[MS-UTSP]:
SharePoint Usage Tracking Stored Procedures 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	Editorial	Changed language and formatting in the technical content.
07/23/2010	1.05	Minor	Clarified the meaning of the technical content.
09/27/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	1.05	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	1.05	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the SharePoint Usage Tracking Stored Procedures Protocol, which supports the collection, storage, and reporting of usage and diagnostic data.

1.1 Glossary

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

Coordinated Universal Time (UTC)
GUID
user agent

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

anchor text
back-end database server
best bet
content source
content type identifier
crawl
crawl component
crawler
farm
front-end Web server
full-text index catalog
item
login name
metadata store
provision
query component
relative path
request identifier
result set
return code
search application
search query
search scope index
session identifier
site
site collection
site collection identifier
site identifier
site subscription identifier
Status-Code
stored procedure
timestamp
transaction
Transact-Structured Query Language (T-SQL)
Uniform Resource Identifier (URI)
Uniform Resource Locator (URL)
view
Web application
Web application identifier

XML namespace
XML namespace prefix

The following terms are specific to this document:

farm identifier: A GUID that identifies the server farm where a request originated.

logging provider: A module that is used to store usage and diagnostic data for a server farm and can provide that data in response to queries from protocol clients.

partitioned table: A table that is built on a partition scheme and whose data is divided horizontally into units that can be spread across more than one file group in a database.

retention period: A configuration setting that specifies the number of days to maintain usage data that is stored by a logging provider.

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

1.2 References

1.2.1 Normative References

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

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

[MS-SQL] Microsoft Corporation, "SQL Server 2000 Architecture and XML/Internet Support", Volume 1 of Microsoft SQL Server 2000 Reference Library, Microsoft Press, 2001, ISBN 0-7356-1280-3, http://msdn.microsoft.com/en-us/library/dd631854(v=SQL.10).aspx

[MS-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

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

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

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

1.2.2 Informative References

[LotusNotes] IBM, "Lotus Notes - Business email solution", http://www-01.ibm.com/software/lotus/products/notes/

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

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

1.3 Protocol Overview (Synopsis)

This protocol supports the storage, retrieval and reporting of usage and diagnostic data. It is used by the protocol client to store usage data of various kinds, including user request information, performance counters, data on slow or expensive queries, and other relevant performance data. The data is stored using modules referred to here as logging providers. It is extensible, so providers can be developed, installed, and provisioned on a deployed **farm**.

This document includes both the core logging infrastructure stored procedures as well as stored procedures created by a subset of protocol client implementations. A minimal implementation of this protocol includes only the result sets, tables, and stored procedures that are related to the core logging procedures.

1.4 Relationship to Other Protocols

This protocol uses the Tabular Data Stream Protocol, as described in [MS-TDS], as its transport between the **front-end Web server** acting as a client (or possibly other clients), and the **back-end database server**, acting as a server.

This is shown in the following layered diagram:

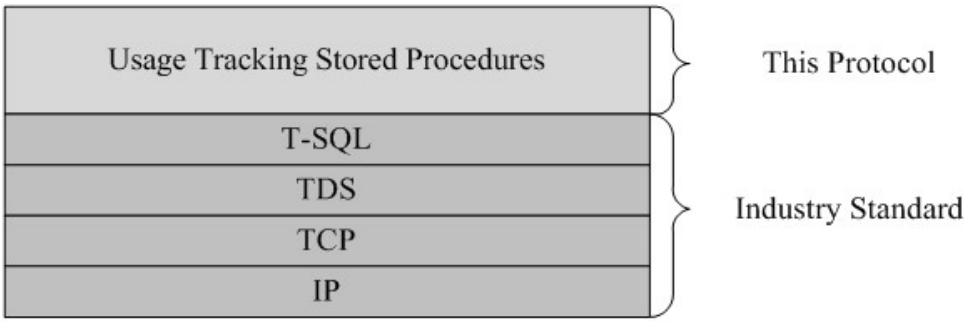

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

The operations described by this protocol operate between a client and a back-end database server. The client is expected to know the location and connection information for the databases.

This protocol requires that the protocol client has appropriate permissions to call the stored procedures stored on the back-end database server.

1.6 Applicability Statement

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

1.7 Versioning and Capability Negotiation

Security and Authentication Methods: This protocol supports the SSPI and SQL Authentication with the protocol server role as described in [MS-TDS].

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

[MS-TDS] specifies the transport protocol used to call the stored procedures, query SQL tables, get **return codes**, and return **result sets**.

2.2 Common Data Types

None.

2.2.1 Simple Data Types and Enumerations

None.

2.2.2 Bit Fields and Flag Structures

None.

2.2.3 Binary Structures

None.

2.2.4 Result Sets

This protocol specifies the following result sets.

2.2.4.1 prc_EnsureIndexHelper.ResultSet0

The **prc_EnsureIndexHelper.Result0** result set MUST be ignored by the client.

This result set is defined using **T-SQL** syntax as follows:

```
prc_EnsureIndexHelper.ResultSet0.UnnamedColumn0 nvarchar(283),
```

prc_EnsureIndexHelper.ResultSet0.UnnamedColumn0: This value MUST be ignored by the client.

2.2.4.2 proc_GetMostActiveUsers.ResultSet0

The **proc_GetMostActiveUsers.ResultSet0** result set returns information about the most active users. This result set MUST be returned and the rows MUST be arranged in descending order of number of HTTP requests made by the user.

This result set is defined using T-SQL syntax as follows:

```
User nvarchar(300),
Hits int,
LastAccessTime datetime,
```

User: Contains the **login name** for the user that made the request.

Hits: Contains the number of HTTP requests made by the user.

LastAccessTime: Contains the **UTC** date of the most recent request done during the specified time span.

2.2.4.3 proc_GetSlowestPages.ResultSet0

The **proc_GetSlowestPages.ResultSet0** result set returns information about the slowest pages. This result set MUST be returned and the rows MUST be arranged in descending order of average page latency on the server side.

This result set is defined using T-SQL syntax as follows:

```
Url nvarchar(1536),
AverageDuration float,
MaximumDuration float,
MinimumDuration float,
AverageQueryCount int,
MaximumQueryCount smallint,
MinimumQueryCount smallint,
TotalPageHits int,
```

Url: Contains the full URL of the page.

AverageDuration: Contains the average server-side latency in seconds for all requests to this page.

MaximumDuration: Contains the maximum server-side latency in seconds for all requests to this page.

MinimumDuration: Contains the minimum server-side latency in seconds for all requests to this page.

AverageQueryCount: Contains the average number of queries sent to the back-end database server over all requests to this page.

MaximumQueryCount: Contains the maximum number of queries sent to the back-end database server over all requests to this page.

MinimumQueryCount: Contains the minimum number of queries sent to the back-end database server over all requests to this page.

TotalPageHits: Contains the total number of requests recorded against this page.

2.2.4.4 Crawl Processing Per Activity Result Set

The **Crawl Processing Per Activity** result set returns the time spent on different crawling activities by minute. The **LogTime** field MUST be rounded to the lowest minute. The result set MUST be arranged in ascending order of the **LogTime** field.

```
LogTime datetime,
ProtocolHandlersTotal int,
StandardPropertiesTotal int,
FilterInitializationTotal int,
WaitTotal int,
FilteringTotal int,
WordBreakingTotal int,
OnDataChangeTotal int,
```

```
ChunkProcessingTotal int,
ProcessWordsTotal int,
AddCompletedTotal int,
```

LogTime: The UTC date and time when an entry was logged.

ProtocolHandlersTotal: The aggregated time in milliseconds spent in protocol handlers within **crawl components**. (A protocol handler is a component used by the search service to access specific systems, such as **sites (2)**, file systems, and external Web sites.

StandardPropertiesTotal: The aggregated time in milliseconds spent on retrieving common metadata properties from an **item**'s metadata.

FilterInitializationTotal: The aggregated time in milliseconds spent on binding to protocol handlers.

WaitTotal: The aggregated time in milliseconds spent by an item waiting to be processed once it is queued inside the crawl component.

FilteringTotal: The aggregated time in milliseconds spent on extracting text and properties from items that are crawled.

WordBreakingTotal: The aggregated time in milliseconds spent on word breaking.

OnDataChangeTotal: The aggregated time in milliseconds spent by crawl component plug-ins to register an item as the plug-ins begin to receive crawled data.

ChunkProcessingTotal: The aggregated time in milliseconds spent by the crawl component plug-ins on processing chunks of data for items that are crawled.

ProcessWordsTotal: The aggregated time in milliseconds spent by the crawl component plug-ins on processing separate words for items that are crawled.

AddCompletedTotal: The aggregated time in milliseconds spent by the crawl component plug-ins on completing processing of items that are crawled.

2.2.4.5 Crawl Processing Per Component Result Set

The **Crawl Processing Per Component** result set returns the time spent by different parts of crawl components each minute. The **LogTime** field MUST be rounded to the lowest minute. The result set MUST be arranged in ascending order of the **LogTime** field.

```
LogTime datetime,
Sts3PhTotal int,
Sts4PhTotal int,
FilePhTotal int,
HttpPhTotal int,
WaitTotal int,
FilterInitializationTotal int,
SpsPhTotal int,
IndexerPluginTotal int,
ArpiPluginTotal int,
GathererTotal int,
BdcPhTotal int,
LotusPhTotal int,
RankingPhTotal int,
```

```
OtherPhTotal int,
FilterPhTotal int,
WordBreakerTotal int,
ScopesPluginTotal int,
AnchorTotal int,
OtherTotal int,
```

LogTime: The UTC time when an entry was logged rounded to minutes.

Sts3PhTotal: The aggregated time in milliseconds spent by the protocol handler, which is used for processing the "sts3://" protocol and related URLs.

Sts4PhTotal: The aggregated time in milliseconds spent by the protocol handler, which is used for processing the "sts4://" protocol and related URLs.

FilePhTotal: The aggregated time in milliseconds spent by file system protocol handler.

HttpPhTotal: The aggregated time in milliseconds spent by HTTP protocol handler.

WaitTotal: The aggregated time in milliseconds spent by an item waiting to be processed once it is queued inside the crawl component.

FilterInitializationTotal: The aggregated time in milliseconds spent on initializing of data filter within crawl components.

SpsPhTotal: The aggregated time in milliseconds spent by the User Profiles protocol handler for processing the "sps3://" and "sps3s://" protocols.

IndexerPluginTotal: The aggregated time in milliseconds spent by the module of building **full-text index catalog**.

ArpiPluginTotal: The aggregated time in milliseconds spent by the module of archiving and storing extracted properties in **metadata stores**.

GathererTotal: The aggregated time in milliseconds spent by the module that manages crawling.

BdcPhTotal: The aggregated time in milliseconds spent by Business Data Connectivity protocol handler.

LotusPhTotal: The aggregated time in milliseconds spent by Lotus Notes [LotusNotes] protocol handler.

RankingPhTotal: The aggregated time in milliseconds spent by ranking protocol handler.

OtherPhTotal: The aggregated time in milliseconds spent by protocol handlers other than those specifically called out as columns in this result set (using **Sts3PhTotal**, **STS4PhTotal**, **FilePhTotal**, and so on.)

FilterPhTotal: The aggregated time in milliseconds spent on text and properties extraction from items that are crawled.

WordBreakerTotal: The aggregated time in milliseconds spent by word breaking module.

ScopesPluginTotal: The aggregated time in milliseconds spent to build **search scope index** of crawled items.

AnchorTotal: The aggregated time in milliseconds spent on **anchor text** extraction from crawled items.

OtherTotal: The aggregated time in milliseconds spent on operations other than those covered specifically by one of the other columns of this result set.

2.2.4.6 Crawl Processing Stage Per Item Result Set

The **Crawl Processing Stage Per Item** result set returns the average time spent by crawl components on different stages of crawling per item each minute. The **LogTime** field MUST be rounded to the lowest minute. The result set MUST be arranged in ascending order of the **LogTime** field.

```
LogTime datetime,
LoadingAvg bigint,
WaitingAvg bigint,
ConnectAvg bigint,
StandardPropertiesAvg bigint,
FilteringAvg bigint,
LinkCommitSyncCompletionAvg bigint,
DelayedAvg bigint,
CommitCompletedAvg bigint,
PropertyStoreAvg bigint,
FlushMergeAsyncCompleteAvg bigint,
PropagationAsyncCompleteAvg bigint,
```

LogTime: The UTC time when an entry was logged rounded to the lowest minute.

LoadingAvg: The average time in milliseconds spent on loading items into the crawl component's internal queue.

WaitingAvg: The average time in milliseconds spent by an item waiting to be processed once it is queued inside the crawl component.

ConnectAvg: The average time in milliseconds spent on establishing the connection to the **content source** of the items.

StandardPropertiesAvg: The average time in milliseconds spent on retrieving common metadata properties from an item's metadata.

FilteringAvg: The average time in milliseconds spent by the crawl component to receive all the chunks of data for an item, tokenize it and give all the information to the plug-ins.

LinkCommitSyncCompletionAvg: The average time in milliseconds taken to register preliminary completion of crawling an item and to process the item's links.

DelayedAvg: The average time in milliseconds that an item is delayed due to internal crawl throttling for that item's host.

CommitCompletedAvg: The average time in milliseconds of the time spent registering an item as having been crawled in a given minute.

PropertyStoreAvg: The average time in milliseconds of the time spent persisting properties in the metadata store.

FlushMergeAsyncCompleteAvg: The average time in milliseconds spent on persisting data to the full-text index catalog.

PropagationAsyncCompleteAvg: The average time in milliseconds spent on propagating data to the **query components (2)**.

2.2.4.7 Crawl Queue Result Set

The **Crawl Queue** result set returns the number of items in **crawl** queues. Each row of the result set MUST correspond to the number of items in the queue of links to be processed and the number of items in the queue of transactions for each minute. The **LogTime** field MUST be rounded to the lowest minute. The result set MUST be arranged in ascending order of the **LogTime** field.

```
LogTime datetime,
TransactionsQueuedTotal int,
LinksToBeProcessedTotal int,
```

LogTime: The UTC time when an entry was logged rounded to the lowest minute.

TransactionsQueuedTotal: The number items in the queue of **transactions (1)** for the given minute.

LinksToBeProcessedTotal: The number items in the queue of links to be crawled for the given minute.

2.2.4.8 Crawl Rate Per Content Source Result Set

The **Crawl Rate Per Content Source** result set returns the number of items crawled from each type of content sources. Each row of the result set MUST correspond to the number of items crawled from a single content source within each minute. The **LogTime** field MUST be rounded to the lowest minute. The result set MUST be arranged in ascending order of the **LogTime** field.

```
ContentSourceName nvarchar(100),
LogTime datetime,
NumDocumentsTotal int,
```

ContentSourceName: The name of the content source.

LogTime: The UTC time of statistics entry rounded to minutes.

NumDocumentsTotal: The number of items crawled from a content source at a corresponding minute.

2.2.4.9 Crawl Rate Per Type Result Set

The **Crawl Rate Per Type** result set returns the counts of the specific crawl actions performed for the items that were encountered during the crawl. Each row of the result set MUST correspond to the number of items crawled within each minute. The **LogTime** field MUST be rounded to the lowest minute. The result set MUST be arranged in ascending order of the **LogTime** field.

```
LogTime datetime,
ModifiedTotal int,
ErrorTotal int,
DeleteTotal int,
NotModifiedTotal int,
SecurityOnlyTotal int,
```

```
NotIndexedTotal int,
```

LogTime: The UTC time of statistics entry rounded to the lowest minute.

ModifiedTotal: The number of items encountered during the given minute that were updated since the last crawl.

ErrorTotal: The number of items encountered during the given minute that were not crawled because of errors.

DeleteTotal: The number of items encountered during the given minute that were deleted from the content source since the last crawl.

NotModifiedTotal: The number of items encountered during the given minute that were not modified since the last crawl.

SecurityOnlyTotal: The number of items encountered during the given minute for which security data was modified since the last crawl.

NotIndexedTotal: The number of items encountered during the given minute that were explicitly ignored by the crawl.

2.2.4.10 Query Latency Result Set

The **Query Latency** result set returns average time of **search queries** processing on different stages. Each row of the result set MUST correspond to the average time spent on each search query processing stage for each minute. The **LogTime** field MUST be rounded to the lowest minute. The time spent on each stage MUST be counted in milliseconds and averaged by the number of search queries. The result set MUST be arranged in ascending order of the **LogTime** field.

```
LogTime datetime,
NumQueries int,
MsSpentInTripoliQueryAvg int,
MsSpentInPropStoreQueryAvg int,
MsSpentInDupesRemovalAvg int,
MsSpentInSecurityTrimmingAvg int,
MsSpentInFinalSortAvg int,
MsSpentInBestBetsResultsRetrievalAvg int,
MsSpentInMultipleResultsRetrievalAvg int,
MsSpentInHighConfidenceResultsRetrievalAvg int,
MsSpentInResultsPopulationAvg int,
MsSpentOtherAvg int,
```

LogTime: The UTC time when the data was logged rounded to the lowest minute.

NumQueries: The number of search queries processed within the corresponding minute.

MsSpentInTripoliQueryAvg: The average time in milliseconds spent on searching in full-text index catalog.

MsSpentInPropStoreQueryAvg: The average time in milliseconds spent on retrieving information from metadata stores plus **MsSpentInTripoliQueryAvg**.

MsSpentInDupesRemovalAvg: The average time in milliseconds spent on duplicates removing plus **MsSpentInPropStoreQueryAvg**.

MsSpentInSecurityTrimmingAvg: The average time in milliseconds spent on security trimming plus **MsSpentInDupesRemovalAvg**.

MsSpentInFinalSortAvg: The average time in milliseconds spent on sorting of results plus **MsSpentInSecurityTrimmingAvg**.

MsSpentInBestBetsResultsRetrievalAvg: The average time in milliseconds spent on calculating **best bets** plus **MsSpentInFinalSortAvg**.

MsSpentInMultipleResultsRetrievalAvg: The average time in milliseconds spent on retrieving multiple results plus **MsSpentInBestBetsResultsRetrievalAvg**.

MsSpentInHighConfidenceResultsRetrievalAvg: The average time in milliseconds spent on retrieving high confidence results plus **MsSpentInMultipleResultsRetrievalAvg**.

MsSpentInResultsPopulationAvg: The average time in milliseconds spent on results population plus **MsSpentInHighConfidenceResultsRetrievalAvg**.

MsSpentOtherAvg: The average time in milliseconds spent on other operations plus **MsSpentInResultsPopulationAvg**.

2.2.4.11 Recent Crawl Stat Result Set

The **Recent Crawl Stat** result set returns information about the number of items crawled within the last 5 minutes. The result set MUST contain one row if crawl data exists for the **search application** within the last 5 minutes or zero rows otherwise.

```
MaxLogTime datetime,
MinLogTime datetime,
NumDocuments int,
```

MaxLogTime: The UTC time when the last item was crawled within the last 5 minutes. This value MUST be rounded to the lowest minute.

MinLogTime: The UTC time when the first item was crawled within the last 5 minutes. This value MUST be rounded to the lowest minute.

NumDocuments: The number of items crawled within the last 5 minutes.

2.2.4.12 Recent Query Stat Result Set

The **Recent Query Stat** result set returns information about the number of processed search queries within the last 5 minutes. The result set MUST contain one row if search data exists for the search application within the last 5 minutes or zero rows otherwise.

```
MaxLogTime datetime,
MinLogTime datetime,
QueryCount int,
```

MaxLogTime: The UTC time when the last search query was processed within the last 5 minutes. The time MUST be rounded to the lowest minute.

MinLogTime: The UTC time when the first search query was processed within the last 5 minutes. The time MUST be rounded to the lowest minute.

QueryCount: The number of queries processed within the last 5 minutes.

2.2.4.13 prc_GetLastUTCDate.prc_GetLastUTCDate.Default.ResultSet0

This result set MUST be returned and MUST contain one row that corresponds to the computer and logging provider that meets the criteria defined in the input parameters if such a computer and logging provider exist. The **prc_GetLastUTCDate.prc_GetLastUTCDate.Default.ResultSet0** result set is defined using T-SQL syntax, as follows:

```
UTCDate datetime,
```

UTCDate: Contains the last UTC date on which data was written from a specified computer for the given logging provider.

2.2.4.14 Search_GetCrawlRatePerContentSourceSummary.ResultSet0

The **Crawl Rate Per Content Source Summary** result set returns the average crawl rate by each type of content source during each crawl. Each row of the result set MUST correspond to the average number of items crawled per minute for each crawl of each content source taking place within specified time span. The result set MUST be arranged in ascending order of the **CrawlStartTime** field.

```
CrawlId int,
ContentSourceName nvarchar(100),
CrawlStartTime datetime,
CrawlDurationMinutes int,
CrawlRate int,
```

CrawlId: The unique identifier of the crawl.

ContentSourceName: The name of the content source.

CrawlStartTime: The UTC time of when the first item was crawled from the content source within a given time span.

CrawlDurationMinutes: The difference in minutes between times when the last and the first items were crawled from the content source within a given time span.

CrawlRate: The average number of items crawled from the content source per minute during CrawlDurationMinutes time span.

2.2.4.15 Search_GetCrawlRatePerTypeSummary.ResultSet0

The **Crawl Rate Per Type Summary** result set returns the crawl rate and statistics on the actions performed during the crawl for a search application within a given time span. The result set MUST contain one row if data exists for the search application for this time period, or zero rows otherwise.

```
CrawlRate int,
ItemsTotal int,
AddedModifiedTotal int,
NoIndexTotal int,
DeletedTotal int,
NotModifiedTotal int,
SecurityOnlyTotal int,
```

```
ErrorTotal int,
```

CrawlRate: The average number of items processed within a given time span.

ItemsTotal: The total number of items processed within a given time span.

AddedModifiedTotal: The total number of items encountered within a given time span that were updated since the last crawl.

NoIndexTotal: The total number of items encountered within a given time span that were explicitly ignored by the crawl.

DeletedTotal: The total number of items encountered within a given time span that were deleted from the content source since the last crawl.

NotModifiedTotal: The total number of items encountered within a given time span that were not modified since the last crawl.

SecurityOnlyTotal: The total number of items encountered within a given time span for which security data was modified since the last crawl.

ErrorTotal: The total number of items encountered within a given time span that were not crawled because of errors.

2.2.5 Tables and Views

This protocol specifies the following tables and views.

2.2.5.1 BlockingQueries

This **BlockingQueries** view contains information about queries that were blocked in the back-end database server due to contention for resources needed by both the blocking and waiting queries. The blocking query is either using, or is itself waiting for, a resource needed by the waiting query.

```
PartitionId tinyint NOT NULL,
RowId uniqueidentifier NOT NULL,
LogTime datetime NOT NULL,
MachineName nvarchar(128) NOT NULL,
Database_Name nvarchar(max) NULL,
Resource_Type nvarchar(64) NULL,
Resource_Name nvarchar(max) NULL,
Wait_Mode nvarchar(64) NULL,
Block_Mode nvarchar(64) NULL,
Last_Execution_Time datetime NULL,
Waiting_Time bigint NULL,
Waiting_Sid bigint NULL,
Blocking_Sid bigint NULL,
Blocking_Blocker_Sid bigint NULL,
Waiting_Resource nvarchar(max) NULL,
Waiting_Type nvarchar(max) NULL,
Blocking_Database_Name nvarchar(max) NULL,
Blocking_User_Name nvarchar(max) NULL,
Blocking_Machine nvarchar(max) NULL,
Blocking_Process_Id nvarchar(max) NULL,
Blocking_Statement_Start int NULL,
Blocking_Statement_End int NULL,
```

```
Waiting_Statement_Start int NULL,
Waiting_Statement_End int NULL,
Blocking_Query_Text nvarchar(max) NULL,
Waiting_Query_Text nvarchar(max) NULL,
RowCreatedTime datetime NOT NULL,
```

PartitionId: The identifier of the **partitioned table** associated with the row.

RowId: The unique identifier of this row.

LogTime: The UTC date indicating when the query that caused the blocking was initiated.

MachineName: The name of the computer from which the **logging provider**'s data was written.

Database_Name: The name of the database on which the blocking and waiting queries occurred.

Resource_Type: The type of resource for which the queries contended.

Resource_Name: The name or identifier of the particular resource for which the queries contended.

Wait_Mode: The requested locking mode of the waiting query.

Block_Mode: The requested locking mode of the blocking query.

Last_Execution_Time: Timestamp when the request arrived.

Waiting_Time: The duration in milliseconds of the blocked state of the query.

Waiting_Sid: Identifier of the session that executed the waiting query.

Blocking_Sid: Identifier of the session that is blocking the waiting query.

Blocking_Blocker_Sid: Identifier of the session that is blocking the blocking query. If this column is NULL, the blocking_sid was holding the lock that was blocking the waiting query. If this column is NOT NULL, the request itself was blocked by the query this session was executing.

Waiting_Resource: Returns a value identifying the database resource for which the request was waiting.

Waiting_Type: Returns a value identifying the type of lock for which the query was waiting.

Blocking_Database_Name: Name of the database against which the request was executed.

Blocking_User_Name: Login name that executed the blocking query.

Blocking_Machine: Name of the client workstation that executed the blocking query. The value is NULL for internal sessions.

Blocking_Process_Id: Process identifier of the client application that executed the blocking query. The value is NULL for internal sessions.

Blocking_Statement_Start: The zero-based offset of the beginning of the blocking statement in Blocking_Query_Text.

Blocking_Statement_End: The zero-based offset of the end of the blocking statement in Blocking_Query_Text.

Waiting_Statement_Start: The zero-based offset of the beginning of the waiting statement in Waiting_Query_Text.

Waiting_Statement_End: The zero-based offset of the end of the waiting statement in Waiting_Query_Text.

Blocking_Query_Text: Complete text of the blocking SQL query. MUST be NULL for encrypted objects.

Waiting_Query_Text: Complete text of the waiting SQL query. MUST be NULL for encrypted objects.

RowCreatedTime: The UTC date indicating when the row was created.

2.2.5.2 fn_PartitionIdRangeMonthly

The **fn_ PartitionIdRangeMonthly** function is used to retrieve a result set containing the list of partitioned table identifiers for the data of a logging provider over a given time span. The T-SQL syntax for the function is as follows.

```
FUNCTION [dbo].[fn_PartitionIdRangeMonthly]
(
    @BeginTime datetime,
    @EndTime datetime,
)
```

@BeginTime: The UTC start time of the time span for which partitioned table identifiers are requested. The **@BeginTime** parameter MUST be specified and MUST NOT be NULL.

@EndTime: The UTC end time of the time span for which partitioned table identifiers are requested. The **@EndTime** parameter MUST be specified and MUST be greater than **@BeginTime**.

This function MUST return a table that contains the list of partitioned table identifiers for the given time span. If there are no partitioned table identifiers for the given time span, the function MUST return a table with zero rows.

```
PartitionId tinyint NULL,
```

PartitionId: The identifier for the partitioned table of the logging provider.

2.2.5.3 RequestUsage

The **RequestUsage** view is called to retrieve all information stored in the partitioned tables for the **RequestUsage** logging provider. This view can be queried to return data pertaining to all client web requests that have been made since the beginning of the **retention period**.

```
PartitionId tinyint NOT NULL,
RowId uniqueidentifier NOT NULL,
LogTime datetime NOT NULL,
MachineName nvarchar(128) NOT NULL,
FarmId uniqueidentifier NULL,
SiteSubscriptionId uniqueidentifier NULL,
UserLogin nvarchar(300) NULL,
CorrelationId uniqueidentifier NULL,
WebApplicationId uniqueidentifier NULL,
```

```
ServerUrl nvarchar(256) NULL,
SiteId uniqueidentifier NULL,
SiteUrl nvarchar(256) NULL,
WebId uniqueidentifier NULL,
WebUrl nvarchar(256) NULL,
DocumentPath nvarchar(256) NULL,
ContentTypeId nvarchar(1024) NULL,
QueryString nvarchar(512) NULL,
BytesConsumed int NULL,
HttpStatus smallint NULL,
SessionId nvarchar(64) NULL,
ReferrerUrl nvarchar(260) NULL,
ReferrerQueryString nvarchar(512) NULL,
Browser nvarchar(128) NULL,
UserAgent nvarchar(512) NULL,
UserAddress nvarchar(46) NULL,
RequestCount smallint NULL,
QueryCount smallint NULL,
QueryDurationSum bigint NULL,
ServiceCallCount bigint NULL,
ServiceCallDurationSum bigint NULL,
OperationCount bigint NULL,
Duration bigint NULL,
RequestType nvarchar(16) NULL,
Title nvarchar(128) NULL,
RowCreatedTime datetime NOT NULL,
```

PartitionId: The identifier of the partitioned table from which the row originates. This value MUST NOT be NULL or empty.

RowId: The unique identifier of the row. This value MUST NOT be NULL or empty.

LogTime: The UTC **timestamp** indicating when the request was initiated. This value MUST NOT be NULL or empty.

MachineName: The name of the computer from which the logging provider's data was written.

FarmId: The **farm identifier** of the farm from which the request originated.

SiteSubscriptionId: The **site subscription identifier** of the site (2) from which the request originated.

UserLogin: This value MUST be the login name for the user who initiated the request. If the login name is not available, this value MUST be the IP address for the client making the request. If both login name and IP address are unavailable, this value MUST be an empty string.

CorrelationId: The **request identifier** for the current request.

WebApplicationId: The **Web application identifier** for the request.

ServerUrl: The server URL for the request.

SiteId: The **site collection identifier** of the **site collection** for the request.

SiteUrl: The **relative path** of the URL of the site collection for the request.

WebId: The **site identifier** of the site (2) for the request.

WebUrl: The relative path of the URL of the site(2) for the request.

DocumentPath: The document path of the URL for the request.

ContentTypeId: The **content type identifier** for the content associated with the request.

QueryString: The **URI** query property for this request.

BytesConsumed: The total bytes of data downloaded as a result of this request.

HttpStatus: The **Status-Code** for this request.

SessionId: The browser **session identifier (2)** generating the request.

ReferrerUrl: The **URL** for the referring page for this request.

ReferrerQueryString: The URI query component (1) of the referring page for this request.

Browser: The client side browser name initiating the request.

UserAgent: The **user agent** value for the client side browser initiating the request.

UserAddress: The IP address of the client making the request.

RequestCount: The number of request objects created as a result of this request.

QueryCount: The number of back-end database queries generated as a result of this request.

QueryDurationSum: The time in milliseconds taken for all back-end database queries generated as a result of this request.

ServiceCallCount: The number of service calls generated as a result of this request.

ServiceCallDurationSum: The time in milliseconds taken for all service calls generated as a result of this request.

OperationCount: The value specified in this field MUST be ignored.

Duration: The time in milliseconds it took for the request to get executed.

RequestType: The HTTP request type for the client request.

Title: The title of the requested page.

RowCreatedTime: The UTC date when the request was initiated.

2.2.6 XML Structures

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

2.2.6.1 Namespaces

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

Prefix	Namespace URI	Reference
xs	http://www.w3.org/2001/XMLSchema	[XMLSCHEMA1] [XMLSCHEMA2]

2.2.6.2 Simple Types

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

Simple type	Description
GUIDType	A simple type that specifies a **GUID**.

2.2.6.2.1 GUIDType

A simple type used to reference a GUID.

```
<xs:simpleType name="GUIDType">
  <xs:restriction base="xs:string">
    <xs:pattern value="[A-Fa-f0-9]{8}-([A-Fa-f0-9]{4}-){3}[A-Fa-f0-9]{12}"/>
  </xs:restriction>
</xs:simpleType>
```

2.2.6.3 Complex Types

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

Complex type	Description
ContentSourcesType	This type specifies a list of all content sources from all search applications.
ContentSourceType	This type specifies a content source within a search application.

2.2.6.3.1 ContentSourcesType

This complex type specifies a list of all content sources from all search applications.

```
<xs:complexType name="ContentSourcesType">
  <xs:sequence>
    <xs:element name="ContentSource" minOccurs="0" maxOccurs="unbounded"
type="ContentSourceType"/>
  </xs:sequence>
</xs:complexType>
```

ContentSource: A ContentSourceType element that identifies a separate content source.

2.2.6.3.2 ContentSourceType

This complex type specifies a content source within a search application.

```
<xs:complexType name="ContentSourceType">
  <xs:attribute name="id" type="xs:int" />
  <xs:attribute name="appid" type="GUIDType" />
  <xs:attribute name="name" type="xs:string" />
</xs:complexType>
```

id: An int (as specified in [XMLSCHEMA2]) attribute that specifies an integer identifier of a content source.

appid: A GUIDType attribute that specifies a unique identifier of a search application.

name: A string (as specified in [XMLSCHEMA2]) attribute that specifies a name of a content source.

2.2.6.4 Elements

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

Element	Description
ContentSources	This element specifies a list of all content sources from all search applications.

2.2.6.4.1 ContentSources

This element specifies a list of all content sources from all search applications.

```
<xs:element name="ContentSources" type="ContentSourcesType"/>
```

ContentSources: A ContentSourcesType element that specifies a list of all content sources from all search applications.

2.2.6.5 Attributes

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

2.2.6.6 Groups

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

2.2.6.7 Attribute Groups

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

3 Protocol Details

This section provides detailed information about the protocol server and the protocol client.

3.1 Server Details

The back-end database protocol server responds to stored procedure calls and transact SQL queries. It returns result sets and return codes and never initiates communication with other endpoints of the protocol.

3.1.1 Abstract Data Model

This protocol provides a data store for the various protocol clients, also known as logging providers. Each logging provider defines the type definition for its own units of storage. A **logging provider type definition** consists of the following:

- The type name.
- Configuration settings:
  - Limit the number of days to store data (retention period).
  - Limit the max size in bytes to store.
- Default columns created by the system.
- Custom columns specified by the logging provider.
- Optional custom indexes.

When a logging provider is provisioned, type-specific views, tables, and stored procedures are generated within the database. This process is initiated by invocation of stored procedure **prc_CreateObjectsHelper**.

- A set of partitioned tables are created to store the data. The underlying table names are not constrained by the implementation.
- A single view is created for each type. The view name is identical to the logging provider's type name. In addition to the custom columns specified during initialization, a view will also contain the following:
  - **PartitionID:** A byte value representing the partitioned table identifier. This value indicates the physical table in which the underlying data in stored.
  - **RowId:** A GUID representing the unique identifier of the row.
  - **LogTime:** The UTC date and time indicating when the data was collected.
  - **MachineName:** The name of the computer from which the data was collected.
  - **RowCreatedTime:** The UTC date and time when the data was written to the database.
- Data insertion sprocs are created for utilization by the provider. These are of the form **prc_Insert<TypeDefinitionName>**, where **TypeDefinitionName** is the name of the logging provider type. All parameters are required. The parameters for this stored procedure include all of the custom column names followed by:

- **@MachineName:** The name of the computer from which the data was collected
- **@LogTime:** The UTC date time when the data was collected

- The data enumeration stored procedure is named **prc_Enum<TypeDefinitionName>**, where **TypeDefinitionName** is the name of the logging provider type. Rows are returned sorted by LogTime. All parameters are optional. The parameters for this stored procedure are as follows:
  - **@BeginTime:** The minimum UTC LogTime for which to retrieve data
  - **@EndTime:** The maximum UTC LogTime for which to retrieve data
  - **@MachineName:** The name of the computer from which the data was collected
  - **@RowsToReturn:** A 32-bit integer indicating the maximum number of rows to return.

Implementations of the protocol client can create custom objects such as stored procedures within this database. Custom stored procedures for the following logging provider types are referenced in this document:

- **BlockingQueries**
- **RequestUsage**
- **Search**

A protocol server also supports the following operations:

- **Provider Provisioning:** All database objects required by a logging provider type definition are provisioned by a call to **prc_CreateObjectsHelper**.
- **Provider Unprovisioning:** Database objects and metadata for a logging provider type definition can be deleted with a call to **prc_CleanObjectsHelper**.
- **Setting Retention Period:** The maximum retention period per logging provider can be set by calling **proc_AlterRetentionForType**.
- **Creating Custom Indexes:** Indexes can be created or dropped on the partitioned tables or views by calling **prc_EnsureIndexHelper**.
- **Returning Last Write Date:** A protocol client can discover the last time data was written for a type definition and computer name by calling **prc_GetLastUTCDate**.

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

This section provides information about message processing events and sequencing rules.

3.1.5.1 prc_CleanObjectsHelper

The **prc_CleanObjectsHelper** procedure is called to delete all database objects and metadata associated with a given logging provider. Objects deleted by prc_CleanObjectsHelper include all partitioned tables, **views** and **stored procedures** for the logging provider.

```
PROCEDURE prc_CleanObjectsHelper (
@TypeName nvarchar(100),
@Debug bit = 0,

);
```

@TypeName: The name of the logging provider to be deleted. This parameter MUST be specified and MUST identify an existing logging provider.

@Debug: Reserved. A bit flag that MUST be set to 0.

Error code values:

Value	Description
99901	An error occurred while deleting the objects associated with this particular provider.

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.2 prc_CreateObjectsHelper

The **CreateObjectsHelper** procedure is called to create all database objects for a given logging provider. Objects created include all partitioned tables, views and stored procedures for the logging provider. See section 3.1.1 for specifics on the objects created by this stored procedure.

```
PROCEDURE prc_CreateObjectsHelper (
@TypeName nvarchar(100),
@Columns nvarchar(3800),
@RetentionPeriod tinyint = 31,
@MaxTotalBytes bigint = 3100000000,
@Debug bit = 0,

);
```

@TypeName: The name of the logging provider that MUST be **provision**ed. This parameter MUST be specified and MUST NOT be NULL or empty.

@Columns: A comma-delimited list of SQL column definitions. This parameter MUST be specified and MUST contain definitions for all of the columns that MUST be provisioned with the partitioned tables for the logging provider.

@RetentionPeriod: The number of partitioned tables to create for this logging provider. This parameter MUST be specified and the value MUST be between 0 and 31.

@MaxTotalBytes: The total size of partitioned tables allowed for this logging provider. This parameter MUST be set to a value greater than zero to enable logging for this provider.

@Debug: Reserved. A bit flag that MUST be set to 0.

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.3 prc_EnsureIndexHelper

The **prc_EnsureIndexHelper** stored procedure is called to create or drop a single SQL index on a view. (A SQL index is a data structure used by SQL to enable it to quickly find table or view records based on the values of one or more columns.)

```
PROCEDURE prc_EnsureIndexHelper (
@TypeName nvarchar(100),
@IndexName sysname,
@Columns nvarchar(3800),
@Drop bit = 0,
@Debug bit = 0,

);
```

@TypeName: The name of the logging provider or view. This parameter MUST be specified and MUST identify a valid logging provider or view name.

@IndexName: The name of the SQL index to create or drop. This parameter MUST be specified.

@Columns: The comma-delimited list of columns in the given logging provider for which an index MUST be created or dropped. Each column name may be followed by a space and "ASC" or "DESC", indicating whether that column should be indexed in ascending or descending order, respectively. This parameter MUST be specified and MUST identify all column names for the logging provider for which an SQL index MUST be created or MUST be dropped.

@Drop: A bit value that indicates whether to drop or create a SQL index. The value of the parameter MUST be listed in the following table.

Value	Description
0	SQL Index MUST be created.
1	SQL Index MUST be dropped.

@Debug: Reserved. A bit value that MUST be set to 0.

Return Values: An integer that MUST be 0.

Result Sets:

The result set returned by this stored procedure, prc_EnsureIndexHelper.ResultSet0, MUST be ignored by the client.

3.1.5.4 prc_GetLastUTCDate

The **prc_GetLastUTCDate** stored procedure is called to obtain the most recent UTC date for the last data written from the specified computer for a given logging provider.

```
PROCEDURE prc_GetLastUTCDate (
@TypeName nvarchar(100),
@MachineName nvarchar(128),
@MinUTCDate datetime = null,
@Debug bit = 0,

);
```

@TypeName: The name of the logging provider from which to obtain the most recent UTC date. This parameter MUST be specified and MUST identify a logging provider.

@MachineName: The name of the computer used to query the most recent UTC date from which the logging provider's data was written.

@MinUTCDate: The smallest UTC date to return. If this parameter is NULL, then the smallest UTC date to return MUST be equal to the current UTC date subtracted by a number of days equal to the logging provider's retention period.

This value is returned if no data has been previously written for the logging provider from the specified computer, or if the last UTC date on which data was written is smaller.

@Debug: Reserved. A bit flag that MUST be set to 0.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a prc_GetLastUTCDate.prc_GetLastUTCDate.Default.ResultSet0

3.1.5.5 proc_AlterRetentionForType

The **proc_AlterRetentionForType** stored procedure is called to specify the retention period, in days, for a given logging provider.

```
PROCEDURE proc_AlterRetentionForType (
@TypeName nvarchar(100),
@RetentionPeriod tinyint = 31,
@Debug bit = 0,

);
```

@TypeName: The name of the logging provider for which to specify a retention period. This parameter MUST be specified and MUST identify a logging provider.

@RetentionPeriod: An integer representing the new retention period in days for the specified logging provider. This parameter MUST be specified and MUST be a value greater than or equal to 0 and less than or equal to 31.

@Debug: Reserved. A bit flag that MUST be set to 0.

Return Values: An integer that MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5.6 proc_GetMostActiveUsers

The **proc_GetMostActiveUsers** stored procedure is called to obtain the users that performed the most HTTP requests in a given time span.

```
PROCEDURE proc_GetMostActiveUsers (
@StartTime datetime = null,
@EndTime datetime = null,
@WebApplicationId uniqueidentifier = null,
@MachineName nchar(128) = null,
@MaxRows bigint = 100,

);
```

@StartTime: The starting UTC date of the time span for which user activity was measured. @StartTime MUST NOT be null.

@EndTime: The ending UTC date of the time span for which user activity was measured. **@EndTime** MUST NOT be null and MUST NOT be less than **@StartTime**.

@WebApplicationId: The unique identifier of the **Web application (1)** from which the user request was originated. **@WebApplicationID** MUST be the identifier of an existing Web application (1) or null. If **@WebApplicationId** is null, the result set MUST return data originated by all Web applications (1). Otherwise, **@WebApplicationId** MUST be the identifier of an existing Web application (1) and the result set values MUST correspond to user activity statistics originated by the specified Web application (1).

@MachineName: The name of the computer from which the **RequestUsage** logging provider's data was written. **@MachineName** MUST be the name of an existing computer or null. If **@MachineName** is null, the result set MUST correspond to user activity statistics originated by all computers. Otherwise, **@MachineName** MUST be the name of an existing computer and the result set values MUST correspond to user activity statistics originated by the specified computer.

@MaxRows: The maximum number of rows to return. This value MUST be equal or greater than zero and the result set MUST NOT contain a number of rows greater than this value.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a proc_GetMostActiveUsers.ResultSet0 result set.

3.1.5.7 proc_GetSlowestPages

The **proc_GetSlowestPages** stored procedure is called to obtain the pages with highest server-side page latency within a given time span.

```
PROCEDURE proc_GetSlowestPages (
@StartTime datetime = null,
@EndTime datetime = null,
@WebApplicationId uniqueidentifier = null,
@MachineName nchar(128) = null,
@MaxRows bigint = 100,
```

);

@StartTime: The starting UTC date of the time span for which the page activity was measured.

@EndTime: The ending UTC date of the time span for which the page activity was measured. **@EndTime** MUST be NULL or greater than **@StartTime**.

@WebApplicationId: The unique identifier of the Web application (1) from which the user request was originated. **@WebApplicationId** MUST be the identifier of an existing Web application (1) or NULL. If **@WebApplicationId** is NULL, the result set MUST return data originated by all Web applications (1). Otherwise, **@WebApplicationId** MUST be the identifier of an existing Web application (1) and the result set values MUST correspond to user activity statistics originated by the specified Web application (1).

@MachineName: The name of the computer from which the RequestUsage logging provider's data was written. **@MachineName** MUST be the name of an existing computer or null. If @MachineName is NULL, the result set MUST correspond to user activity statistics originated by all computers. Otherwise, **@MachineName** MUST be the name of an existing computer and the result set values MUST correspond to user activity statistics originated by the specified computer.

@MaxRows: The maximum number of rows to return. This value MUST be equal or greater than zero and the result set MUST NOT contain a number of rows greater than this value.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a proc_GetSlowestPages.ResultSet0 result set.

3.1.5.8 Search_GetCrawlProcessingPerActivity

The **Search_GetCrawlProcessingPerActivity** stored procedure is called to retrieve the time spent on different activities during crawling. The procedure MUST return the time in milliseconds per activity for each minute in a given time span. If no items were crawled during a given minute then the procedure MUST NOT return an entry for that minute. The procedure MUST return data for either the specified search application or the total data among all search applications as specified by the **@applicationId** parameter. The stored procedure MUST return zero rows in the result set if the search application with the specified identifier does not exist or **@endDate** is earlier than **@startDate**.

```
PROCEDURE Search_GetCrawlProcessingPerActivity (
@applicationId uniqueidentifier,
@startDate datetime,
@endDate datetime,
```

);

@applicationId: The unique identifier of the search application for which the stored procedure MUST return data. If the value of the **@applicationId** parameter is "00000000-0000-0000-0000-000000000000", the stored procedure MUST return the total time among all search applications. This parameter MUST NOT be NULL.

@startDate: The starting UTC time of the time span for which data on crawling activities is returned. This parameter MUST NOT be NULL.

@endDate: The ending UTC time of the time span for which data on crawling activities is returned. This parameter MUST NOT be NULL.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Crawl Processing Per Activity Result Set

3.1.5.9 Search_GetCrawlProcessingPerComponent

The **Search_GetCrawlProcessingPerComponent** stored procedure is called to retrieve the time spent by different components of the **crawler**. The procedure MUST return the time in milliseconds per component for each minute in a given time span. If no items were crawled at some particular minute then procedure MUST NOT return an entry for this minute. The procedure MUST return data for either specified search application or total data among all search applications as specified by the **@applicationId** parameter. The stored procedure MUST return zero rows in the result set if search application with specified identifier does not exist or **@endDate** is earlier than **@startDate**.

```
PROCEDURE Search_GetCrawlProcessingPerComponent (
@applicationId uniqueidentifier,
@startDate datetime,
@endDate datetime,

);
```

@applicationId: The unique identifier of the search application for which the stored procedure MUST return data in the result set. If the value of **@applicationId** parameter is "00000000-0000-0000-0000-000000000000", the stored procedure MUST return total time among all search applications. This parameter MUST NOT be NULL.

@startDate: The starting UTC time of the time span for which data on crawler components to be returned. This parameter MUST NOT be NULL.

@endDate: The ending UTC time of the time span for which data on crawler components to be returned. This parameter MUST NOT be NULL.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Crawl Processing Per Component Result Set

3.1.5.10 Search_GetCrawlProcessingStagePerItem

The **Search_GetCrawlProcessingStagePerItem**<1> stored procedure is called to retrieve the average time spent during different stages of crawling per item. The procedure MUST return the average time in milliseconds per stage for each minute in a given time span. If no items were crawled during a particular minute then the procedure MUST NOT return an entry for this minute. The procedure MUST return data for either specified search application or total data among all search applications as specified by the **@applicationId** parameter. The stored procedure MUST return zero rows in the result set if search application with specified identifier does not exist or **@endDate** is earlier than **@startDate**.

```
PROCEDURE Search_GetCrawlProcessingStagePerItem (
```

```
  @applicationId uniqueidentifier,
  @startDate datetime,
  @endDate datetime,

);
```

@applicationId: The unique identifier of the search application for which the stored procedure MUST return data in the result set. If the value of the **@applicationId** parameter is "00000000-0000-0000-0000-000000000000", the stored procedure MUST return average time among all search applications. This parameter MUST NOT be NULL.

@startDate: The starting UTC time of the time span for which data on crawling stages to be returned. This parameter MUST NOT be NULL.

@endDate: The ending UTC time of the time span for which data on crawling stages to be returned. This parameter MUST NOT be NULL.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Crawl Processing Stage Per Item Result Set

3.1.5.11 Search_GetCrawlQueue

The **Search_GetCrawlQueue** stored procedure is called to retrieve the number of items in the queue of links to be crawled and the number of items in the queue of transactions (1) during the given time frame. The procedure MUST return data for each minute in a given time span. The procedure MUST return data for either specified search application or total data among all search applications as specified by the **@applicationId** parameter. The stored procedure MUST return zero rows in the result set if search application with specified identifier does not exist or **@endDate** is earlier than **@startDate**.

```
  PROCEDURE Search_GetCrawlQueue (
  @applicationId uniqueidentifier,
  @startDate datetime,
  @endDate datetime,

);
```

@applicationId: The unique identifier of the search application for which the stored procedure MUST return data in the result set. If the value of the **@applicationId** parameter is "00000000-0000-0000-0000-000000000000", the stored procedure MUST return total data for all search applications. This parameter MUST NOT be NULL.

@startDate: The starting UTC time of the time span for which crawl queue data to be returned. This parameter MUST NOT be NULL.

@endDate: The ending UTC time of the time span for which crawl queue data to be returned. This parameter MUST NOT be NULL.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Crawl Queue Result Set

3.1.5.12 Search_GetCrawlRatePerContentSource

The **Search_GetCrawlRatePerContentSource** stored procedure is called to retrieve the number of items crawled from each type of content sources. The procedure MUST return crawl rates by each type of content source for each minute within a given time span. If no items were crawled from some particular content source at some minute then the procedure MUST NOT return an entry for this minute for this content source. The procedure MUST return data for either specified search application or total data among all search applications as specified by the **@applicationId** parameter. The stored procedure MUST return zero rows in the result set if search application with specified identifier does not exist or **@endDate** is earlier than **@startDate**.

```
PROCEDURE Search_GetCrawlRatePerContentSource (
@contentSources xml,
@applicationId uniqueidentifier,
@startDate datetime,
@endDate datetime,

);
```

@contentSources: An **xml** data type describing all available content sources from all search applications. This parameter MUST be a valid ContentSources element of the **xml** structure.

@applicationId: The unique identifier of the search application for which the stored procedure MUST return crawl rates in the result set. If the value of the **@applicationId** parameter is "00000000-0000-0000-0000-000000000000", the stored procedure MUST return total data for all search applications. This parameter MUST NOT be NULL.

@startDate: The starting UTC time of the time span for which crawl rate to be returned. This parameter MUST NOT be NULL.

@endDate: The ending UTC time of the time span for which crawl rate to be returned. This parameter MUST NOT be NULL.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Crawl Rate Per Content Source Result Set

3.1.5.13 Search_GetCrawlRatePerType

The **Search_GetCrawlRatePerType** stored procedure is called to retrieve the counts of the specific crawl actions performed for the items that were encountered during the crawl. The procedure MUST return data for each minute within a given time span. If no items were crawled at some particular minute then the procedure MUST NOT return an entry for this minute. The procedure MUST return data for either specified search application or total data among all search applications as specified by the **@applicationId** parameter. The stored procedure MUST return zero rows in the result set if search application with specified identifier does not exist or **@endDate** is earlier than **@startDate**.

```
PROCEDURE Search_GetCrawlRatePerType (
@applicationId uniqueidentifier,
@startDate datetime,
```

```
@endDate datetime,
```

);

@applicationId: The unique identifier of the search application for which the stored procedure MUST return data in the result set. If the value of the **@applicationId** parameter is "00000000-0000-0000-0000-000000000000", the stored procedure MUST return total data for all search applications. This parameter MUST NOT be NULL.

@startDate: The starting UTC time of the time span for which numbers of processed items to be returned. This parameter MUST NOT be NULL.

@endDate: The ending UTC time of the time span for which numbers of processed items to be returned. This parameter MUST NOT be NULL.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Crawl Rate Per Type Result Set

3.1.5.14 Search_GetQueryLatency

The **Search_GetQueryLatency** stored procedure is called to retrieve the average time of search queries processing on different stages. The procedure MUST return the average time spent per stage for each minute in a given time span. If no search queries were processed at any stage at some particular minute then the procedure MUST NOT return an entry for this minute. The procedure MUST return data for either specified search application or average data among all search applications as specified by the **@applicationId** parameter. The stored procedure MUST return zero rows in the result set if search application with specified identifier does not exist or **@endDate** is earlier than **@startDate**.

```
PROCEDURE Search_GetQueryLatency (
@applicationId uniqueidentifier,
@startDate datetime,
@endDate datetime,
```

);

@applicationId: The unique identifier of the search application for which the stored procedure MUST return times of search queries processing in the result set. If the value of the **@applicationId** parameter is "00000000-0000-0000-0000-000000000000", the stored procedure MUST return average data for all search applications. This parameter MUST NOT be NULL.

@startDate: The starting UTC time of the time span for which times of search queries processing to be returned. This parameter MUST NOT be NULL.

@endDate: The ending UTC time of the time span for which times of search queries processing to be returned. This parameter MUST NOT be NULL.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Query Latency Result Set

3.1.5.15 Search_GetRecentStats

The **Search_GetRecentStats** stored procedure is called to retrieve the number of processed search queries and the number of crawled items within the last 5 minutes for the specified search application.

```
PROCEDURE Search_GetRecentStats (
@applicationId uniqueidentifier,

);
```

@applicationId: The unique identifier of the search application.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Recent Query Stat Result Set

This stored procedure MUST return a Recent Crawl Stat Result Set

3.1.5.16 Search_GetCrawlRatePerContentSourceSummary

The **Search_GetCrawlRatePerContentSourceSummary** stored procedure is called to retrieve the total number of items crawled from each type of content sources separately for each crawl. The procedure MUST return crawl rates within a given time span. The procedure MUST return data for either specified search application or total data among all search applications as specified by the **@applicationId** parameter. The stored procedure MUST return zero rows in the result set if search application with specified identifier does not exist or if **@endDate** is earlier than **@startDate**.

```
PROCEDURE Search_GetCrawlRatePerContentSourceSummary (
@contentSources xml,
@applicationId uniqueidentifier,
@startDate datetime,
@endDate datetime,

);
```

@contentSources: An **xml** data type describing all available content sources from all search applications. This parameter MUST be a valid ContentSources element of the **xml** structure.

@applicationId: The unique identifier of the search application for which the stored procedure MUST return crawl rates in the result set. If the value of the **@applicationId** parameter is "00000000-0000-0000-0000-000000000000", the stored procedure MUST return data for all search applications. This parameter MUST NOT be NULL.

@startDate: The starting UTC time of the time span for which crawl rate to be returned. This parameter MUST NOT be NULL.

@endDate: The ending UTC time of the time span for which crawl rate to be returned. This parameter MUST NOT be NULL.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Search_GetCrawlRatePerContentSourceSummary.ResultSet0

3.1.5.17 Search_GetCrawlRatePerTypeSummary

The **Search_GetCrawlRatePerTypeSummary** stored procedure is called to retrieve the total number of the specific crawl actions performed for the items that were encountered during the crawl within a given time span. The procedure MUST return data for either specified search application or total data among all search applications as specified by the **@applicationId** parameter. The stored procedure MUST return zero rows in the result set if search application with specified identifier does not exist or **@endDate** is earlier than **@startDate**.

```
PROCEDURE Search_GetCrawlRatePerTypeSummary (
@applicationId uniqueidentifier,
@startDate datetime,
@endDate datetime,

);
```

@applicationId: The unique identifier of the search application for which the stored procedure MUST return crawl rates in the result set. If the value of the **@applicationId** parameter is "00000000-0000-0000-0000-000000000000", the stored procedure MUST return total information for all search applications. This parameter MUST NOT be NULL.

@startDate: The starting UTC time of the time span for which the number of processed items to be returned. This parameter MUST NOT be NULL.

@endDate: The ending UTC time of the time span for which the number of processed items to be returned. This parameter MUST NOT be NULL.

Return Values: An integer that MUST be 0.

Result Sets:

This stored procedure MUST return a Search_GetCrawlRatePerTypeSummary.ResultSet0

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

3.2 Client Details

None.

3.2.1 Abstract Data Model

None.

3.2.2 Timers

None.

3.2.3 Initialization

Initialization of a logging provider is performed by invoking the stored procedure described in section 3.1.5.2.

3.2.4 Higher-Layer Triggered Events

None.

3.2.5 Message Processing Events and Sequencing Rules

None.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

4 Protocol Examples

This section provides specific example scenarios for generating reports from the usage data, adding and deleting a new usage provider, inserting data for a usage provider, generating a report for a new usage provider, and configuring the retention period.

4.1 Generating a Report from the Usage Data

The following examples show how to create reports from the usage data.

4.1.1 Generating a Report of the Top Slowest Pages

To generate a report of the pages with higher average duration during the last one day for all user requests for **WebApplicationId** 1427092D-3B0E-4DCD-AF80-79847A18BC20 and **MachineName** TestMachine, consider the following T-SQL syntax used by the protocol client to call the **proc_GetSlowestPages.**

```
declare @stime datetime
declare @etime datetime
set @stime = getDate() - 1
set @etime = getDate()
exec dbo.proc_GetSlowestPages
@StartTime = @stime,
@EndTime = @etime,
@WebApplicationId = '1427092D-3B0E-4DCD-AF80-79847A18BC20',
@MachineName = 'TestMachine'
```

The protocol server responds with a result set containing information about the slowest pages based on the preceding query. Consider the following result set which could be returned by the protocol server.

Url	AverageDuration	Maximum Duration	Minimum Duration	Average Query Count	MaximumQuery Count	MinimumQuery Count	Total Page Hits
http://server.example.com/	0.118	0.177	0.085	0	0	0	3
http://server.example.com/my	2.149	6.054	0.031	0	0	0	3

4.1.2 Generating a Report of the Most Active Users

To generate a report of the users that performed most requests during the last one day for **WebApplicationId** 1427092D-3B0E-4DCD-AF80-79847A18BC20 and **MachineName** TestMachine, consider the following T-SQL syntax used by the protocol client to call the **proc_GetMostActiveUsers**.

```
declare @stime datetime
declare @etime datetime
set @stime = getDate() - 1
set @etime = getDate()
exec dbo.proc_GetMostActiveUsers
@StartTime = @stime,
```

```
@EndTime = @etime,
@WebApplicationId = '1427092D-3B0E-4DCD-AF80-79847A18BC20',
@MachineName = 'TestMachine'
```

The protocol server responds with a result set containing information specified by the preceding call. Consider the following result set that could be returned by the protocol server.

User	Hits	LastAccessTime	SuccessRate
0#.w|domain\serviceaccount	9288	2010-01-19 09:27:32.240	1
domain\user1	11	2010-01-18 23:49:30.037	0.909090909090909
domain\user2	7	2010-01-19 00:18:21.107	1
domain\user3	3	2010-01-18 21:13:17.877	1

4.1.3 Generating a Report from the RequestUsage View

To generate a report of selected request fields, such as userlogin, serverurl, weburl, documentpath, browser, bytesconsumed, httpstatus, useragent, for all user requests for **WebApplicationId** 1427092D-3B0E-4DCD-AF80-79847A18BC20 yesterday, consider the following T-SQL syntax used by the protocol client to query the **RequestUsage** view.

```
declare @stime datetime
declare @etime datetime
set @stime = getdate() - 2
set @etime = getdate() - 1
create table #partitions (partitionid tinyint)
insert into #partitions (partitionid)
select partitionid from dbo.fn_partitionidrangemonthly(@stime, @etime)
select userlogin, serverurl, weburl, documentpath, browser, bytesconsumed, httpstatus,
useragent
from requestusage as t with (readpast)
inner join #partitions as p
on t.partitionid = p.partitionid
where webapplicationid = '1427092d-3b0e-4dcd-af80-79847a18bc20'
and ([logtime] between @stime and @etime)
drop table #partitions
```

Consider the following result set that could be returned by the protocol server.

user login	serverurl	weburl	documentpath	browser	Bytesconsumed	httpstatus	useragent
0#.w|domain\serviceaccount	http://server.domain.com:32843		/5e1c8c6b9ed2406ab7d3355b8374f481/ProfilePropertyService.svc		0	0	
domain\user1	http://server.domain.com:50000		/_admin/adminconfigservicesresults.aspx	IE7	0	302	Mozilla/4.0 (com

user login	serverurl	weburl	documentpath	browser	Bytes consumed	http status	useragent
							patible; MSIE 7.0; Windows NT 6.0; WOW64; SLCC1; .NET CLR 2.0.50727; .NET CLR 3.0.30729; .NET CLR 3.5.30729)

4.1.4 Generating a Report from the BlockingQueries View

To generate a report of the top ten blocking queries ordered by the waiting time, in the last one day, consider the following T-SQL syntax used by the protocol client to query the **BlockingQueries** view.

```
declare @stime datetime
declare @etime datetime
set @stime = getdate() - 1
set @etime = getdate()
create table #partitions (partitionid tinyint)
insert into #partitions (partitionid)
select partitionid from dbo.fn_partitionidrangemonthly(@stime, @etime)
select top 10
logtime, last_execution_time, blocking_machine, blocking_database_name, waiting_time,
waiting_sid, blocking_sid, blocking_blocker_sid, blocking_process_id, waiting_resource,
waiting_type, blocking_statement, waiting_statement, blocking_query_text, waiting_query_text
from blockingqueries as t with (readpast)
inner join #partitions as p
on t.partitionid = p.partitionid
where logtime between @stime and @etime
and blocking_blocker_sid = 0
order by waiting_time desc
drop table #partitions
```

4.2 Configuring the Retention Period

To specify the retention period of thirty days for the requestusage provider, consider the following T-SQL syntax used by the protocol client to call the **proc_AlterRetentionForType.**

```
exec proc_AlterRetentionForType
@TypeName = 'RequestUsage',
@RetentionPeriod = 30
```

The protocol server changes the retention period and returns a value of 0, which is ignored by the protocol client.

4.3 Adding a New Usage Provider

To add a new usage provider, the protocol client would call the **prc_CreateObjectsHelper** stored procedure. Consider the following T-SQL syntax used by the protocol client, to add a new usage provider with definition 'MyUsageProvider'. The provider defines four columns for its schema, namely EventTime whose type is datetime, Severity whose type is tinyint, Source whose type is nvarchar(255) and MessageText whose type is nvarchar(4000). The provider also specifies the retention period of twenty days and the maximum total bytes of one million.

```
exec prc_CreateObjectsHelper
@TypeName = 'MyUsageProvider',
@Columns = 'EventTime datetime , Severity tinyint , Source nvarchar(255) , MessageText
nvarchar(4000)',
@RetentionPeriod = 20,
@MaxTotalBytes = 1000000
```

This call will auto-provision all database objects for the new usage provider including the partitioned tables, partitioned view and a couple of stored procedures, namely, **prc_EnumMyUsageProvider** and **prc_InsertMyUsageProvider.** The protocol server will return a value of 0, which is ignored, and no result set.

4.4 Inserting Data for a Usage Provider

To insert data for a usage provider, the protocol client would call the **prc_Insert<DefinitionName>** stored procedure, that has been auto-provisioned when the protocol client calls the **prc_CreateObjectsHelper** stored procedure, passing **DefinitionName** for the @TypeName parameter. Consider the following T-SQL syntax used by the protocol client, to insert a row of data for the 'MyUsageProvider' usage definition.

```
declare @logtime datetime
set @logtime = getDate()
exec prc_InsertMyUsageProvider
@MachineName = 'TestMachine',
@LogTime = @logtime,
@EventTime = '2009-03-27 17:37:45.850',
@Severity = 1,
@Source = 'MySource',
@MessageText = 'This is an example'
```

The protocol server inserts the given values as specified, returns a value of 0, which is ignored and returns no result set.

4.5 Generating a Report for a New Usage Provider

To generate a report for a new usage provider, the protocol client would query the **<DefinitionName>** view, that has been auto-provisioned when the protocol client calls the **prc_CreateObjectsHelper** stored procedure, passing **DefinitionName** for the @TypeName parameter. Consider the following T-SQL syntax used by the protocol client to query the top ten rows, in the last one day, for the 'MyUsageProvider' usage definition.

```
declare @stime datetime
declare @etime datetime
set @stime = getdate() - 1
set @etime = getdate()
create table #partitions (partitionid tinyint)
insert into #partitions (partitionid)
select partitionid from dbo.fn_partitionidrangemonthly(@stime, @etime)
select top 10
logtime, machinename, eventtime, severity, source, messagetext
from myusageprovider as t with (readpast)
inner join #partitions as p
on t.partitionid = p.partitionid
where logtime between @stime and @etime
```

Assuming that example 4.4 was executed, the following result set will be returned by the protocol server.

logtime	machinename	eventtime	severity	source	messagetext
2010-01-19 10:46:24.747	TestMachine	2009-03-27 17:37:45.850	1	MySource	This is an example

4.6 Deleting a Usage Provider

To delete a usage provider, the protocol client would call the **prc_CleanObjectsHelper** stored procedure. Consider the following T-SQL syntax used by the protocol client, to delete the usage provider with definition 'MyUsageProvider'.

```
exec prc_CleanObjectsHelper
@TypeName = 'MyUsageProvider'
```

The protocol server deletes the given usage provider, returns a value of 0, which is ignored and does not return any result set.

5 Security

5.1 Security Considerations for Implementers

None.

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® SharePoint® Foundation 2010

Exceptions, if any, are noted below. If a service pack or Quick Fix Engineering (QFE) number appears with the product version, behavior changed in that service pack or QFE. The new behavior also applies to subsequent service packs of the product unless otherwise specified. If a product edition appears with the product version, behavior is different in that product edition.

Unless otherwise specified, any statement of optional behavior in this specification that is prescribed using the terms SHOULD or SHOULD NOT implies product behavior in accordance with the SHOULD or SHOULD NOT prescription. Unless otherwise specified, the term MAY implies that the product does not follow the prescription.

<1> Section 3.1.5.10: The **Search_GetCrawlProcessingStagePerItem** stored procedure has been removed from the product in the RTM version.

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

S

T

V

X