[MS-UTSP2]:
SharePoint Usage Tracking Stored Procedures Version 2 Protocol Specification

Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.
- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL's, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.
- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.
- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft's delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.
- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.
- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

Preliminary Documentation. This Open Specification provides documentation for past and current releases and/or for the pre-release (beta) version of this technology. This Open Specification is final documentation for past or current releases as specifically noted in the document, as applicable; it is preliminary documentation for the pre-release (beta) versions. Microsoft will release final documentation in connection with the commercial release of the updated or new version of this technology. As the documentation may change between this preliminary version and the final version of this technology, there are risks in relying on preliminary documentation. To the extent that you incur additional development obligations or any other costs as a result of relying on this preliminary documentation, you do so at your own risk.

Revision Summary

Date	Revision History	Revision Class	Comments
01/20/2012	0.1	New	Released new document.
04/11/2012	0.1	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	0.1	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.

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]:

Coordinated Universal Time (UTC)
correlation
GUID
user agent

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

application server
back-end database server
content source
content type identifier
crawl
crawl component
farm
farm identifier
front-end Web server
full-text index catalog
item
logging provider
login name
metadata store
partitioned table
provisioned
query
query component
relative path
request identifier
result set
retention period
return code
role
role assignment
search application
search query
session identifier
site
site collection
site collection identifier
site identifier
site subscription identifier
Status-Code

stored procedure
timer job
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:

diagnostics data: The information collected for a monitored scope.

monitored scope: A section of program code about which data is collected for diagnostic purposes as the code is executed.

shared service application: A logical grouping of service components that can be consumed by remote server farms.

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

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

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

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

1.3 Overview

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:

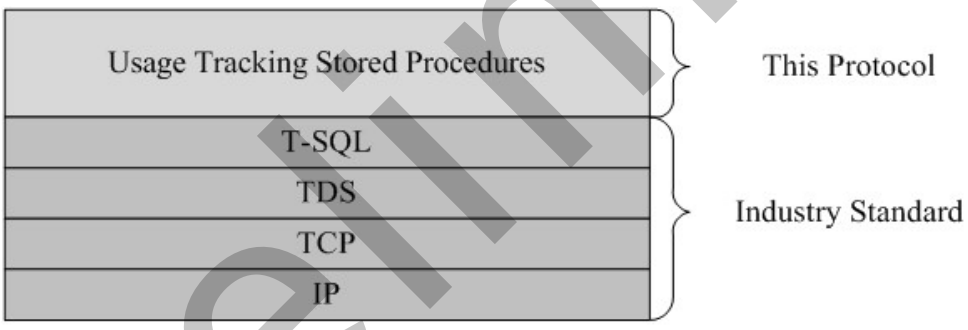

Figure 1: This protocol in relation to other protocols

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

No common simple data types or enumerations are defined in this protocol.

2.2.2 Bit Fields and Flag Structures

No common bit field or flag structures are defined in this protocol.

2.2.3 Binary Structures

No common binary structures are defined in this protocol.

2.2.4 Result Sets

This protocol specifies the following result sets.

2.2.4.1 Crawl Processing Per Activity Result Set

The **Crawl Processing Per Activity** result set returns the time spent on different crawling activities per 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 the crawl component plug-ins to register an item when 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.

2.2.4.2 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 the data, and pass 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)**.

2.2.4.3 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.

2.2.4.4 Recent Crawl Stat Result Set

The **Recent Crawl Stat** result set returns information about the number of crawled items 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 crawled items within the last 5 minutes.

2.2.4.5 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.

2.2.4.6 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.

2.2.4.7 prc_CreateObjectsHelper.ResultSet0

Reserved. This result set MUST be empty.

```
UnnamedColumn0 int,
```

UnnamedColumn0: This column name is not used and is left blank.

2.2.4.8 prc_CreateObjectsHelper.ResultSet1

Reserved. This result set MUST be empty.

```
UnnamedColumn0 int,
```

UnnamedColumn0: This column name is not used and is left blank.

2.2.4.9 prc_CreateObjectsHelper.ResultSet2

Reserved. This result set MUST be empty.

```
UnnamedColumn0 int,
```

UnnamedColumn0: This column name is not used and is left blank.

2.2.4.10 prc_CreateObjectsHelper.ResultSet3

Reserved. This result set MUST be empty.

```
UnnamedColumn0 int,
```

UnnamedColumn0: This column name is not used and is left blank.

2.2.4.11 proc_GetDiagnosticsData.ResultSet0

The **proc_GetDiagnoticsData.ResultSet0** result set returns diagnostics data. This result set MUST be returned and the rows MUST be arranged in ascending order of log time.

```
CorrelationId uniqueidentifier NULL,
ScopeId bigint NULL,
Name nvarchar(256) NULL,
ValueInt bigint NULL,
ValueFloat float NULL,
ValueString nvarchar(512) NULL,
ValueBinary varbinary(max) NULL,
LogTime datetime NOT NULL,
```

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

ScopeId: The identifier of the **diagnostics data** scope within the current request.

Name: The name of the diagnostics data.

ValueInt: The integer value of the diagnostics data.

ValueFloat: The floating point value of the diagnostics data.

ValueString: The string of the diagnostics data.

ValueBinary: The binary value of the diagnostics data.

LogTime: The **timestamp** when the current diagnostics data row is logged.

2.2.4.12 proc_GetCorrelationIdAndUsers.ResultSet0

The **proc_GetCorrelationIdAndUsers.ResultSet0** result set returns **monitored scope** requests. This result set MUST be returned and the rows MUST be arranged in ascending order of log time.

```
CorrelationId uniqueidentifier NULL,
Name nvarchar(256) NULL,
ValueString nvarchar(512) NULL,
```

CorrelationId: The request identifier of the request.

Name: This value MUST be 'CurrentUser'

ValueString: The string of the diagnostics data.

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(256) NOT 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 nvarchar(max) NULL,
Waiting_Statement nvarchar(max) NULL,
Blocking_Query_Text nvarchar(max) NULL,
Waiting_Query_Text nvarchar(max) NULL,
Blocking_Query_Plan nvarchar(max) NULL,
Waiting_Query_Plan 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: The zero-based offset of the beginning of the blocking statement in **Blocking_Query_Text**.

Waiting_Statement: The zero-based offset of the beginning 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.

Blocking_Query_Plan: The execution plan of the blocking SQL query. MUST be NULL for encrypted objects.

Waiting_Query_Plan: The execution plan 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 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,
OperationCount bigint NULL,
Duration bigint NULL,
RequestType nvarchar(16) NULL,
Title nvarchar(128) NULL,
RowCreatedTime datetime NOT NULL,
QueryDurationSum bigint NULL,
ServiceCallCount smallint NULL,
ServiceCallDurationSum bigint NULL,
SqlLogicalReads bigint NULL,
CPUMCycles bigint NULL,
```

```
DistributedCacheReads bigint NULL,
DistributedCacheReadsDuration bigint NULL,
DistributedCacheReadsSize bigint NULL,
DistributedCacheWrites bigint NULL,
DistributedCacheWritesDuration bigint NULL,
DistributedCacheWritesSize bigint NULL,
DistributedCacheMisses bigint NULL,
DistributedCacheHits bigint NULL,
ManagedMemoryBytes bigint NULL,
ManagedMemoryBytesLOH bigint NULL,
IisLatency bigint 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.

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

Duration: The time in milliseconds it took for the front-end Web server to process the request.

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.

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.

SqlLogicalReads: The number of 8 kilobyte blocks read from storage on the back-end database server as a result of this request.

CPUMCycles: The number of CPU megacycles spent processing the request in the client application on the front-end Web server.

DistributedCacheReads: Total number of read requests to the distributed cache service.

DistributedCacheReadsDuration: Total time spent, in milliseconds, during read requests to the distributed cache service.

DistributedCacheReadsSize: Total data, in bytes read from the distributed cache service.

DistributedCacheWrites: Total number of write requests to the distributed cache service.

DistributedCacheWritesDuration: Total time spent, in milliseconds, during write requests to the distributed cache service.

DistributedCacheWritesSize: Total data, in bytes, written in the distributed cache service.

DistributedCacheMisses: Total cache misses on the distributed cache service.

DistributedCacheHits: Total successful cache hits on the distributed cache service.

ManagedMemoryBytes: The number of bytes allocated in managed memory heaps in the client application on the front-end Web server as a result of this request.

ManagedMemoryBytesLOH: The number of bytes allocated in the large object managed memory heap on the front-end Web server as a result of this request.

IisLatency: The time in milliseconds taken in the front-end Web server after the request has been received the front-end Web server, but before the **Web application (2)** begins processing the request.

2.2.5.4 Configuration

This table is provided to store customizable settings such as the total size and number of partitioned tables for each logging provider.

```
ConfigName nvarchar(255) NOT NULL,
ConfigValue nvarchar(255) NOT NULL,
```

ConfigName: A name for the setting. This value MUST NOT be NULL or empty.

ConfigValue: The value of the setting. This value MUST NOT be NULL.

2.2.5.5 MonitoredScopeDiagnosticsData

This table stores diagnostics data.

```
PartitionId tinyint NULL,
RowId uniqueidentifier NOT NULL,
LogTime datetime NOT NULL,
MachineName nvarchar(128) NOT NULL,
CorrelationId uniqueidentifier NULL,
ScopeId bigint NULL,
Name nvarchar(256) NULL,
Flag bigint NULL,
ValueInt bigint NULL,
ValueFloat float NULL,
ValueString nvarchar(512) NULL,
ValueBinary varbinary(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 timestamp when the current diagnostics data row is logged.

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

CorrelationId: The request identifier of the current request.

ScopeId: The identifier of the diagnostics data scope within the current request.

Name: The name of the diagnostics data.

Flag: An integer bit field used to categorize the type of **MonitoredScope** (section 2.2.5.6). Each nonzero bit of the integer MUST be one of the values in the following table.

SqlQueries	**0x1**
SPRequests	0x2

SqlQueries	0x1
ServiceCall	0x4
CriticalTrace	0x8
FileLevel	0x10
CurrentUser	0x20
WebPartEvents	0x40
VerboseTrace	0x80
CPUCycles	0x100

ValueInt: The integer value of the diagnostics data.

ValueFloat: The floating point value of the diagnostics data.

ValueString: The string of the diagnostics data.

ValueBinary: The binary value of the diagnostics data.

RowCreatedTime: The UTC date and time when the data was written to the database.

2.2.5.6 MonitoredScopes

This table stores diagnostics data.

```
PartitionId tinyint NULL,
RowId uniqueidentifier NOT NULL,
LogTime datetime NOT NULL,
MachineName nvarchar(128) NOT NULL,
CorrelationId uniqueidentifier NULL,
Name nvarchar(256) NULL,
ScopeId bigint NULL,
ParentId bigint NULL,
StartTime datetime2 NULL,
EndTime datetime2 NULL,
Flags bigint 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 Monitored Scope was initiated.

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

CorrelationId: The request identifier of the current request.

Name: The name of the diagnostics data.

ScopeId: The identifier of the diagnostics data scope within the current request.

ParentId: The identifier of the parent scope, or -1 if this scope has no parent.

StartTime: The starting UTC date of the time span of the monitored scope.

EndTime: The ending UTC date of the time span of the monitored scope.

Flags: An integer bit field used to categorize the type of MonitoredScope. Each nonzero bit of the integer MUST be one of the values in the following table

SqlQueries	**0x1**
SPRequests	0x2
ServiceCall	0x4
CriticalTrace	0x8
FileLevel	0x10
CurrentUser	0x20
WebPartEvents	0x40
VerboseTrace	0x80
CPUCycles	0x100

RowCreatedTime: The UTC date and time when the data was written to the database.

2.2.5.7 PerformanceCountersDefinitions

The **PerformanceCountersDefinitions** table stores metadata pertaining to the performance counters logging provider. The table can be queried to return the definition of performance counters that are collected from a front-end Web server, and stored as part of usage data.

```
Id int NULL,
Machine nvarchar(256) NULL,
Category nvarchar(2048) NULL,
Counter nvarchar(2048) NULL,
Instance nvarchar(2048) NULL,
```

Id: The identifier for the performance counter definition.

Machine: The name of a front-end Web server.

Category: The category of the performance counter.

Counter: The name of the enabled performance counter.

Instance: The name of an instance of the performance counter.

2.2.5.8 TimerJobUsage

The **TimerJobUsage** view is called to retrieve all information stored in the partitioned tables for the **TimerJobUsage** logging provider. This view can be queried to return data pertaining to all timer job executions during the retention period.

```
PartitionId tinyint 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,
ServiceId uniqueidentifier NULL,
WebApplicationId uniqueidentifier NULL,
JobId uniqueidentifier NULL,
ServerId uniqueidentifier NULL,
Status int NULL,
StartTime datetime NULL,
EndTime datetime NULL,
WebApplicationName nvarchar(255) NULL,
JobTitle nvarchar(255) NULL,
RequestCount int NULL,
QueryCount int NULL,
Duration bigint NULL,
RowCreatedTime datetime NOT NULL,
QueryDurationSum bigint NULL,
ServiceCallCount smallint NULL,
ServiceCallDurationSum bigint NULL,
CPUMCycles bigint 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 **timer job** completed. This value MUST NOT be NULL or empty.

MachineName: The name of the **application server** on which the timer job ran.

FarmId: The farm identifier of the farm on which the timer job ran.

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

UserLogin: This value MUST be an empty string.

CorrelationId: The request identifier for the timer job.

ServiceId: The unique identifier of the **shared service application** associated with the timer job.

WebApplicationId: The unique identifier of the Web application associated with the timer job.

JobId: The unique identifier for the timer job.

ServerId: The unique identifier of the application server on which the timer job ran.

Status: An integer field used to categorize the type of data written. This value MUST be one of the values in the following table:

Initialized	**0x1**

Initialized	**0x1**
Succeeded	0x2
Failed	0x3
Retry	0x4
Aborted	0x5

StartTime: The starting UTC date of the timer job.

EndTime: The ending UTC date of the timer job.

WebApplicationName: The name of the Web application for which the timer job is associated. This value MUST be NULL for a timer job that is not associated with a Web application.

JobTitle: A text description of the timer job.

RequestCount: The number of web requests made by the timer job.

QueryCount: The number of database server **queries** executed by the timer job.

Duration: The time in milliseconds taken for the timer job to execute.

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

QueryDurationSum: The aggregated time in milliseconds taken for all back-end database server queries performed by the timer job.

ServiceCallCount: The count of shared service application web requests performed by this timer job.

ServiceCallDurationSum: The aggregated time in milliseconds taken for all shared service application web requests generated as a result of this timer job.

CPUMCycles: The number of CPU megacycles spent processing the timer job on the application server.

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 Common Details

None.

3.2 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.

3.2.1 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**.

3.2.2 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 provides information about message processing events and sequencing rules.

3.2.5.1 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 zero.

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 zero.

Result Sets: MUST NOT return any result sets.

3.2.5.2 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 3.2.1 for specifics on the objects created by this stored procedure.

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

@TypeName: The name of the logging provider that MUST be **provisioned**. 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 zero 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 zero.

Return Values: An integer that MUST be zero.

Result Sets:

This stored procedure MUST return a prc_CreateObjectsHelper.ResultSet0

This stored procedure MUST return a prc_CreateObjectsHelper.ResultSet1

This stored procedure MUST return a prc_CreateObjectsHelper.ResultSet2

This stored procedure MUST return a prc_CreateObjectsHelper.ResultSet3

3.2.5.3 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 zero.

Return Values: An integer that MUST be zero.

Result Sets:

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

3.2.5.4 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 zero and less than or equal to 31.

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

Return Values: An integer that MUST be zero.

Result Sets: MUST NOT return any result sets.

3.2.5.5 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 zero.

Result Sets: MUST NOT return any result sets.

3.2.5.6 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 zero.

Result Sets:

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

3.2.5.7 prc_CreateRole

The **prc_CreateRole** stored procedure is called to create a new **role** in the back-end database server. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE prc_CreateRole (
@RoleName nvarchar(100)
);
```

@RoleName: The name of the created role. This MUST NOT be NULL. If an existing role is found in the back-end database server with the same name, all existing **role assignments** for the existing role MUST be deleted, and the existing role MUST be deleted. The created role MUST have no role assignments.

Return Values: An integer that MUST be zero.

Result Sets: MUST NOT return any result sets.

3.2.5.8 proc_GetCorrelationIdAndUsers

The **proc_GetCorrelationIdAndUsers** stored procedure is called to obtain the request identifiers for all monitored scopes executed after a given time.

```
PROCEDURE proc_GetCorrelationIdAndUsers (
@StartTime datetime2
);
```

@StartTime: The starting UTC date of the time span for which the monitored scope request identifiers are requested.

Return Values: An integer which MUST be zero.

Result Sets: This stored procedure must return a proc_GetCorrelationIdAndUsers.ResultSet0.

3.2.5.9 proc_GetDiagnosticsData

The **proc_GetDiagnosticsData** stored procedure is called to retrieve a list of diagnostics data with specified *CorrelationId*, *ScopeId*, and *Name*.

```
PROCEDURE proc_GetDiagnosticsData (
@CorrelationId uniqueidentifier
,@ScopeId bigint = null
,@Name nvarchar(256) = null
);
```

@CorrelationId: A request identifier for the current request. It MUST NOT be NULL.

@ScopeId: The identifier of diagnostics data scope within the current request. If this is NULL, all the diagnostics data with the given request identifier matching the name pattern MUST be retrieved. If this is not NULL, all the retrieved diagnostics data MUST have the given *CorrelationId* and *ScopeId* matching the given name pattern.

@Name: The name pattern of the diagnostics data. It can include wildcard characters. For example, if this is "A%", this stored procedure returns only the diagnostics data with names beginning with either "A" or "a". If this is NULL, the stored procedure MUST retrieve all diagnostics data with the given *CorrelationId* and *ScopeID*.

Return Values: An integer that MUST be zero.

Result Sets: This stored procedure MUST return a **proc_GetDiagnosticsData.ResultSet0** (section 2.2.4.11).

3.2.5.10 proc_GetMonitoredScope

The **proc_GetMonitoredScopes** stored procedure is called to retrieve a diagnostics data scope being run under a specified request identifier. The procedure MUST return data which correlates to the correlation identifier as specified by the *@CorrelationId* parameter. The stored procedure MUST return zero rows in the result set, if the CorrelationID does not exist or there are no monitored scopes associated with it.

```
PROCEDURE proc_GetMonitoredScope (
@CorrelationId uniqueidentifier
```

```
,@ScopeId bigint
);
```

@CorrelationId: A request identifier for the current request. It MUST not be NULL.

@ScopeId: The identifier of the diagnostics data scope within the current request. If this is null, all diagnostics data MUST be retrieved for the given request identifier.

Return Values: An integer which MUST be zero.

Result Sets: This stored procedure MUST return a result set from MonitoredScopes table.

3.2.5.11 proc_GetMonitoredScopes

The **proc_GetMonitoredScopes** stored procedure is called to retrieve a list of monitored scopes being run under a specified **correlation** identifier. The procedure MUST return data which correlates to the Correlation ID as specified by the *@CorrelationId* parameter. The stored procedure MUST return zero rows in the result set, if the CorrelationID does not exist or there is no monitored scopes associated with it.

```
PROCEDURE proc_GetMonitoredScopes (
@CorrelationId uniqueidentifier
);
```

@CorrelationId: A globally unique identifier (GUID) used to specify the entity under which monitored scopes are run.

Return Values: An integer which MUST be zero.

Result Sets:

This stored procedure MUST return a result set from MonitoredScopes table.

3.2.6 Timer Events

None.

3.2.7 Other Local Events

None.

3.3 Client Details

None.

3.3.1 Abstract Data Model

None.

3.3.2 Timers

None.

3.3.3 Initialization

Initialization of a logging provider is performed by invoking the stored procedure described in section 3.2.5.2.

3.3.4 Higher-Layer Triggered Events

None.

3.3.5 Message Processing Events and Sequencing Rules

None.

3.3.6 Timer Events

None.

3.3.7 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 zero, 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 zero, 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 zero, 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, which 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 zero, 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 2013 Preview

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 3.2.5.6: 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

T

V

X