1.5.83 proc_MSS_UnloadTransactions

This stored procedure operates on the records from the Crawl Queue (section 3.1.1.4). It marks all the records which match a given BatchID value as not being crawled. A record of the Crawl Queue indicates a document which is not currently crawled when the value of BatchId field is 0.

This stored procedure MUST set Batch Identifier to "0" for all records from the Crawl Queue, as specified in section 3.1.1.4, with Batch Identifier equal to @BatchID.

If crawl component with identifier @ComponentID does not exist or is disabled in Local Crawl Components Set (section 3.1.1.26) the stored procedure MUST raise an error.

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

```
PROCEDURE proc_MSS_UnloadTransactions(
    @ComponentID                int,
    @BatchID                    bigin
```

```
);
```

@ComponentID: Unique identifier of the crawl component.

@BatchID: Unique identifier of crawl batch.

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

Result Sets: MUST NOT return any result sets.

3.1.5.84 proc_MSS_SetAnnotationsPendingStatus

The proc_MSS_SetAnnotationsPendingStatus stored procedure adds and updates records in the Pending Annotations Status set (section 3.1.1.10).

If Pending Annotations Status set is empty the stored procedure MUST insert new record into this set with the following values:

- ObsoleteTime is set to @ObsoleteTime input value
- LastUpdateTime and LastFullUpdate time are set to @UpdateTime input value
- Status is set to @Status input value

If Pending Annotations Status set is not empty the stored procedure MUST update all the records in this set with the following values

- ObsoleteTime is set to @ObsoleteTime input value
- LastUpdateTime time is set to @UpdateTime input value
- LastFullUpdateTime is set to @UpdateTime input value if @Status equals 2 and is not changed otherwise
- Status is set to @Status input value

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

```
PROCEDURE proc_MSS_SetAnnotationsPendingStatus (
    @ObsoleteTime              datetime,
    @UpdateTime                datetime,
    @Status                    int
);
```

@ObsoleteTime: UTC time which states which annotations in the Annotations set are considered obsolete: records in Annotation Set with UpdateTime smaller than ObsoleteTime are considered obsolete

@UpdateTime: UTC which is the time when one of the records in the annotations set has been updated: largest value of UpdateTime in Annotations Set

@Status: status of pending annotations

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

Result Sets: MUST NOT return any result sets.

3.1.5.85 proc_MSS_TruncateAnnotationsTables

The stored procedure proc_MSS_TruncateAnnotationsTables MUST remove all record from Pending Annotations set (section 3.1.1.9) and Pending Annotations Status set (section 3.1.1.10).

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

```
PROCEDURE proc_MSS_TruncateAnnotationsTables ();
```

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

Result Sets: MUST NOT return any result sets.

3.1.5.86 proc_MSS_UpdateAnchorDocPropsBlob

The proc_MSS_UpdateAnchorDocPropsBlob stored procedure adds, deletes and updates records in the Anchor Document Properties BLOB set as specified in section 3.1.1.38. The parameter addRows MUST contain the number of Update Anchor Add Records defined in section 2.2.3.1 that are contained in the addBlob parameter. The addBlob parameter MUST contain the offset and length of each propertyTrackerBlob in the stream of propertyTrackerBlobs within the parameter addDataBlob.

For each propertyTrackerBlob the stored procedure MUST add, update or delete data from the Anchor Document Properties BLOB set as specified in section 3.1.1.38 using the Update Anchor Add Records as described in section 2.2.3.1.

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

```
PROCEDURE proc_MSS_UpdateAnchorDocPropsBlob (
    @trace                  int,
    @addRows                int,
    @addBlob                varbinary(max),
    @addDataBlob            varbinary(max)
);
```

@trace: MUST be ignored.

@addRows: MUST be the number of Update Anchor Add records as specified in section 2.2.3.1 within the Update Anchor Add Structure as specified in section 2.2.3.2 stored addBlob parameter. MUST NOT be NULL and MUST be greater than 0.

@addBlob: The Update Anchor Add Structure section 2.2.3.2 is used to update, insert or delete records in the Anchor Document Properties BLOB set section 3.1.1.38.

@addDataBlob: The stream of propertyTrackerBlobs in the anchor document properties BLOB set section 3.1.1.38.

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

Result Sets: MUST NOT return any result sets.

3.1.5.87 proc_MSS_UpdateCrawlStatistics

This stored procedure is used to update crawl component statistics (SuccessCount, WarningCount, ErrorCount, DeleteCount, NotModifiedCount, SecurityOnlyErrorCount, RetryCount) (section

3.1.1.25). This stored procedure MUST updates crawl statistic part of crawl component status, each value MUST be updated by adding corresponding non negative input value.

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

```
PROCEDURE proc_MSS_UpdateCrawlStatistics(
    @ComponentID                int,
    @CrawlID                    int,
    @SuccessCount               int,
    @WarningCount               int,
    @ErrorCount                 int,
    @DeleteCount                int,
    @NotModifiedCount           int,
    @SecurityOnlyCount          int,
    @SecurityOnlyErrorCount     int,
    @LevelHighErrorCount        int,
    @RetryCount                 int
);
```

@ComponentID: unique identifier of crawl component.

@CrawlID: unique identifier of an ongoing crawl the component with id @ComponentID is participating in

@SuccessCount: a nonnegative value which MUST be added to SuccessCount within the crawl statistics of Crawl Component Status Set for the ComponentID and CrawlID passed in.

@WarningCount: a nonnegative value which MUST be added to WarningCount within the crawl statistics of Crawl Component Status Set for the ComponentID and CrawlID passed in.

@ErrorCount: a nonnegative value which MUST be added to ErrorCount within the crawl statistics of Crawl Component Status Set for the ComponentID and CrawlID passed in.

@DeleteCount: a nonnegative value which MUST be added to DeleteCount within the crawl statistics of Crawl Component Status Set for the ComponentID and CrawlID passed in.

@NotModifiedCount: a nonnegative value which MUST be added to NotModifiedCount within the crawl statistics of Crawl Component Status Set for the ComponentID and CrawlID passed in.

@SecurityOnlyCount: a nonnegative value which MUST be added to SecurityOnlyCount within the crawl statistics of Crawl Component Status Set for the ComponentID and CrawlID passed in.

@SecurityOnlyErrorCount: a nonnegative value which MUST be added to SecurityOnlyErrorCount within the crawl statistics of Crawl Component Status Set for the ComponentID and CrawlID passed in.

@LevelHighErrorCount: a nonnegative value which MUST be added to LevelHighErrorCount within the crawl statistics of Crawl Component Status Set for the ComponentID and CrawlID passed in.

@RetryCount: a nonnegative value which MUST be added to RetryCount within the crawl statistics of Crawl Component Status Set for the ComponentID and CrawlID passed in

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

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

3.1.5.88 proc_MSS_UpdateDocCount

This stored procedure MUST set the count of the documents in the crawl store with Ordinal equal to the input parameter @GthrDBID, as specified in [MS-SRCHTP] section 3.1.1.3, to @DocCount.

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

```
PROCEDURE proc_MSS_UpdateDocCount(
    @GthrDBID           int,
    @DocCount           int
);
```

@GthrDBID: Unique identifier of crawl store

@DocCount: Number of documents in crawl store with unique identifier @GthrDBID

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

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

3.1.5.89 proc_MSS_UpdateLinksBitmapField

The **proc_MSS_UpdateLinksBitmap** stored procedure is called to update the values of LinksBitmap fields of Crawl URL History items as specified in section 3.1.1.2. The procedure MUST update LinksBitmap field of each Crawl URL History item which has at least one common bit set to 1 in both LinksBitmap field and the value passed in.

All of the bits in LinksBitmap passed in that are set to 1 MUST be set to 1 in the Crawl URL History item LinksBitmap. Bits in the Crawl URL History item LinksBitmap that were set to 1 prior to the call of this store procedure MUST continue to be set to 1. Other bits MUST be set to 0.

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

```
PROCEDURE proc_MSS_UpdateLinksBitmapField (
    @LinksBitmap                    int
);
```

@LinksBitmap: The bitmap of crawl store identifiers.

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

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

3.1.5.90 proc_MSS_UpdatePropertyStore

The proc_MSS_UpdatePropertyStore stored procedure is used to add, update and delete rows in the metadata store. The ComponentID must match the ComponentID of a row in the Metadata Store Crawl Components defined in section 3.1.1.42 and the status of the crawl component MUST be 1 otherwise the stored procedure returns immediately without updating anything. When the @queueType is equal to 2, all data for the documents updated in the MSSDocProps table specified in [MS-SQLPQ2] section 2.2.5.2, the MSSDocResults table specified in [MS-SQLPQ2] section 2.2.5.3, the MSSDocSdids table specified in [MS-SQLPQ2] section 2.2.5.4, and the Definitions set specified in section 3.1.1.40 MUST be deleted before data for the respective tables are re-added.

When the @queueType as specified in section 2.2.1.18 is either 1 or 2, and when the urlSignature within the Alert History Update Record as specified in section 2.2.3.15 has not been seen before, then a row is inserted into the MSSDocProps table with the DocId of the Alert History Update Record, a pid equal to 321, llVal equal to the CrawlTime of the Alert History Update Record and strVal and strVal2 equal to null and a second row is inserted into the MSSDocProps table with the DocId of the Alert History Update Record, a pid equal to 322, llVal equal to the CrawlTime of the Alert History Update Record and strVal and strVal2 equal to null. When the schemaSignature of the Alert History Update Record for a document has not changed and the contentSignature has changed, then the row in the MSSDocProps table matching the document's DocId and pid equal to 322 should be updated with the llVal column equal to the crawlTime from the Alert History Update Record, however when the schemaSignature for a document has changed or the contentSignature has not changed, then the row in MSSDocProps matching the DocId with pid equal to 322 should have an llVal column value equal to the value of the crawlTime from the original insert for that document or from the last crawlTime that the schemaSignature did not change and the contentSignature did change.

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

```
PROCEDURE proc_MSS_UpdatePropertyStore (
    @trace                          int,
    @ComponentID                    int,
    @queueType                      int,
    @deleteDocIdRows                int,
    @deleteDocIdBlob                varbinary(max),
    @addRows                        int,
    @addBlob                        varbinary(max),
    @addStringBlob                  nvarchar(max),
    @updateRows                     int,
    @updateBlob                     varbinary(max),
    @updateStringBlob               nvarchar(max),
    @deleteRows                     int,
    @deleteBlob                     varbinary(max),
    @deleteRowsWithHash             int,
    @deleteWithHashBlob             varbinary(max),
    @docResultsRows1                int,
    @docResultsBlob1                varbinary(max),
    @docResultsStringBlob1          nvarchar(max),
    @docResultsBinaryBlob1          varbinary(max),
    @docResultsRowsAnchor           int,
    @docResultsBlobAnchor           varbinary(max),
    @docResultsStringBlobAnchor     nvarchar(max),
    @docSdIdsRows                   int,
    @docSdIdsBlob                   varbinary(max),
    @alertHistoryRows               int,
    @alertHistoryBlob               varbinary(max),
    @definitionRows                 int,
    @definitionBlob                 varbinary(max),
    @definitionStringBlob           nvarchar(max)
);
```

@trace: This parameter MUST be ignored.

@ComponentID: The unique identifier of the crawl component.

@queueType: Update queue type as specified in section 2.2.1.18.

@deleteDocIdRows: MUST be the number of document identifiers within the Id Blob structure specified in [MS-SQLPQ2] section 2.2.1.2 stored deleteDocIdBlob parameter. MUST NOT be NULL.

@deleteDocIdBlob: Id Blob structure specified in [MS-SQLPQ2] section 2.2.1.2 containing an array of document identifiers of documents whose data should be removed from the metadata store.

@addRows: MUST be the number of DocProps Add records, as specified in section 2.2.3.3, within the addBlob specified in section 2.2.3.4. MUST NOT be null.

@addBlob: MUST be a DocProps Add structure, as specified in section 2.2.3.4, which is used to add rows to the mssDocProps table specified in [MS-SQLPQ2] section 2.2.5.2 using strings parsed from the addStringBlob as specified in section 2.2.3.4. MUST be ignored when addRows is 0.

@addStringBlob: A string of Unicode characters which is separated into strings using offsets and lengths in the addBlob structure specified in section 2.2.3.4.

@updateRows: MUST be the number of DocProps Add records, as specified in section 2.2.3.3, within updateBlob specified in section 2.2.3.4. MUST NOT be null.

@updateBlob: MUST be a DocProps Add structure, as section 2.2.3.4, which is used to update rows in the MSSDocProps table specified in [MS-SQLPQ2] section 2.2.5.2 using strings parsed from the updateStringBlob as specified in section 2.2.3.4. MUST be ignored when updateRows is 0.

@updateStringBlob: A string of Unicode characters which is separated into strings using offsets and lengths in the updateBlob structure specified in section 2.2.3.4.

@deleteRows: MUST be the number of DocProps Delete Records specified in section 2.2.3.5 within the DocProps Delete Structure specified in section 2.2.3.6. MUST NOT be null.

@deleteBlob: MUST be a DocProps Delete Structure specified in section 2.2.3.6 which is used to delete all the values for a property which are contained in 1 or more rows within a document in the MSSDocProps table specified in [MS-SQLPQ2] section 2.2.5.2.

@deleteRowsWithHash: MUST be the number of DocProps Delete With Hash Records specified in section 2.2.3.7 within the DocProps Delete With Hash Structure specified in section 2.2.3.8. MUST NOT be null.

@deleteWithHashBlob: MUST be a DocProps Delete With Hash Structure specified in section 2.2.3.8 which is used to delete a value or values of a property for a document in the MSSDocProps table specified in [MS-SQLPQ2] section 2.2.5.2.

@docResultsRows1: MUST be the number of DocResults Update Records specified in section 2.2.3.9 within the DocResults Update Structure specified in section 2.2.3.10. MUST NOT be null.

@docResultsBlob1: MUST be a DocResults Update Structure specified in section 2.2.3.10 which is used to update rows in the MSSDocResults table specified in [MS-SQLPQ2] section 2.2.5.2.

@docResultsStringBlob1: A string of Unicode characters which is separated into strings using offsets and lengths in docResultsBlob1 specified in section 2.2.3.10.

@docResultsBinaryBlob1: A stream of bytes which is separated into separate property values using the offsets and lengths in the DocResults Update structure specified in section 2.2.3.10.

@docResultsRowsAnchor: MUST be the number of DocResults Anchor Update Records specified in section 2.2.3.11 within the DocResults Anchor Update Structure specified in section 2.2.3.12. MUST NOT be null.

@docResultsBlobAnchor: MUST be a DocResults Anchor Update Structure specified in section 2.2.3.12 which is used to update rows in the MSSDocResults table specified in [MS-SQLPQ2] section 2.2.5.2.

@docResultsStringBlobAnchor: A string of Unicode characters which is separated into strings using offsets and lengths in docResultsBlobAnchor specified in section 2.2.3.12.

@docSdIdsRows: MUST be the number of DocSdIds Update Records specified in section 2.2.3.13 within the DocSdIds Update Structure specified in section 2.2.3.14. MUST NOT be null.

@docSdIdsBlob: MUST be a DocSdIds Update Structure specified in section 2.2.3.14 which is used to update rows in the MSSDocSdIds table specified in [MS-SQLPQ2] section 2.2.5.3.

@alertHistoryRows: MUST be the number of Alert History Update Records specified in section 2.2.3.15 within the Alert History Update Structure specified in section 2.2.3.16. MUST NOT be null.

@alertHistoryBlob: MUST be an Alert History Update Structure specified in section 2.2.3.16 when alertHistoryRows is greater than 0. MUST be ignored when alertHistoryRows is equal to 0.

@definitionRows: MUST be the number of Definitions Update Records specified in section 2.2.3.17 within the Definitions Update Structure specified in section 2.2.3.18. MUST NOT be null.

@definitionBlob: MUST be a Definitions Update Structure specified in section 2.2.3.18 which is used to update records in the Definitions set specified in section 3.1.1.40.

@definitionStringBlob: A string of Unicode characters which is separated into strings using offsets and lengths in definitionsBlob specified in section 2.2.3.18.

3.1.5.91 proc_MSS_CleanupWithInterval

The **proc_MSS_CleanupWithInterval** stored procedure is called to clear all the crawl data for which the **crawl status** is in one of the done, forbid or stopped state (section 2.2.1.4) and crawl is older than the parameter CleanupInterval time.

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

```
PROCEDURE proc_MSS_CleanupWithInterval (
    @CleanupInterval        int
);
```

@CleanupInterval: Specifies the number of days between in cleanup interval.

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

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

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Office SharePoint Server Client Details

The protocol client role is described in this section. The client role can make requests for crawl operations.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

None.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

This section describes sequences performed on the client as part of this protocol. The following sequences are given here:

- Full crawl
- Incremental crawl
- Delete crawl
- Status changes
- Configuration property modification
- Scope compilation
- Request master merge

3.2.5.1 Full Crawl Sequence

This section contains the sequence used to crawl all items in a content source. The specific sequence is as follows:

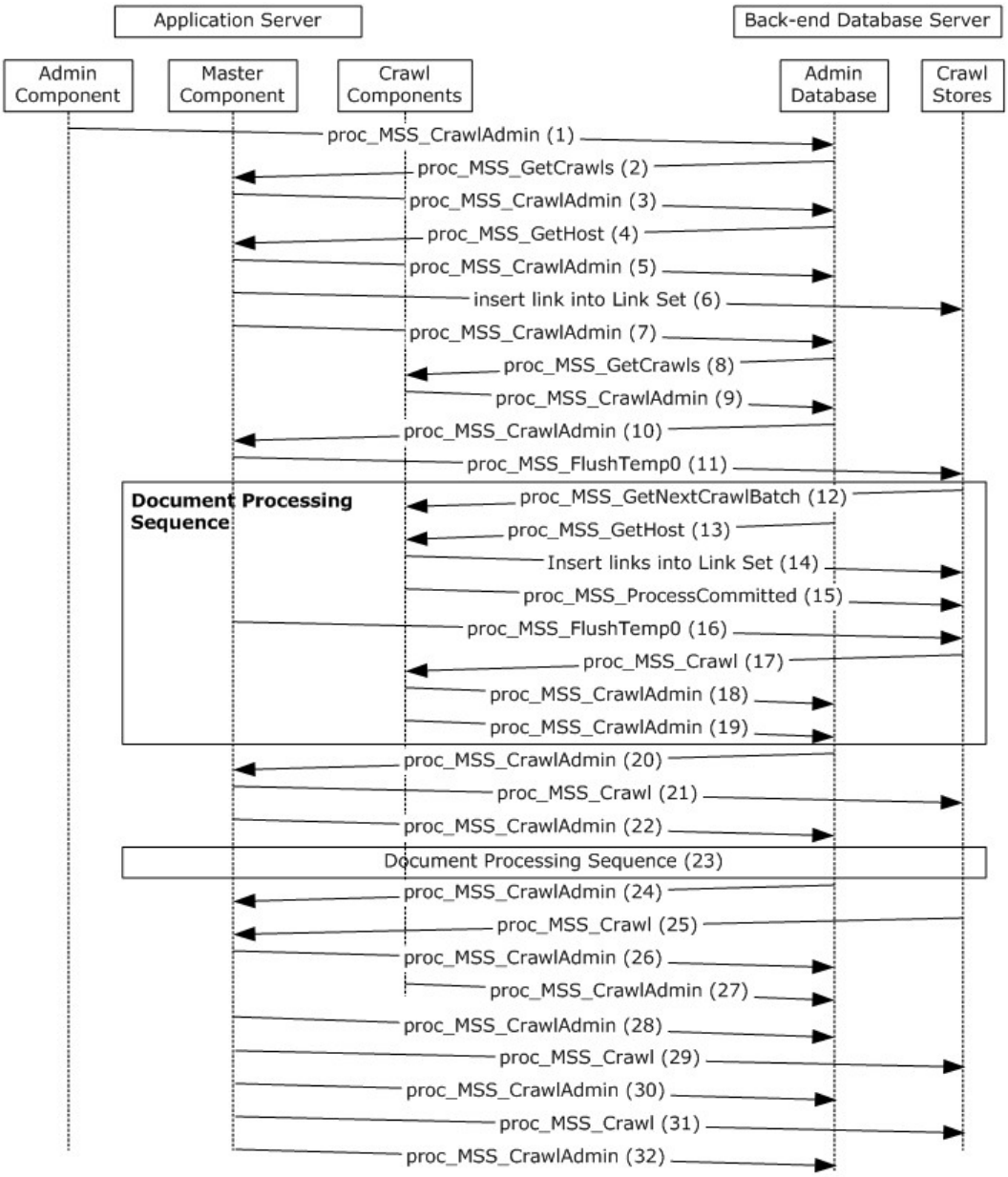


Figure 5: Full crawl sequence diagram

The steps in the drawing are described here:

1. Admin component calls proc_MSS_CrawlAdmin with @CrawlStage equals 100 to start the crawl; get CrawlID

2. Master component calls proc_MSS_GetCrawls with @CatalogID equals "Portal Content" to get list of crawls

3. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 102 to set content source for newly created crawl

4. Master component calls proc_MSS_GetHost to get Crawl Store Identifier which will process given host

5. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 108, to inform that all crawl components from the crawl store which process the host MUST join the crawl

6. Master component insert link into Link Set of the selected Crawl Store

7. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 103, notifies that start addresses were successfully added and now waits for crawl components

8. Crawl components call proc_MSS_GetCrawls to check if there any crawls they need to join

9. Crawl components call proc_MSS_CrawlAdmin with @CrawlState equals 16 to notify that they are can start crawling

10. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 104 to check if all crawl components done with starting the crawl

11. Master component call proc_MSS_FlushTemp0 to move data from Links Table to Crawl Queue

12. Crawl components call proc_MSS_GetNextCrawlBatch to assign some not yet crawled documents from Crawl Queue as being crawled by this crawl component and return those documents

13. Crawl components call proc_MSS_GetHost to get identifier of the crawl store which will process the newly discovered link

14. Crawl components insert newly discovered links into Link Set

15. Crawl components call proc_MSS_ProcessCommitted for every link to mark it as processed

16. Master component calls proc_MSS_FlushTemp0 to move data from Links Set to Crawl Queue and GOTO 11

17. Crawl components call proc_MSS_Crawl with @CrawlStage equals 141 to check if it is done with the crawl

18. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 142 to inform that it is not doing anything (if it is not doing anything)

19. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 143 to inform that it is not doing anything (if it is doing anything)

20. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 140 to check if crawl components have finished with first part and if yes move crawl to Unvisited to Queue stage

21. Master component calls proc_MSS_Crawl with @CrawlStage equals 145 to start next crawl stage

22. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 144 to start Delete Unvisited crawl stage

23. Crawl components perform Documents Processing Sequence

24. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 140 to check if crawl components have finished with that part of the crawl and if yes move crawl to Wait for All Databases Stage

25. Master component calls proc_MSS_Crawl with @CrawlStage equals 151 against all crawl stored to check if the all crawl stored have finished with the crawl

26. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 150 to start crawl completion; sets sub status to Wait For Crawl Components

27. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 147 to inform that they have finished with the crawl

28. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 146 to check if all crawl components have finished with the crawl and if yes update Crawl History Set; sets sub status to Substatus Completing

29. Master component calls proc_MSS_Crawl with @CrawlStage equals 149 against all crawl stores to update crawl statistics and prepare datasets for anchor text crawl

30. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 152; sets sub status to Delete Pending

31. Master component calls proc_MSS_Crawl with @CrawlStage equals 153

32. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 148 to report that crawl is done is to start anchor text crawl

3.2.5.2 Incremental Crawl Sequence

This section contains the sequence that is used to perform another crawl of all items that have changed in a content source after the last crawl. The specific sequence is as follows:

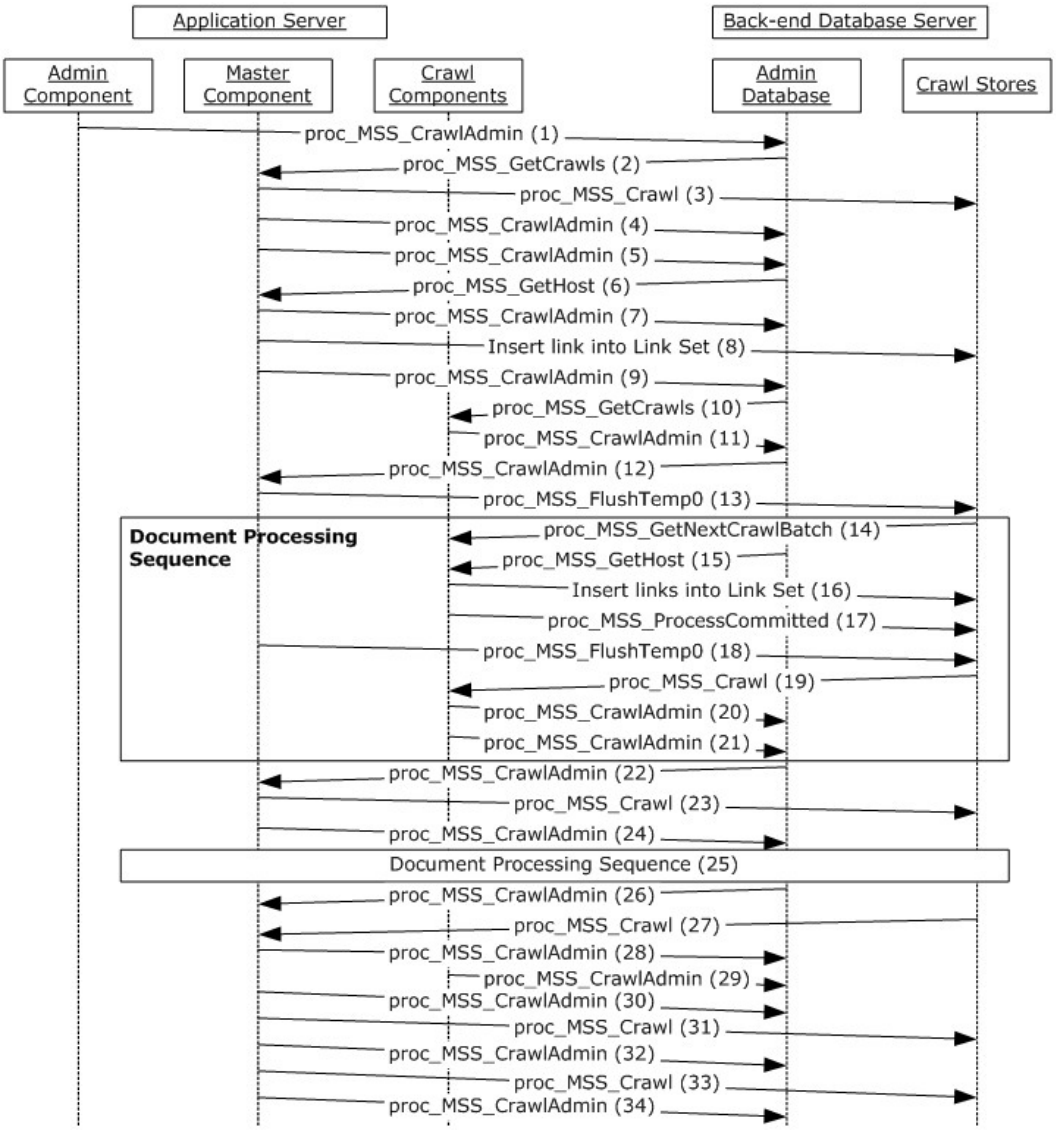


Figure 6: Incremental crawl sequence diagram

The steps in the drawing are described here:

1. Admin component calls proc_MSS_CrawlAdmin with @CrawlStage equals 100 to start the crawl; get CrawlID

2. Master component calls proc_MSS_GetCrawls with @CatalogID equals "Portal Content" to get list of crawls

3. Master component calls proc_MSS_Crawl with @CrawlStage equals 109 against all crawl stores to move documents from Crawl History Set to Crawl Queue

4. If documents were copied in previous statement master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 108 to notify that all crawl components from specified crawl store MUST join the crawl

5. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 102 to set content source for newly created crawl

6. Master component calls proc_MSS_GetHost to get Crawl Store Identifier which will process given host

7. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 108, to inform that all crawl components from the crawl store which process the host MUST join the crawl

8. Master component insert link into Link Set of the selected Crawl Store

9. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 103, notifies that start addresses were successfully added and now waits for crawl components

10. Crawl components call proc_MSS_GetCrawls to check if there any crawls they need to join

11. Crawl components call proc_MSS_CrawlAdmin with @CrawlState equals 16 to notify that they are can start crawling

12. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 104 to check if all crawl components done with starting the crawl

13. Master component call proc_MSS_FlushTemp0 to move data from Links Table to Crawl Queue

14. Crawl components call proc_MSS_GetNextCrawlBatch to assign some not yet crawled documents from Crawl Queue as being crawled by this crawl component and return those documents

15. Crawl components call proc_MSS_GetHost to get identifier of the crawl store which will process the newly discovered link

16. Crawl components insert newly discovered links into Link Set

17. Crawl components call proc_MSS_ProcessCommitted for every link to mark it as processed

18. Master component calls proc_MSS_FlushTemp0 to move data from Links Table to Crawl Queue and GOTO 11

19. Crawl components call proc_MSS_Crawl with @CrawlStage equals 141 to check if it is done with the crawl

20. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 142 to inform that it is not doing anything (if it is not doing anything)

21. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 143 to inform that it is not doing anything (if it is doing anything)

22. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 140 to check if crawl components have finished with first part and if yes move crawl to Unvisited to Queue stage

23. Master component calls proc_MSS_Crawl with @CrawlStage equals 145 to start next crawl stage

24. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 144 to start Delete Unvisited crawl stage

25. Crawl components perform Documents Processing Sequence

26. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 140 to check if crawl components have finished with that part of the crawl and if yes move crawl to Wait for All Databases Stage

27. Master component calls proc_MSS_Crawl with @CrawlStage equals 151 against all crawl stored to check if the all crawl stored have finished with the crawl

28. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 150 to start crawl completion; sets sub status to Wait For Crawl Components

29. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 147 to inform that they have finished with the crawl

30. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 146 to check if all crawl components done with the crawl and if yes update Crawl History Set; sets sub status to Substatus Completing

31. Master component calls proc_MSS_Crawl with @CrawlStage equals 149 against all crawl stores to update crawl statistics and prepare datasets for anchor text crawl

32. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 152; sets sub status to Delete Pending

33. Master component calls proc_MSS_Crawl with @CrawlStage equals 153

34. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 148 to report that crawl is done is to start anchor text crawl

3.2.5.3 Delete Crawl Sequence

This section contains the crawl sequence used to delete items from the full-text index catalog and metadata index when the items were removed from an existing content source. The specific sequence is as follows:

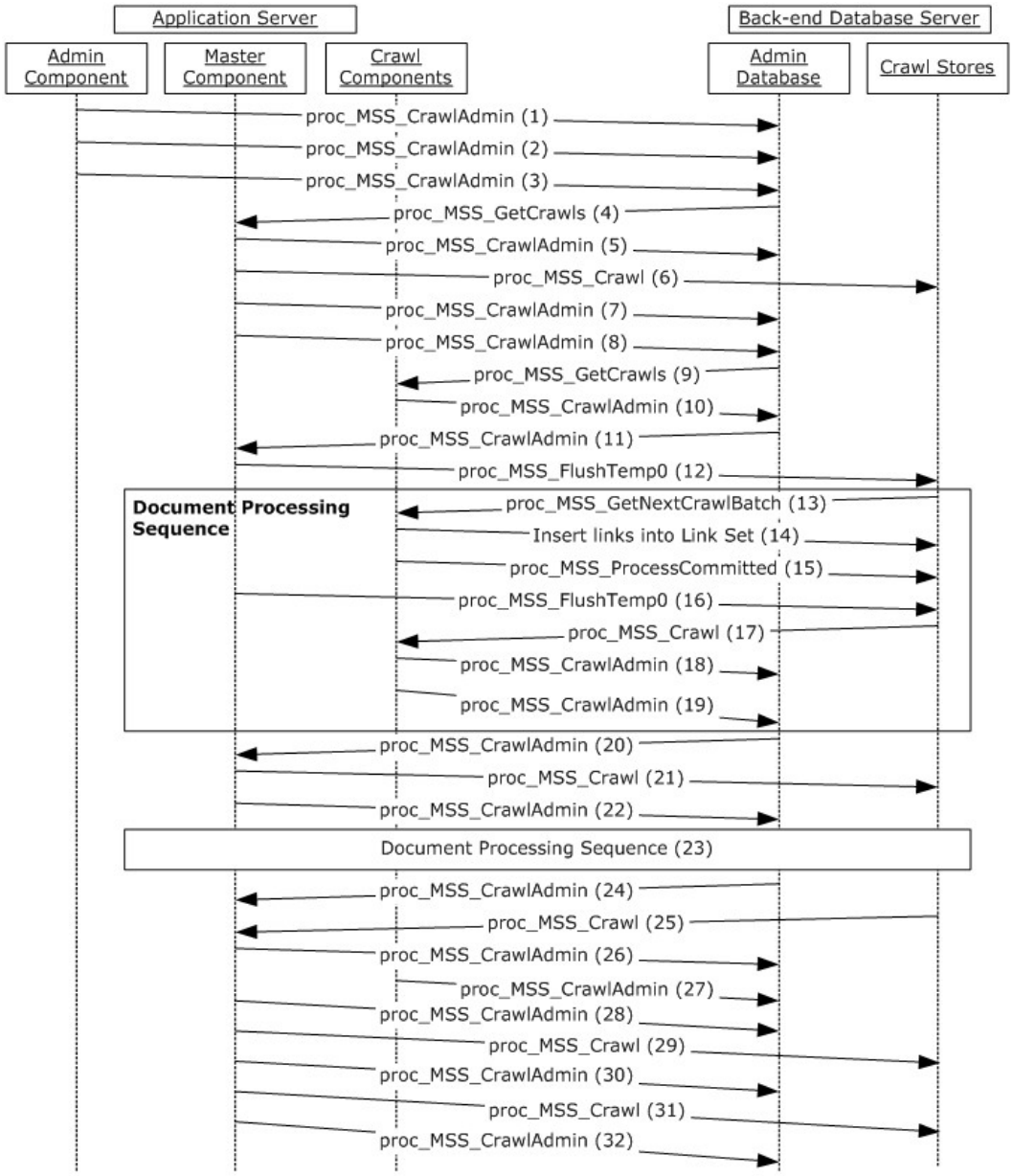


Figure 7: Delete crawl sequence diagram

The steps in the drawing are described here:

1. Admin component calls proc_MSS_CrawlAdmin with @CrawlStage equals 160 to delete entire content source; new record is added into Requested Delete Crawls Set

2. Admin component calls proc_MSS_CrawlAdmin with @CrawlStage equals 161 to delete start address inside content source; new record is added into Requested Delete Crawls Set

3. Admin component calls proc_MSS_CrawlAdmin with @CrawlStage equals 164

4. Master component calls proc_MSS_GetCrawls with @CatalogID equals "Portal Content" to get list of crawls

5. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 102 to set content source for newly created crawl

6. Master component calls proc_MSS_Crawl with @CrawlStage equals 107 against all crawl stores to move documents from Crawl History Set to Crawl Queue

7. If documents were copied in previous statement master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 108 to notify that all crawl components from specified crawl store MUST join the crawl

8. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 103, notifies that start addresses were successfully added and now waits for crawl components

9. Crawl components call proc_MSS_GetCrawls to check if there any crawls they need to join

10. Crawl components call proc_MSS_CrawlAdmin with @CrawlState equals 16 to notify that they are can start crawling

11. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 104 to check if all crawl components done with starting the crawl

12. Master component call proc_MSS_FlushTemp0 to move data from Links Table to Crawl Queue

13. Crawl components call proc_MSS_GetNextCrawlBatch to assign some not yet crawled documents from Crawl Queue as being crawled by this crawl component and return those documents

14. Crawl components insert newly discovered links into Link Set

15. Crawl components call proc_MSS_ProcessCommitted for every link to mark it as processed

16. Master component calls proc_MSS_FlushTemp0 to move data from Links Table to Crawl Queue and GOTO 11

17. Crawl components call proc_MSS_Crawl with @CrawlStage equals 141 to check if it is done with the crawl

18. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 142 to inform that it is not doing anything (if it is not doing anything)

19. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 143 to inform that it is not doing anything (if it is doing anything)

20. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 140 to check if crawl components have finished with first part and if yes move crawl to Unvisited to Queue stage

21. Master component calls proc_MSS_Crawl with @CrawlStage equals 145 to start next crawl stage

22. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 144 to start Delete Unvisited crawl stage

23. Crawl components perform Documents Processing Sequence

24. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 140 to check if crawl components have finished with that part of the crawl and if yes move crawl to Wait for All Databases Stage

25. Master component calls proc_MSS_Crawl with @CrawlStage equals 151 against all crawl stored to check if the all crawl stored have finished with the crawl

26. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 150 to start crawl completion; sets sub status to Wait For Crawl Components

27. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 147 to inform that they have finished with the crawl

28. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 146 to check if all crawl components done with the crawl and if yes update Crawl History Set; sets sub status to Substatus Completing

29. Master component calls proc_MSS_Crawl with @CrawlStage equals 149 against all crawl stores to update crawl statistics and prepare datasets for anchor text crawl

30. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 152; sets sub status to Delete Pending

31. Master component calls proc_MSS_Crawl with @CrawlStage equals 153

32. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 148 to report that crawl is done is to start anchor text crawl

3.2.5.4 Anchor Text Crawl

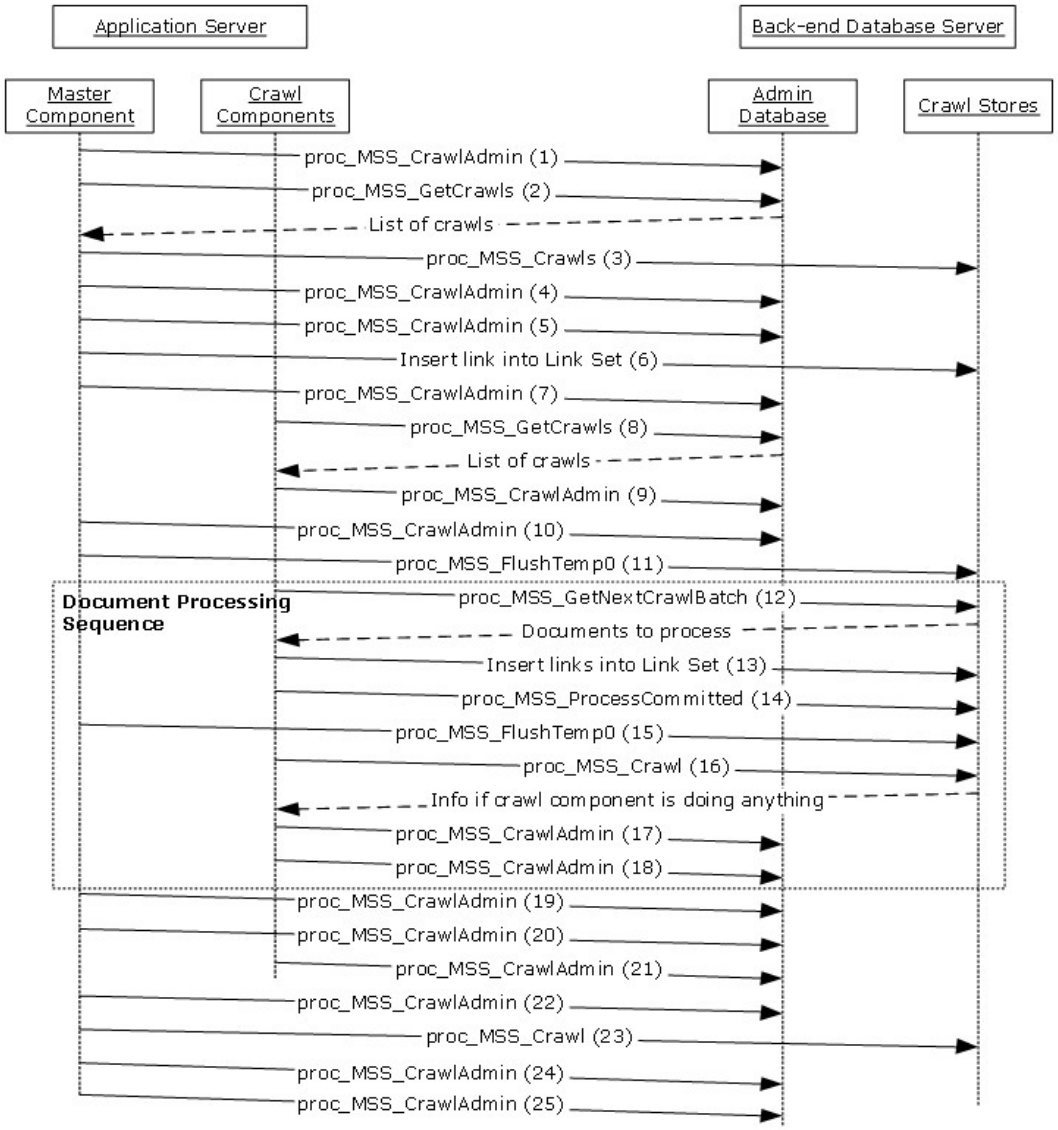


Figure 8: Anchor text crawl diagram

1. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 148 to either start a new anchor text crawl after successful portal content crawl or restart failed anchor text crawl

2. Master component calls proc_MSS_GetCrawls with @CatalogId equal "Anchor Project" to get list of crawls

3. Master component calls proc_MSS_Crawl with @CrawlStage equals 170 against all crawl stores

4. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 102 to set content source identifier for newly created crawl

5. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 108, to inform that all crawl components from the crawl store which process the host MUST join the crawl

6. Master component insert link into Links Set of all crawl stores

7. Master component calls proc_MSS_CrawlAdmin with @CrawlState equals 103, notifies that start addresses were successfully added and now waits for crawl components

8. Crawl components call proc_MSS_GetCrawls to check if there any crawl they need to join

9. Crawl components call proc_MSS_CrawlAdmin with @CrawlState equals 16

10. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 104 to check if all crawl components done with starting the crawl

11. Crawl Components call proc_MSS_FlushTemp0 to move data from MssTranTempTable0 to MSSCrawlQueue

12. Crawl components call proc_MSS_GetNextCrawlBatch to assign some not yet crawled documents from Crawl Queue as being crawled by this crawl component and return those documents

13. Crawl components insert newly discovered link into Link Set

14. Crawl components call proc_MSS_ProcessCommitted for every link to mark it as processed

15. Master component calls proc_MSS_FlushTemp0 to move data from Links Set to Crawl Queue

16. Crawl components call proc_MSS_Crawl with @CrawlStage equals 141 to check if it is done with the crawl

17. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 142 to inform that it is not doing anything

18. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 143 to inform that it is not doing anything

19. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 140 to check if crawl components have finished with current stage of crawl and if yes move crawl to Wait for all databases

20. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 150 to start crawl completion; sets sub status to Wait For Crawl Components

21. Crawl components call proc_MSS_CrawlAdmin with @CrawlStage equals 147 to inform that they have finished with the crawl

22. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 146 to check if all crawl components done with the crawl and if yes update Crawl History Set; sets sub status to Completing

23. Master component calls proc_MSS_Crawl with @CrawlStage equals 149 against all crawl stores

24. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 152; sets sub status to Delete Pending

25. Master component calls proc_MSS_CrawlAdmin with @CrawlStage equals 148 to report that crawl is done

3.2.5.5 Status Changes

This section contains the status change sequence used to reflect changes to search component status. The flow is displayed in the following diagram:

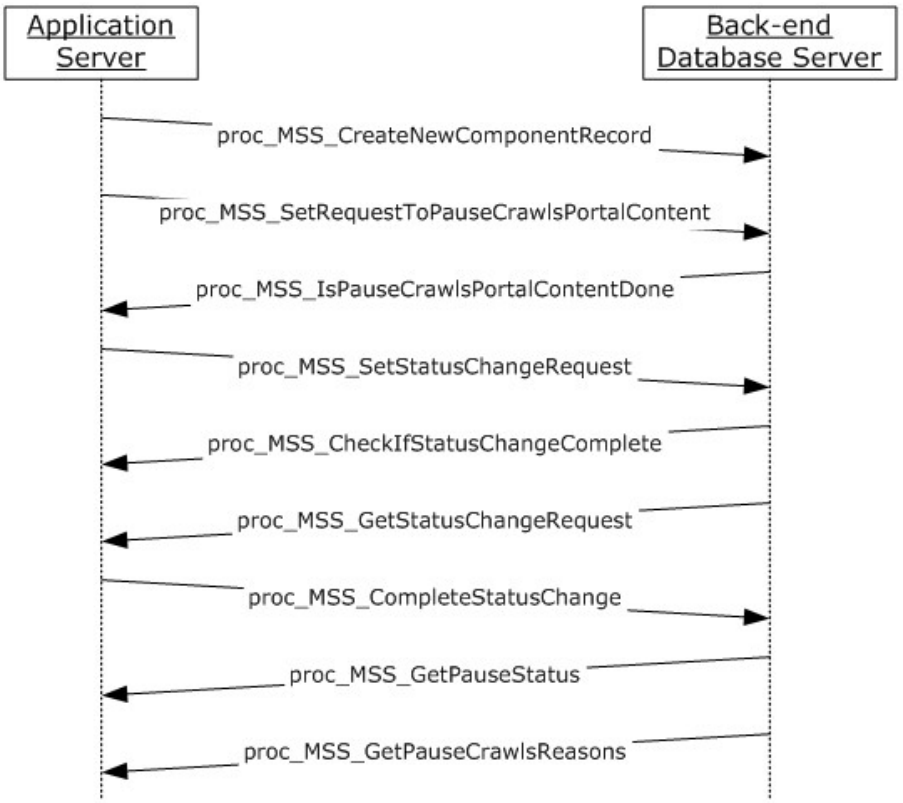


Figure 9: Data Flow for Status Changes

The following text explains the individual steps in the diagram:

When a new search component is initialized, it creates a component record in the back-end database server (proc_MSS_CreateNewComponentRecord). Subsequent to this, the protocol client (application server) can make calls to the back-end database server to register status change requests for search components (proc_MSS_SetStatusChangeRequest). The status change may be a reset or pause request. A pause may be requested for crawls or master merges, with reasons specified in Section 2.2.2.1. During database repartitioning as specified in [MS-SRCHTP] Section 3.1.1.4 , the application server makes a call requesting crawls to be paused for the portal content project of all components (proc_MSS_SetRequestToPauseCrawlsPortalContent), before requesting a pause status change for all components (proc_MSS_SetStatusChangeRequest). After each status change request, the application server that has requested the change waits for the change to take effect. This is done by calling the back-end database server repeatedly to check if the status change has completed (proc_MSS_IsPauseCrawlsPortalContentDone, proc_MSS_CheckIfStatusChangeComplete).

The applications servers on which search components are hosted call the back-end database server every (10) seconds to check if there is a status change request for the relevant components (proc_MSS_GetStatusChangeRequest). The application server processes the status change, and

when the change is complete, makes a call to the back-end database server to register that the status change is complete (proc_MSS_CompleteStatusChange).

Application servers on which search components are hosted call the back-end database server every (10) seconds to check if crawls or master merges are paused (proc_MSS_GetPauseStatus). When needed, the front-end web server makes a call to determine whether crawls are paused, and the reason that they are paused (proc_MSS_GetPauseCrawlsReasons).

3.2.5.6 Configuration Properties

This section contains the configuration property sequence used to modify configuration properties. The flow is displayed in the following diagram:

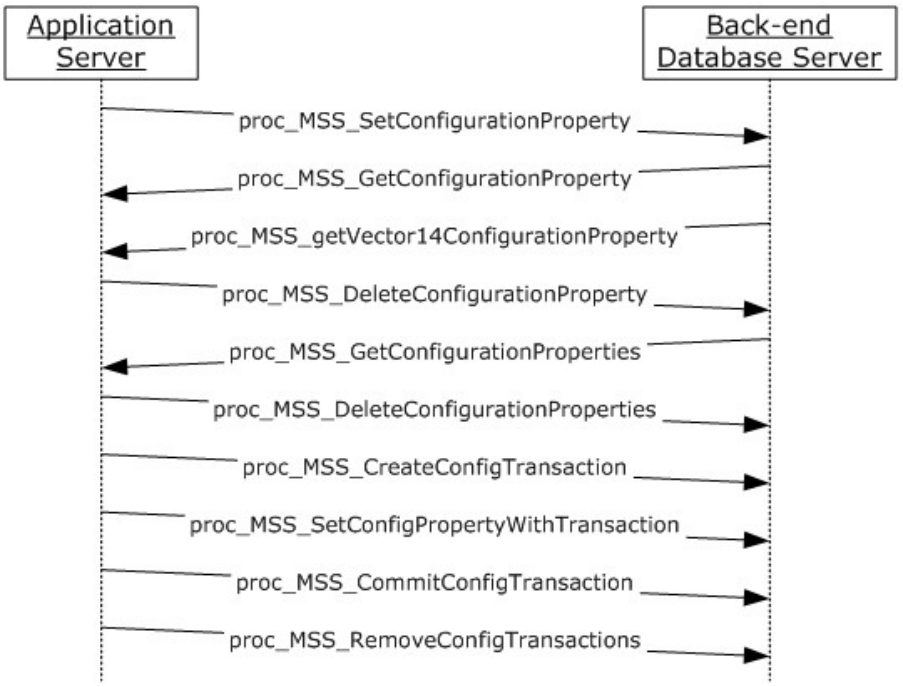


Figure 10: Configuration property sequence diagram

The following text defines the steps of the sequence:

For the specification of configuration properties see Section 3.1.1.36. When an application server needs to set a configuration property, it makes a call to the back-end database server (proc_MSS_SetConfigurationProperty). The application server makes subsequent calls to retrieve (proc_MSS_GetConfigurationProperty) or delete (proc_MSS_DeleteConfigurationProperty) a configuration property. The application server can also retrieve (proc_MSS_GetConfigurationProperties) or delete (proc_MSS_DeleteConfigurationProperties) multiple configuration properties using wildcard rules.

Many configuration properties can be set together as a single operation using configuration property transactions. For the specification of configuration property transactions see Section 3.1.1.36. The application server first makes a call to the back-end database server to create a new transaction (proc_MSS_CreateConfigTransaction). The application server makes calls to set configuration properties within this transaction (proc_MSS_SetConfigPropertyWithTransaction). The configuration properties will be uncommitted. On completion of setting properties for the transaction, the

application server makes a call to the back-end database server to commit the transaction (proc_MSS_CommitConfigTransaction), which commits all the configuration properties in the transaction. The application server can also remove all uncommitted configuration transactions by making a call to the back-end database server (proc_MSS_RemoveConfigTransactions).

3.2.5.7 Scope Compilation

This section contains the sequence used for search scope compilation. The flow is displayed in the following diagram:

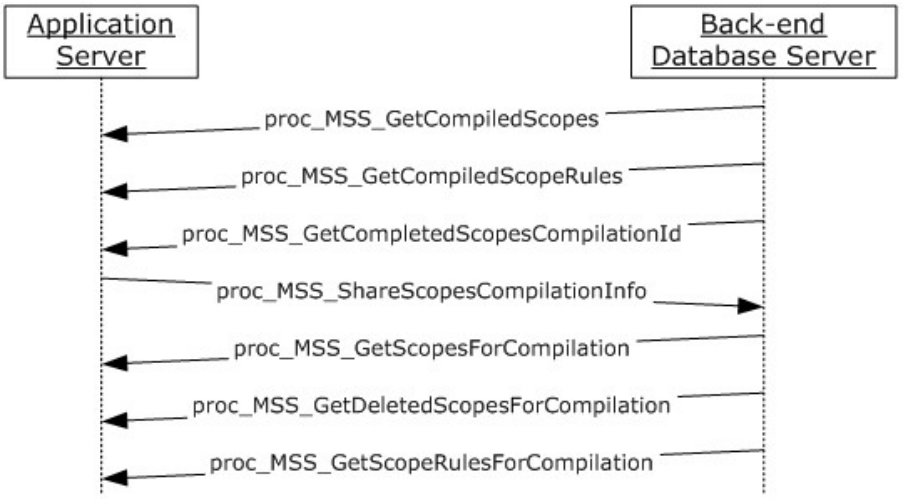


Figure 11: Scope compilation diagram

The following text describes the sequence:

When the search catalog is reset or Microsoft® SharePoint® 2010 Products and Technologies service is started, the application server makes a call to the back-end database server to retrieve information about **compiled search scopes** (proc_MSS_GetCompiledScopes) and compiled search scope rules (proc_MSS_GetCompiledScopeRules). When the search catalog is reset, or a crawl component is reset, the application server makes a call to the back-end database server to get the latest search scope compilation identifier (proc_MSS_GetCompletedScopesCompilationId).

The application server makes a call (proc_MSS_ShareScopesCompilationInfo) to the back-end database server every 30 seconds to start or update search scope compilation if necessary. It also calls this stored procedure to inform the back-end database server when it has completed a search scope compilation. When a search scope compilation is started, the application server makes a call to the back-end database server to retrieve the search scopes to be compiled (proc_MSS_GetScopesForCompilation) the deleted search scopes to be compiled (proc_MSS_GetDeletedScopesForCompilation) and the search scope rules to be compiled (proc_MSS_GetScopeRulesForCompilation) in the current search scope compilation.

3.2.5.8 Request Master Merge

For each full-text index catalog "Portal_Content" and "AnchorProject", the following actions MUST be taken by the admin server on a recurring basis. The recurrences SHOULD be one minute apart, but all finite time intervals are allowed by the protocol.

The application server calls the **proc_MSS_GetIncompleteCommands** stored procedure (section 3.1.5.39) with *@ObjectName* set to the value by

```
<object name>=<search application guid>-<role>-<component number>/<project
name>/indexer
```

where

- `<search application guid>` is the unique identifier of the **search application**,
- `<role>` is "query" if this application server is a query component (2), and "crawl" if it is a crawl component,
- `<component number>` is the QueryComponentNumber value ([MS-SRCHTP] section 3.1.1.2) of the query component (2), or the CrawlComponentNumber value ([MS-SRCHTP] section 3.1.1.3) of the crawl component, and
- `<project name>` is "Portal_Content" or "AnchorProject" as appropriate to the full-text index catalog.

For each command in the received commands result set (section 3.1.5.39.1) whose Description value is "request master merge", a master merge process ([MS-CIFO] section 2.9) MUST be performed, beginning within the next minute.

After ensuring that the master merge process ([MS-CIFO] section 2.9) will be performed, the application server calls the **proc_MSS_ReportCommandCompleted** stored procedure (section 3.1.5.73) with the *@CommandID* parameter (section 3.1.5.73) set to the commandID value found in the command result set (section 3.1.5.39.1).

3.2.6 Timer Events

3.2.7 Other Local Events

3.3 Server Details for Windows SharePoint Server

3.3.1 Abstract Data Model

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

The following structures described in Microsoft® SharePoint® 2010 Products and Technologies are used in Microsoft® SharePoint® Foundation 2010:

- Crawl Status Set (section 3.1.1.1)
- Crawl URL History Set (section 3.1.1.2)
- Deleted URL Set (section 3.1.1.3)
- Crawl Queue (section 3.1.1.4)
- Links Set (section 3.1.1.5)
- Crawled Hosts Set (section 3.1.1.7)

- Anchor Text Info Set (section 3.1.1.8)
- Annotations Set (section 3.1.1.37)
- Pending Annotations Set (section 3.1.1.9)
- Anchor Change Set (section 3.1.1.14)
- Pending Anchor Change Set (section 3.1.1.15)
- Reported Crawl Error Set (section 3.1.1.16)
- Current DocID Chunks Set (section 3.1.1.17)
- Requested Delete Crawls Set (section 3.1.1.19)
- Changed Anchor Target Documents (section 3.1.1.20)
- Changed Anchor Source Documents (section 3.1.1.21)
- Changed Anchor Committed Documents (section 3.1.1.22)
- Changed Anchor Deleted Documents (section 3.1.1.23)
- Crawl Component Status (section 3.1.1.25)
- Local Crawl Components (section 3.1.1.26)
- Local Completed Crawls (section 3.1.1.27)
- Crawl Store Status (section 3.1.1.28)
- Scopes to Compile (section 3.1.1.29)
- Scope Rules to Compile (section 3.1.1.30)
- Deleted Scopes (section 3.1.1.31)
- Deleted Scopes to Compile (section 3.1.1.32)
- Compiled Scopes (section 3.1.1.33)
- Compiles Scope Rules (section 3.1.1.34)
- Component Activity (section 3.1.1.35)
- Configuration Properties (section 3.1.1.36)

3.3.2 Timers

None.

3.3.3 Initialization

None.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

The following stored procedures are being called from the protocol server:

proc_MSS_AddAndReturnCrawledProperty (section 3.1.5.1)

proc_MSS_AddCrawledPropertyCategoryWithDefaults (section 3.1.5.2)

proc_MSS_CheckIfStatusChangeComplete (section 3.1.5.3)

proc_MSS_CommitConfigTransaction (section 3.1.5.5)

proc_MSS_CompleteStatusChange (section 3.1.5.6)

proc_MSS_Crawl (section 3.1.5.7)

proc_MSS_CrawlAdmin (section 3.1.5.8)

proc_MSS_CreateNewComponentRecord (section 3.1.5.10)

proc_MSS_CreateConfigTransaction (section 3.1.5.11)

proc_MSS_DeleteConfigurationProperties (section 3.1.5.12)

proc_MSS_DeleteConfigurationProperty (section 3.1.5.13)

proc_MSS_FlushTemp0 (section 3.1.5.14)

proc_MSS_GetCompiledScopeRules (section 3.1.5.16)

proc_MSS_GetCompiledScopes (section 3.1.5.17)

proc_MSS_GetCompletedScopesCompilationID (section 3.1.5.18)

proc_MSS_GetConfigurationProperties (section 3.1.5.20)

proc_MSS_GetConfigurationProperty (section 3.1.5.21)

proc_MSS_GetContentSourceDocCount (section 3.1.5.22)

proc_MSS_GetCrawledProperties (section 3.1.5.23)

proc_MSS_GetCrawls (section 3.1.5.25)

proc_MSS_GetCurrentRegistryVersion (section 3.1.5.26)

proc_MSS_GetDeleteCrawl (section 3.1.5.32)

proc_MSS_GetDeletedScopesForCompilation (section 3.1.5.33)

proc_MSS_GetDocCount (section 3.1.5.34)

proc_MSS_GetDocStatus (section 3.1.5.35)

proc_MSS_GetError (section 3.1.5.36)

proc_MSS_GetHost (section 3.1.5.38)

proc_MSS_GetManagedProperties (section 3.1.5.40)

proc_MSS_GetNextCrawlBatch (section 3.1.5.45)

proc_MSS_GetNextDocIDChunk (section 3.1.5.47)

proc_MSS_GetPauseCrawlsReasons (section 3.1.5.48)

proc_MSS_GetPauseStatus (section 3.1.5.49)

proc_MSS_GetPreferredNameList (section 3.1.5.50)

proc_MSS_GetRankingModels (section 3.1.5.51)

proc_MSS_GetSampleExtremes (section 3.1.5.52)

proc_MSS_GetSchemaHighLevelInfo (section 3.1.5.53)

proc_MSS_GetSchemaMappings (section 3.1.5.54)

proc_MSS_GetSchemaParameters (section 3.1.5.55)

proc_MSS_GetScopeRulesForCompilation (section 3.1.5.56)

proc_MSS_GetScopesForCompilation (section 3.1.5.57)

proc_MSS_GetSDID (section 3.1.5.58)

proc_MSS_GetStaticRankingFeatures (section 3.1.5.59)

proc_MSS_GetStatus (section 3.1.5.60)

proc_MSS_GetStatusChangeRequest (section 3.1.5.61)

proc_MSS_ProcessCommitted (section 3.1.5.66)

proc_MSS_PushSD (section 3.1.5.67)

proc_MSS_Recompile (section 3.1.5.69)

proc_MSS_RemoveConfigTransactions (section 3.1.5.71)

proc_MSS_ResetCatalog (section 3.1.5.74)

proc_MSS_SetConfigPropertyWithTransaction (section 3.1.5.75)

proc_MSS_SetConfigurationProperty (section 3.1.5.76)

proc_MSS_SetCrawledPropertyIsSampleCacheFull (section 3.1.5.78)

proc_MSS_SetStatusChangeRequest (section 3.1.5.80)

proc_MSS_ShareScopesCompilationInfo (section 3.1.5.81)

proc_MSS_TruncateCleanupTable (section 3.1.5.82)

proc_MSS_UnloadTransactions (section 3.1.5.83)

proc_MSS_UpdateCrawlStatistics (section 3.1.5.87)

proc_MSS_UpdateDocCount (section 3.1.5.88)

proc_MSS_UpdatePropertyStore (section 3.1.5.90)

3.3.6 Timer Events

None.

3.3.7 Other Local Events

None.

3.4 Windows SharePoint Server Client Details

3.4.1 Abstract Data Model

None.

3.4.2 Timers

None.

3.4.3 Initialization

None.

3.4.4 Higher-Layer Triggered Events

None.

3.4.5 Message Processing Events and Sequencing Rules

This section describes sequences performed on the client as part of this protocol. These include the following:

- Full crawl
- Incremental crawl
- Delete crawl
- Status changes
- Configuration property modification

3.4.5.1 Full Crawl Sequence

Full crawl data flow is identical to one described in section 3.2.5.1 with the following configuration detail:

The configuration of search on the protocol client is as follows:

- There is only one component (that is, all messages come from a single component)
- There is only one database (that is, all structures stored in one database)

3.4.5.2 Incremental Crawl Sequence

Incremental crawl data flow is identical to one described in section 3.2.5.2 with the following configuration detail:

The configuration of search on the protocol client is as follows:

- There is only one component (that is, all messages come from a single component)
- There is only one database (that is, all structures stored in one database)

3.4.5.3 Delete Crawl Sequence

Delete crawl data flow is identical to one described in section 3.2.5.3 with the following configuration detail:

The configuration of search on the protocol client is as follows:

- There is only one component (that is, all messages come from a single component)
- There is only one database (that is, all structures stored in one database)

3.4.5.4 Configuration Properties

Configuration properties data flow is identical to one described in section 3.2.5.6.

3.4.5.5 Scope Compilation

Scope compilation data flow is identical to one described in section 3.2.5.7.

3.4.6 Timer Events

None.

3.4.7 Other Local Events

None.

4 Protocol Examples

This section describes three possible interactions between a client and server through the use of this protocol.

4.1 Add a New Content Source

This example describes the configuration changes of adding a new content source. A search service application named b9174869-840b-4cdc-b655-49a3a56905cb is demonstrated with 1 crawl component:

- Crawl-0: a crawl component

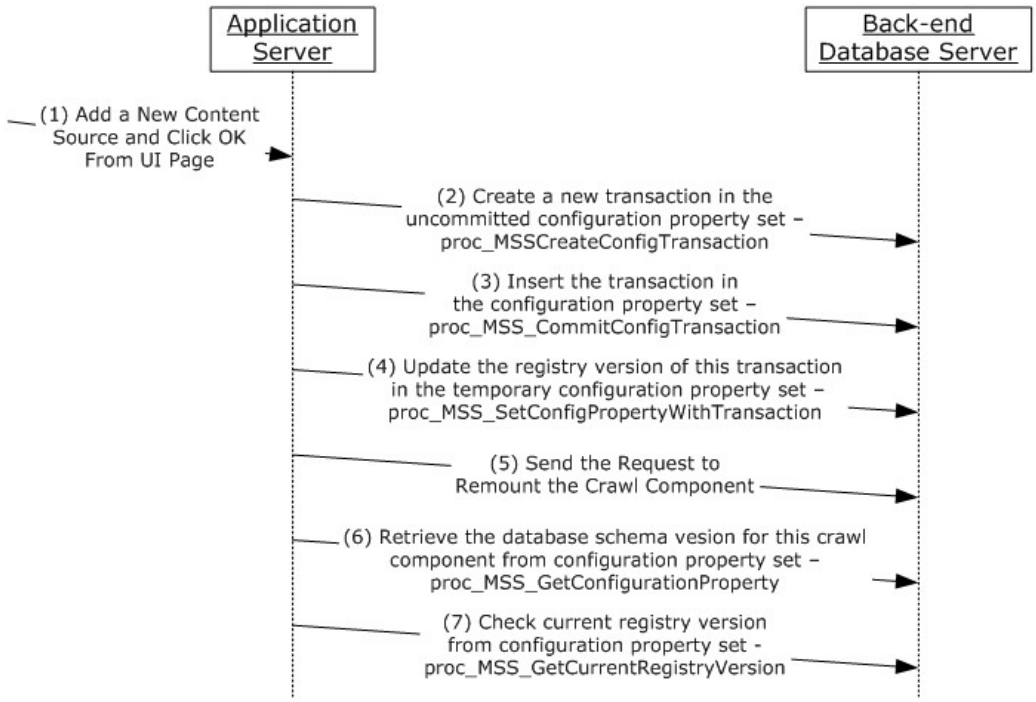


Figure 12: Adding a new content source

The steps are explained in the following:

1. The application server receives a request from the user interface to add a content source.

2. The application server executes **proc_MSS_CreateConfigTransaction** on the back-end database server to create a new configuration transaction and insert it into the uncommitted configuration property set. This stored procedure returns the transaction identifier as the return code, that is, 2 in this example.

3. The application server executes **proc_MSS_CommitConfigTransaction** on the back-end database server to delete the transaction created in Step 3 from the uncommitted configuration set and insert it into the committed set with the following parameter.

   - @TransactionID is set to 2; the value returned in Step 2.

4. The application server executes **proc_MSS_SetConfigPropertyWithTransaction** on the back-end database server to update the registry version of the transaction committed in Step 3 with the following parameters.

   - @Name is set to "CurrentRegistryVersion"
   - @Value is set to 0x000002e9 (745 as decimal)
   - @TransactionID is set to 2

5. The application server sends a request to remount Crawl-0.

6. The application server executes **proc_MSS_GetConfigurationProperty** on the back-end database server to check the database schema version of Crawl-0with the following parameters.

   - @Name is set to '67c2cdf0-9347-427f-927c-0850fa1ef450-crawl-0'
   - @Value OUTPUT will be set to 14.0.4006.3007 by this stored procedure

7. The application server executes **proc_MSS_GetCurrentRegistryVersion** on the back-end database server to get the current registry version for Crawl-0.

   - @LikeKey is set to '67c2cdf0-9347-427f-927c-0850fa1ef450-crawl-0'
   - @CurrentRegistryVersion OUTPUT will be set to 745 by this stored procedure

4.2 Add a New Host Distribution Rule

This example describes the status changes related to adding a new **host distribution rule**.

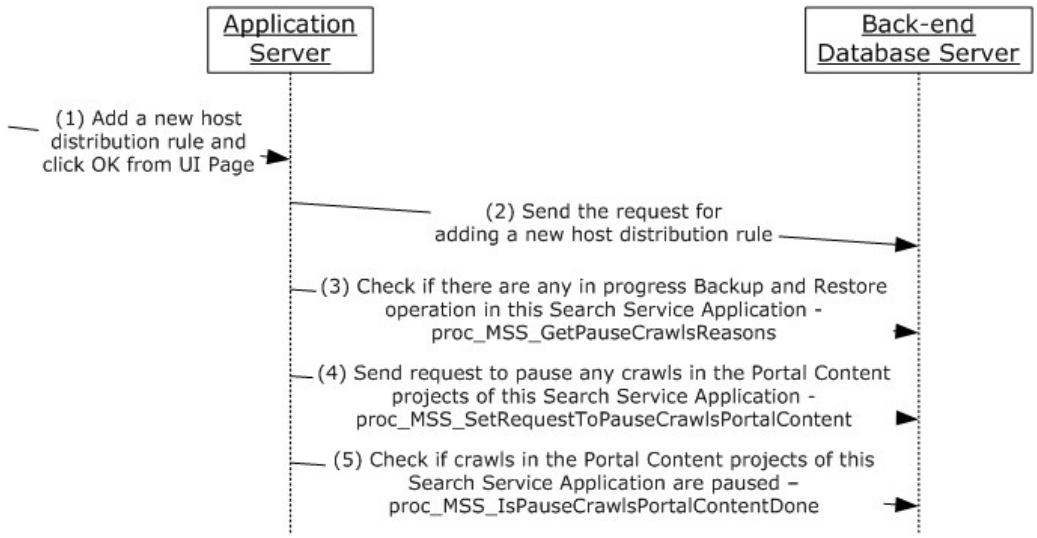


Figure 13: Adding a new host distribution rule

The steps are explained in the following paragraph.

Adding a new host distribution rule is carried out when a user clicks OK to create the new host distribution rule from the UI page.

1. The application server receives a request from the user interface to apply a new host distribution rule.

2. The application server forwards this request to the search service.

3. The application server calls **proc_MSS_GetPauseCrawlsReasons** on the back-end database server to check if there are any in progress Backup and Restore operations. If there are, adding a new rule will be canceled.

4. The application server calls **proc_MSS_SetRequestToPauseCrawlsPortalContent** on the back-end database server to request pausing all crawls on the Portal Content projects of this Search Service Application. It's executed with the following parameter:

   - @Reason is set to be 0x00000020 (SAPR_PAUSE_FOR_REFACTORING)

5. The application server calls **proc_MSS_IsPauseCrawlsPortalContentDone** on the back-end database server to verify if all crawls on the Portal Content projects of this Search Service Application are paused.

5 Security

5.1 Security Considerations for Implementers

Security for this protocol is controlled by the user rights to the databases on the back-end database server, which is negotiated as part of the TDS protocol ([MS-TDS]).

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

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

- Microsoft® FAST™ Search Server 2010
- Microsoft® SharePoint® Foundation 2010
- Microsoft® SQL Server® 2005
- Microsoft® SQL Server® 2008
- Microsoft® SQL Server® 2008 R2

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 3.1.5.2: 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.

<2> Section 3.1.5.5: [1] If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<3> Section 3.1.5.6: [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.

<4> Section 3.1.5.7: [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.

<5> Section 3.1.5.10: 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.

<6> Section 3.1.5.11: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<7> Section 3.1.5.12: [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.

<8> Section 3.1.5.13: [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.

<9> Section 3.1.5.21: [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.

<10> Section 3.1.5.22: [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.

<11> Section 3.1.5.26: 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.

<12> Section 3.1.5.27: 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.

<13> Section 3.1.5.34: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<14> Section 3.1.5.36: [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 3.1.5.43: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<16> Section 3.1.5.46: [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.

<17> Section 3.1.5.47: [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.

<18> Section 3.1.5.48: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<19> Section 3.1.5.49: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<20> Section 3.1.5.60: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<21> Section 3.1.5.63: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<22> Section 3.1.5.66: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<23> Section 3.1.5.67: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<24> Section 3.1.5.69: If a given stored procedure performs an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<25> Section 3.1.5.70: If a given stored procedure performs an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<26> Section 3.1.5.71: [1] If a given stored procedure performs an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<27> Section 3.1.5.72: [1] If a given stored procedure performs an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<28> Section 3.1.5.75: [1] If a given stored procedure performs an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<29> Section 3.1.5.76: [1] If a given stored procedure performs an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<30> Section 3.1.5.77: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<31> Section 3.1.5.78: 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 3.1.5.79: 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 3.1.5.80: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<34> Section 3.1.5.82: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<35> Section 3.1.5.89: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

<36> Section 3.1.5.91: If a given stored procedure does an INSERT, UPDATE, or DELETE SQL operation in the database, the stored procedure returns one or more extra result sets that contain the number of records affected by the operation.

7 Change Tracking

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

8 Index

A

Abstract data model
  Anchor Change – server 56
  Anchor Document Properties Blob – server 62
  Anchor Text Info – server 53
  Annotations – server 62
  Changed Anchor Committed Documents – server 57
  Changed Anchor Deleted Documents – server 58
  Changed Anchor Source Documents – server 57
  Changed Anchor Target Documents – server 57
  client (section 3.2.1 179, section 3.4.1 198)
  Colleague Links – server 53
  Commands – server 63
  Compiled Scope Rules – server 60
  Compiled Scopes – server 60
  Component Activity – server 61
  Configuration Properties – server 61
  Crawl Component Status – server 58
  crawl queue – server 50
  Crawl Status – server 47
  Crawl Store Status – server 59
  crawl URL history – server 47
  crawled Hosts – server 53
  Current DocID Chunk – server 57
  Current DocID Chunks – server 56
  Definitions – server 62
  Deleted Scopes – server 60
  Deleted Scopes to Compile – server 60
  Deleted URL – server 50
  links – server 51
  Local Completed Crawls – server 59
  Local Crawl Components – server 58
  Metadata store crawl components – server 63
  Pending Anchor Change – server 56
  Pending Annotations – server 54
  Pending Annotations Status – server 55
  reported Crawl Error – server 56
  Requested Delete Crawls – server 57
  Scope Rules to Compile – server 59
  Scopes to Compile – server 59
  Search Component Name – server 62
  server (section 3.1.1 47, section 3.3.1 194)
  Social Distance Deleted Property – server 55
  Social Distance New Property – server (section 3.1.1.13 56, section 3.1.1.43 63)
  Social Distance Property – server 55
  UserID DocID Pair – server 58
Add a new content source example 200
Add a new host distribution rule 201
Alert History Update Record binary structure 35
Alert History Update Structure binary structure 36
Anchor Change 56
Anchor Change Type simple type 22
Anchor Document Properties Blob 62
Anchor text crawl sequence 189
Anchor Text Info 53
Annotations 62
Applicability 15
Attribute groups - overview 46
Attributes - overview 46

B

Binary structures
  Alert History Update Record 35
  Alert History Update Structure 36
  Definitions Update Record 36
  Definitions Update Structure 37
  DocProps Add Record 25
  DocProps Add Structure 26
  DocProps Delete Record 27
  DocProps Delete Structure 27
  DocProps Delete With Hash Record 28
  DocProps Delete With Hash Structure 28
  DocResults Anchor Update Record 33
  DocResults Anchor Update Structure 33
  DocResults Update Record 29
  DocResults Update Structure 32
  DocSdIds Update Record 34
  DocSdIds Update Structure 34
  Transaction Commit Record 37
  Transaction Data Blob 44
  Transaction string 45
  Transactions Commit Structure 44
  Update Anchor Add Record 24
  Update Anchor Add Structure 25
Binary structures - overview 24

C

Capability negotiation 15
Change tracking 207
Changed Anchor Committed Documents 57
Changed Anchor Deleted Documents 58
Changed Anchor Source Documents 57
Changed Anchor Target Documents 57
Client
  abstract data model (section 3.2.1 179, section 3.4.1 198)
  anchor text crawl sequence 189
  configuration properties sequence 199
  configuration property sequence 192
  delete crawl sequence (section 3.2.5.3 185, section 3.4.5.3 199)
  full crawl sequence (section 3.2.5.1 179, section 3.4.5.1 198)
  higher-layer triggered events (section 3.2.4 179, section 3.4.4 198)
  incremental crawl sequence (section 3.2.5.2 182, section 3.4.5.2 199)
  initialization (section 3.2.3 179, section 3.4.3 198)
  local events 199
  message processing (section 3.2.5 179, section 3.4.5 198)
  Office SharePoint Server interface 179

overview 179
request master merge sequence 193
scope compilation sequence (section 3.2.5.7 193, section 3.4.5.5 199)
sequencing rules (section 3.2.5 179, section 3.4.5 198)
status change sequence 191
timer events 199
timers (section 3.2.2 179, section 3.4.2 198)
Colleague Links 53
Commands set 63
Common data types
overview 16
Compiled Scope Rules 60
Compiled Scopes 60
Complex types - overview 46
Component Activity 61
Configuration properties 61
overview 13
Configuration properties sequence 199
Configuration property sequence 192
Content Source Status simple type 20
Crawl Completing Sub Status simple type 17
Crawl Component State simple type 19
Crawl Component Status 58
Crawl Crawling Sub Status simple type 17
Crawl Error Codes simple type 18
Crawl Error Levels simple type 18
Crawl Error Logging Levels simple type 18
Crawl queue 50
Crawl Request Type simple type 16
Crawl Starting Sub Status simple type 17
Crawl Status 47
Crawl Status simple type 17
Crawl Stopping Sub Status simple type 18
Crawl Store Status 59
Crawl Stores with Links flag structure 23
Crawl Type simple type 16
Crawl URL history 47
Crawled Hosts 53
Current DocID Chunk 57
Current DocID Chunks 56

D

Data model - abstract
client (section 3.2.1 179, section 3.4.1 198)
server (section 3.1.1 47, section 3.3.1 194)
Data types
Anchor Change Type simple type 22
common 16
Content Source Status simple type 20
Crawl Completing Sub Status simple type 17
Crawl Component State simple type 19
Crawl Crawling Sub Status simple type 17
Crawl Error Codes simple type 18
Crawl Error Levels simple type 18
Crawl Error Logging Levels simple type 18
Crawl Request Type simple type 16
Crawl Starting Sub Status simple type 17
Crawl Status simple type 17
Crawl Stopping Sub Status simple type 18
Crawl Type simple type 16
Index Type simple type 21
Item Delete Reasons simple type 22
Item Type simple type 21
Managed Type simple type 19
Project Identifier simple type 16
Project Name simple type 21
Search Catalog Status simple type 20
Transaction Scope simple type 20
Transaction Type simple type 19
Update Queue Type simple type 21
Data types - simple
Anchor Change Type 22
Content Source Status 20
Crawl Completing Sub Status 17
Crawl Component State 19
Crawl Crawling Sub Status 17
Crawl Error Codes 18
Crawl Error Levels 18
Crawl Error Logging Levels 18
Crawl Request Type 16
Crawl Starting Sub Status 17
Crawl Status 17
Crawl Stopping Sub Status 18
Crawl Type 16
Index Type 21
Item Delete Reasons 22
Item Type 21
Managed Type 19
Project Identifier 16
Project Name 21
Search Catalog Status 20
Transaction Scope 20
Transaction Type 19
Update Queue Type 21
Definitions 62
Definitions Update Record binary structure 36
Definitions Update Structure binary structure 37
Delete crawl sequence (section 3.2.5.3 185, section 3.4.5.3 199)
Deleted Scopes 60
Deleted Scopes to Compile 60
Deleted URL 50
DocProps Add Record binary structure 25
DocProps Add Structure binary structure 26
DocProps Delete Record binary structure 27
DocProps Delete Structure binary structure 27
DocProps Delete With Hash Record binary structure 28
DocProps Delete With Hash Structure binary structure 28
DocResults Anchor Update Record binary structure 33
DocResults Anchor Update Structure binary structure 33
DocResults Update Record binary structure 29
DocResults Update Structure binary structure 32
DocSdIds Update Record binary structure 34
DocSdIds Update Structure binary structure 34

E

Elements - overview 46
End Path Flags flag structure 24
Events
  local - client 199
  local - server (section 3.1.7 178, section 3.3.7 198)
  timer - client 199
  timer - server (section 3.1.6 178, section 3.3.6 198)
Examples
  add a new content source 200
  add a new host distribution rule 201
  overview 200

F

Fields - vendor-extensible 15
Flag structures
  Crawl Stores with Links 23
  End Path Flags 24
  Pause Crawls Reasons 22
  Transaction Flags 23
Full crawl sequence (section 3.2.5.1 179, section 3.4.5.1 198)

G

Gatherer DB IDs Result Set method 122
Get Mappings for Crawled Properties by Identifier Result Set method 132
Glossary 9
Groups - overview 46

H

Higher-layer triggered events
  client (section 3.2.4 179, section 3.4.4 198)
  server (section 3.1.4 63, section 3.3.4 195)

I

Implementer - security considerations 203
Incremental crawl sequence (section 3.2.5.2 182, section 3.4.5.2 199)
Index of security parameters 203
Index Type simple type 21
Informative references 11
Initialization
  client (section 3.2.3 179, section 3.4.3 198)
  server (section 3.1.3 63, section 3.1.3 63, section 3.3.3 195)
Interfaces - client
  Office SharePoint Server 179
Interfaces - server
  Office SharePoint Server 47
Introduction 9
Item Delete Reasons simple type 22
Item Type simple type 21

L

Links 51
Local Completed Crawls 59
Local Crawl Components 58
Local events
  client 199
  server (section 3.1.7 178, section 3.3.7 198)

M

Managed Type simple type 19
Message processing
  client (section 3.2.5 179, section 3.4.5 198)
  server 196
Messages
  Alert History Update Record binary structure 35
  Alert History Update Structure binary structure 36
  attribute groups 46
  attributes 46
  binary structures 24
  common data types 16
  complex types 46
  Crawl Stores with Links flag structure 23
  Definitions Update Record binary structure 36
  Definitions Update Structure binary structure 37
  DocProps Add Record binary structure 25
  DocProps Add Structure binary structure 26
  DocProps Delete Record binary structure 27
  DocProps Delete Structure binary structure 27
  DocProps Delete With Hash Record binary structure 28
  DocProps Delete With Hash Structure binary structure 28
  DocResults Anchor Update Record binary structure 33
  DocResults Anchor Update Structure binary structure 33
  DocResults Update Record binary structure 29
  DocResults Update Structure binary structure 32
  DocSdIds Update Record binary structure 34
  DocSdIds Update Structure binary structure 34
  elements 46
  End Path Flags flag structure 24
  groups 46
  namespaces 46
  Pause Crawls Reasons flag structure 22
  Scope Compilation result set 45
  simple types 46
  table structures 45
  Transaction Commit Record binary structure 37
  Transaction Data Blob binary structure 44
  Transaction Flags flag structure 23
  Transaction string binary structure 45
  Transactions Commit Structure binary structure 44
  transport 16
  Update Anchor Add Record binary structure 24
  Update Anchor Add Structure binary structure 25
  view structures 45
  XML structures 45
Metadata store crawl components 63
Methods
  Gatherer DB IDs Result Set 122

Get Mappings for Crawled Properties by Identifier Result Set 132
proc_MSS_AddAndReturnCrawledProperty 64
proc_MSS_AddCrawledPropertyCategoryWithDefaults 65
proc_MSS_CheckIfStatusChangeComplete 66
proc_MSS_CleanupWithInterval 178
proc_MSS_CommitConfigTransaction 68
proc_MSS_CommitTransactions 148
proc_MSS_CompleteStatusChange 68
proc_MSS_Crawl 69
proc_MSS_CrawlAdmin 87
proc_MSS_CrawlStoreToJoinAllOngoingCrawls 97
proc_MSS_CreateConfigTransaction 98
proc_MSS_CreateNewComponentRecord 98
proc_MSS_DefragGathererIndexes 165
proc_MSS_DeleteConfigurationProperties 99
proc_MSS_DeleteConfigurationProperty 99
proc_MSS_FlushTemp0 100
proc_MSS_GetAllGathererDatabaseIDs 122
proc_MSS_GetAllPropertyStoreIDs 122
proc_MSS_GetAnchorDocIDs 113
proc_MSS_GetCompiledScopeRules 114
proc_MSS_GetCompiledScopes 114
proc_MSS_GetCompletedScopesCompilationId 114
proc_MSS_GetComponentStatusUpToDate 115
proc_MSS_GetConfigurationProperties 116
proc_MSS_GetConfigurationProperty 116
proc_MSS_GetContentSourceDocCount 117
proc_MSS_GetCrawledProperties 117
proc_MSS_GetCrawledPropMappingUpdates 119
proc_MSS_GetCrawls 120
proc_MSS_GetCurrentRegistryVersion 121
proc_MSS_GetDatabaseID 122
proc_MSS_GetDeleteCrawl 123
proc_MSS_GetDeletedScopesForCompilation 124
proc_MSS_GetDocCount 124
proc_MSS_GetDocStatus 125
proc_MSS_GetError 125
proc_MSS_GetGathererDBs 126
proc_MSS_GetHost 127
proc_MSS_GetIncompleteCommands 127
proc_MSS_GetManagedProperties 128
proc_MSS_GetMappingsForCrawledPropertiesById 131
proc_MSS_GetMasterRole 132
proc_MSS_GetMatrixPIAliases 132
proc_MSS_GetNextCrawlBatch 133
proc_MSS_GetNextDocIDChunk 137
proc_MSS_GetPauseCrawlsReasons 138
proc_MSS_GetPauseStatus 138
proc_MSS_GetPreferredNameList 138
proc_MSS_GetRankingModels 139
proc_MSS_GetSampleExtremes 139
proc_MSS_GetSchemaHighLevelInfo 140
proc_MSS_GetSchemaMappings 141
proc_MSS_GetSchemaParameters 141
proc_MSS_GetScopeRulesForCompilation 142
proc_MSS_GetScopesForCompilation 142
proc_MSS_GetSDID 143
proc_MSS_GetStaticRankingFeatures 143
proc_MSS_GetStatus 143
proc_MSS_GetStatusChangeRequest 144
proc_MSS_GetTopAnchorLinks 146
proc_MSS_InvalidateAdminDocIDChunks 136
proc_MSS_IsPauseCrawlsPortalContentDone 147
proc_MSS_IsSystemPausedForRefactoring 67
proc_MSS_PrepareForAnnotationsUpdate 148
proc_MSS_ProcessCommitted 149
proc_MSS_PushSD 162
proc_MSS_QLog_GetRelevanceUpdateData 163
proc_MSS_Recompile 164
proc_MSS_RemoveConfigTransactions 165
proc_MSS_RemoveCrawlStoreFromAllCrawls 165
proc_MSS_ReportCommandCompleted 166
proc_MSS_ResetCatalog 166
proc_MSS_SetAnnotationsPendingStatus 172
proc_MSS_SetConfigPropertyWithTransaction 166
proc_MSS_SetConfigurationProperty 167
proc_MSS_SetCrawledPropertyIsSampleCacheFull 168
proc_MSS_SetRequestToPauseCrawlsPortalContent 168
proc_MSS_SetStatusChangeRequest 169
proc_MSS_ShareScopesCompilationInfo 169
proc_MSS_TruncateAnnotationsTables 173
proc_MSS_TruncateCleanupTable 171
proc_MSS_UnloadTransactions 171
proc_MSS_UpdateAnchorDocPropsBlob 173
proc_MSS_UpdateCrawlStatistics 173
proc_MSS_UpdateDocCount 175
proc_MSS_UpdateLinksBitmapField 175
proc_MSS_UpdatePropertyStore 175
proc_MSSSetCrawlComponentStatus 167
Property Store IDs Result Set 123

N

Namespaces 46
Normative references 11

O

Office SharePoint Server interface (section 3.1 47, section 3.2 179)
Overview
configuration properties – overview (synopsis) 13
scope compilation – overview (synopsis) 14
status changes – overview (synopsis) 13
Overview (synopsis) 12

P

Parameters - security index 203
Pause Crawls Reasons flag structure 22
Pending Anchor Change 56
Pending Annotations 54
Pending Annotations Status 55
Preconditions 15
Prerequisites 15
proc_MSS_AddAndReturnCrawledProperty method 64

proc_MSS_AddCrawledPropertyCategoryWithDefaults method 65
proc_MSS_CheckIfStatusChangeComplete method 66
proc_MSS_CleanupWithInterval method 178
proc_MSS_CommitConfigTransaction method 68
proc_MSS_CommitTransactions method 148
proc_MSS_CompleteStatusChange method 68
proc_MSS_Crawl method 69
proc_MSS_CrawlAdmin method 87
proc_MSS_CrawlStoreToJoinAllOngoingCrawls method 97
proc_MSS_CreateConfigTransaction method 98
proc_MSS_CreateNewComponentRecord method 98
proc_MSS_DefragGathererIndexes method 165
proc_MSS_DeleteConfigurationProperties method 99
proc_MSS_DeleteConfigurationProperty method 99
proc_MSS_FlushTemp0 method 100
proc_MSS_GetAllGathererDatabaseIDs method 122
proc_MSS_GetAllPropertyStoreIDs method 122
proc_MSS_GetAnchorDocIDs method 113
proc_MSS_GetCompiledScopeRules method 114
proc_MSS_GetCompiledScopes method 114
proc_MSS_GetCompletedScopesCompilationId method 114
proc_MSS_GetComponentStatusUpToDate method 115
proc_MSS_GetConfigurationProperties method 116
proc_MSS_GetConfigurationProperty method 116
proc_MSS_GetContentSourceDocCount method 117
proc_MSS_GetCrawledProperties method 117
proc_MSS_GetCrawledPropMappingUpdates method 119
proc_MSS_GetCrawls method 120
proc_MSS_GetCurrentRegistryVersion method 121
proc_MSS_GetDatabaseID method 122
proc_MSS_GetDeleteCrawl method 123
proc_MSS_GetDeletedScopesForCompilation method 124
proc_MSS_GetDocCount method 124
proc_MSS_GetDocStatus method 125
proc_MSS_GetError method 125
proc_MSS_GetGathererDBs method 126
proc_MSS_GetHost method 127
proc_MSS_GetIncompleteCommands method 127
proc_MSS_GetManagedProperties method 128
proc_MSS_GetMappingsForCrawledPropertiesById method 131
proc_MSS_GetMasterRole method 132
proc_MSS_GetMatrixPIAliases method 132
proc_MSS_GetNextCrawlBatch method 133
proc_MSS_GetNextDocIDChunk method 137
proc_MSS_GetPauseCrawlsReasons method 138
proc_MSS_GetPauseStatus method 138
proc_MSS_GetPreferredNameList method 138
proc_MSS_GetRankingModels method 139
proc_MSS_GetSampleExtremes method 139
proc_MSS_GetSchemaHighLevelInfo method 140
proc_MSS_GetSchemaMappings method 141
proc_MSS_GetSchemaParameters method 141
proc_MSS_GetScopeRulesForCompilation method 142
proc_MSS_GetScopesForCompilation method 142
proc_MSS_GetSDID method 143
proc_MSS_GetStaticRankingFeatures method 143
proc_MSS_GetStatus method 143
proc_MSS_GetStatusChangeRequest method 144
proc_MSS_GetTopAnchorLinks method 146
proc_MSS_InvalidateAdminDocIDChunks method 136
proc_MSS_IsPauseCrawlsPortalContentDone method 147
proc_MSS_IsSystemPausedForRefactoring method 67
proc_MSS_PrepareForAnnotationsUpdate method 148
proc_MSS_ProcessCommitted method 149
proc_MSS_PushSD method 162
proc_MSS_QLog_GetRelevanceUpdateData method 163
proc_MSS_Recompile method 164
proc_MSS_RemoveConfigTransactions method 165
proc_MSS_RemoveCrawlStoreFromAllCrawls method 165
proc_MSS_ReportCommandCompleted method 166
proc_MSS_ResetCatalog method 166
proc_MSS_SetAnnotationsPendingStatus method 172
proc_MSS_SetConfigPropertyWithTransaction method 166
proc_MSS_SetConfigurationProperty method 167
proc_MSS_SetCrawledPropertyIsSampleCacheFull method 168
proc_MSS_SetRequestToPauseCrawlsPortalContent method 168
proc_MSS_SetStatusChangeRequest method 169
proc_MSS_ShareScopesCompilationInfo method 169
proc_MSS_TruncateAnnotationsTables method 173
proc_MSS_TruncateCleanupTable method 171
proc_MSS_UnloadTransactions method 171
proc_MSS_UpdateAnchorDocPropsBlob method 173
proc_MSS_UpdateCrawlStatistics method 173
proc_MSS_UpdateDocCount method 175
proc_MSS_UpdateLinksBitmapField method 175
proc_MSS_UpdatePropertyStore method 175
proc_MSSSetCrawlComponentStatus method 167
Product behavior 204
Project Identifier simple type 16
Project Name simple type 21
Property Store IDs Result Set method 123
Protocol overview
  configuration properties 13
  scope compilation 14
  status changes 13

R

References 11
  informative 11
  normative 11
Relationship to other protocols 14

Reported Crawl Error 56
Request master merge sequence 193
Requested Delete Crawls 57
Result sets - messages
  Scope Compilation 45

S

Scope compilation
  overview 14
Scope Compilation result set 45
Scope compilation sequence (section 3.2.5.7 193, section 3.4.5.5 199)
Scope Rules to Compile 59
Scopes to Compile 59
Search Catalog Status simple type 20
Search Component Name 62
Security
  implementer considerations 203
  parameter index 203
Sequences
  anchor text crawl 189
  configuration properties 199
  configuration property 192
  delete crawl (section 3.2.5.3 185, section 3.4.5.3 199)
  full crawl (section 3.2.5.1 179, section 3.4.5.1 198)
  incremental crawl (section 3.2.5.2 182, section 3.4.5.2 199)
  request master merge 193
  scope compilation (section 3.2.5.7 193, section 3.4.5.5 199)
  status change 191
Sequencing rules
  client (section 3.2.5 179, section 3.4.5 198)
  server 196
Server
  abstract data model (section 3.1.1 47, section 3.3.1 194)
  Gatherer DB IDs Result Set method 122
  Get Mappings for Crawled Properties by Identifier Result Set method 132
  higher-layer triggered events (section 3.1.4 63, section 3.3.4 195)
  initialization (section 3.1.3 63, section 3.1.3 63, section 3.3.3 195)
  local events (section 3.1.7 178, section 3.3.7 198)
  message processing 196
  Office SharePoint Server interface 47
  overview 47
  proc_MSS_AddAndReturnCrawledProperty method 64
  proc_MSS_AddCrawledPropertyCategoryWithDefaults method 65
  proc_MSS_CheckIfStatusChangeComplete method 66
  proc_MSS_CleanupWithInterval method 178
  proc_MSS_CommitConfigTransaction method 68
  proc_MSS_CommitTransactions method 148
  proc_MSS_CompleteStatusChange method 68
  proc_MSS_Crawl method 69
  proc_MSS_CrawlAdmin method 87
  proc_MSS_CrawlStoreToJoinAllOngoingCrawls method 97
  proc_MSS_CreateConfigTransaction method 98
  proc_MSS_CreateNewComponentRecord method 98
  proc_MSS_DefragGathererIndexes method 165
  proc_MSS_DeleteConfigurationProperties method 99
  proc_MSS_DeleteConfigurationProperty method 99
  proc_MSS_FlushTemp0 method 100
  proc_MSS_GetAllGathererDatabaseIDs method 122
  proc_MSS_GetAllPropertyStoreIDs method 122
  proc_MSS_GetAnchorDocIDs method 113
  proc_MSS_GetCompiledScopeRules method 114
  proc_MSS_GetCompiledScopes method 114
  proc_MSS_GetCompletedScopesCompilationId method 114
  proc_MSS_GetComponentStatusUpToDate method 115
  proc_MSS_GetConfigurationProperties method 116
  proc_MSS_GetConfigurationProperty method 116
  proc_MSS_GetContentSourceDocCount method 117
  proc_MSS_GetCrawledProperties method 117
  proc_MSS_GetCrawledPropMappingUpdates method 119
  proc_MSS_GetCrawls method 120
  proc_MSS_GetCurrentRegistryVersion method 121
  proc_MSS_GetDatabaseID method 122
  proc_MSS_GetDeleteCrawl method 123
  proc_MSS_GetDeletedScopesForCompilation method 124
  proc_MSS_GetDocCount method 124
  proc_MSS_GetDocStatus method 125
  proc_MSS_GetError method 125
  proc_MSS_GetGathererDBs method 126
  proc_MSS_GetHost method 127
  proc_MSS_GetIncompleteCommands method 127
  proc_MSS_GetManagedProperties method 128
  proc_MSS_GetMappingsForCrawledPropertiesById method 131
  proc_MSS_GetMasterRole method 132
  proc_MSS_GetMatrixPIAliases method 132
  proc_MSS_GetNextCrawlBatch method 133
  proc_MSS_GetNextDocIDChunk method 137
  proc_MSS_GetPauseCrawlsReasons method 138
  proc_MSS_GetPauseStatus method 138
  proc_MSS_GetPreferredNameList method 138
  proc_MSS_GetRankingModels method 139
  proc_MSS_GetSampleExtremes method 139
  proc_MSS_GetSchemaHighLevelInfo method 140
  proc_MSS_GetSchemaMappings method 141
  proc_MSS_GetSchemaParameters method 141
  proc_MSS_GetScopeRulesForCompilation method 142

proc_MSS_GetScopesForCompilation method 142
proc_MSS_GetSDID method 143
proc_MSS_GetStaticRankingFeatures method 143
proc_MSS_GetStatus method 143
proc_MSS_GetStatusChangeRequest method 144
proc_MSS_GetTopAnchorLinks method 146
proc_MSS_InvalidateAdminDocIDChunks method 136
proc_MSS_IsPauseCrawlsPortalContentDone method 147
proc_MSS_IsSystemPausedForRefactoring method 67
proc_MSS_PrepareForAnnotationsUpdate method 148
proc_MSS_ProcessCommitted method 149
proc_MSS_PushSD method 162
proc_MSS_QLog_GetRelevanceUpdateData method 163
proc_MSS_Recompile method 164
proc_MSS_RemoveConfigTransactions method 165
proc_MSS_RemoveCrawlStoreFromAllCrawls method 165
proc_MSS_ReportCommandCompleted method 166
proc_MSS_ResetCatalog method 166
proc_MSS_SetAnnotationsPendingStatus method 172
proc_MSS_SetConfigPropertyWithTransaction method 166
proc_MSS_SetConfigurationProperty method 167
proc_MSS_SetCrawledPropertyIsSampleCacheFull method 168
proc_MSS_SetRequestToPauseCrawlsPortalContent method 168
proc_MSS_SetStatusChangeRequest method 169
proc_MSS_ShareScopesCompilationInfo method 169
proc_MSS_TruncateAnnotationsTables method 173
proc_MSS_TruncateCleanupTable method 171
proc_MSS_UnloadTransactions method 171
proc_MSS_UpdateAnchorDocPropsBlob method 173
proc_MSS_UpdateCrawlStatistics method 173
proc_MSS_UpdateDocCount method 175
proc_MSS_UpdateLinksBitmapField method 175
proc_MSS_UpdatePropertyStore method 175
proc_MSSSetCrawlComponentStatus method 167
Property Store IDs Result Set method 123
sequencing rules 196
timer events (section 3.1.6 178, section 3.3.6 198)
timers (section 3.1.2 63, section 3.3.2 195)
Server - Office SharePoint Server
Anchor Change 56
Anchor Document Properties Blob 62
Anchor Text Info 53
Annotations 62
Changed Anchor Committed Documents 57
Changed Anchor Deleted Documents 58
Changed Anchor Source Documents 57
Changed Anchor Target Documents 57
Colleague Links 53
Commands 63
Compiled Scope Rules 60
Compiled Scopes 60
Component Activity 61
Configuration Properties 61
Crawl Component Status 58
crawl queue 50
Crawl Status 47
Crawl Store Status 59
crawl URL history 47
crawled Hosts 53
Current DocID Chunk 57
Current DocID Chunks 56
Definitions 62
Deleted Scopes 60
Deleted Scopes to Compile 60
Deleted URL 50
links 51
Local Completed Crawls 59
Local Crawl Components 58
Metadata store crawl components 63
Pending Anchor Change 56
Pending Annotations 54
Pending Annotations Status 55
reported Crawl Error 56
Requested Delete Crawls 57
Scope Rules to Compile 59
Scopes to Compile 59
Search Component Name 62
Social Distance Deleted Property 55
Social Distance New Property (section 3.1.1.13 56, section 3.1.1.43 63)
Social Distance Property 55
UserID DocID Pair 58
Simple data types
Anchor Change Type 22
Content Source Status 20
Crawl Completing Sub Status 17
Crawl Component State 19
Crawl Crawling Sub Status 17
Crawl Error Codes 18
Crawl Error Levels 18
Crawl Error Logging Levels 18
Crawl Request Type 16
Crawl Starting Sub Status 17
Crawl Status 17
Crawl Stopping Sub Status 18
Crawl Type 16
Index Type 21
Item Delete Reasons 22
Item Type 21
Managed Type 19
Project Identifier 16
Project Name 21
Search Catalog Status 20
Transaction Scope 20
Transaction Type 19
Update Queue Type 21

Simple types - overview 46
Social Distance Deleted Property 55
Social Distance New Property (section 3.1.1.13 56,
section 3.1.1.43 63)
Social Distance Property 55
Standards assignments 15
Status change sequence 191
Status changes
overview 13
Structures
binary 24
table and view 45
XML 45

T

Table structures - overview 45
Timer events
client 199
server (section 3.1.6 178, section 3.3.6 198)
Timers
client (section 3.2.2 179, section 3.4.2 198)
server (section 3.1.2 63, section 3.3.2 195)
Tracking changes 207
Transaction Commit Record binary structure 37
Transaction Data Blob binary structure 44
Transaction Flags flag structure 23
Transaction Scope simple type 20
Transaction string binary structure 45
Transaction Type simple type 19
Transactions Commit Structure binary structure 44
Transport 16
Triggered events - higher-layer
client (section 3.2.4 179, section 3.4.4 198)
server (section 3.1.4 63, section 3.3.4 195)
Types
complex 46
simple 46

U

Update Anchor Add Record binary structure 24
Update Anchor Add Structure binary structure 25
Update Queue Type simple type 21
UserID DocID Pair 58

V

Vendor-extensible fields 15
Versioning 15
View structures - overview 45

X

XML structures 45