[MS-WSSCFGD]: Windows SharePoint Services: Configuration Database Communications Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability
04/25/2008	0.2	Editorial	Revised and edited the technical content
06/27/2008	1.0	Major	Revised and edited the technical content
12/12/2008	1.01	Editorial	Revised and edited the technical content
03/18/2009	1.02	Editorial	Revised and edited the technical content
07/13/2009	1.03	Major	Changes made for template compliance
08/28/2009	1.04	Editorial	Revised and edited the technical content
11/06/2009	1.05	Editorial	Revised and edited the technical content
02/19/2010	2.0	Editorial	Revised and edited the technical content
03/31/2010	2.01	Editorial	Revised and edited the technical content
04/30/2010	2.02	Editorial	Revised and edited the technical content
06/07/2010	2.03	Editorial	Revised and edited the technical content
06/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
04/11/2012	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	2.04	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the Windows SharePoint Services: Configuration Database Communications Protocol. This protocol specifies the communications needed for one or more clients to share configuration settings by storing those settings in a central location.

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]:

Active Directory
GUID
Hypertext Transfer Protocol (HTTP)
service

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

application server
back-end database server
configuration database
configuration object
connection string
content database
content database lock
distribution list
e-mail alias
e-mail enabled list
empty GUID
front-end Web server
host header
job definition
job lock
list
list identifier
permission level
result set
return code
row version
server-relative URL
site
site collection
site collection identifier
site identifier
site template
SQL authentication
stored procedure
Structured Query Language (SQL)
target instance
timer service
Transact-Structured Query Language (T-SQL)

trusted authentication
Uniform Resource Locator (URL)
Web application

The following terms are specific to this document:

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.

[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".

[MS-WSSCADM] Microsoft Corporation, "Windows SharePoint Services Content Database Administrative Communications Protocol Specification".

[MS-WSSFO] Microsoft Corporation, "Windows SharePoint Services (WSS): File Operations Database Communications Protocol Specification".

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

1.2.2 Informative References

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

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

1.3 Protocol Overview (Synopsis)

Running a distributed service on multiple protocol clients requires all of the protocol clients to be configured identically. Otherwise, protocol clients could unintentionally produce different results given the same input. For example, if incoming **Hypertext Transfer Protocol (HTTP)** requests are distributed randomly between several **front-end Web servers**, it is essential that all of those servers are configured to listen on the same ports and respond with the same content for a specified **URL**. One way of ensuring consistency is to store the configuration data for the service in a central location. This approach has several additional benefits including support for dynamic configurations and the ability to manage a service from a central location.

This protocol specifies the communications between a computer or set of computers running one or more **services** and a **back-end database server** in which the configuration data for the services are stored. The clients of this protocol are computers running services. The protocol server is a device holding the configuration data in what will be known as the **configuration database**.

1.3.1 Configuration Object Management

Much of this protocol is concerned with the communications needed to store, retrieve, update, and perform other operations on configuration objects. Configuration objects are a mechanism of encapsulating groups of application settings.

1.3.2 File Storage

Some services require the same set of files to be present on all protocol clients. To support this requirement, this protocol specifies a second interface specifying the communications between protocol clients and protocol servers required to store and retrieve files used in the operation of the service.

1.3.3 Timer Job Management

Once a service is executing on multiple protocol clients, it becomes necessary to develop a mechanism of distributing certain tasks across those computers. This protocol specifies a mechanism of distributing these tasks using a **timer service** which runs on all clients connected to a configuration database.

1.3.4 E-mail-Enabled Lists

Some services need to store data in a **list** which is identified by an **e-mail alias**. This protocol provides a mechanism of accomplishing this by maintaining a mapping from an e-mail alias to a specific list.

1.3.5 Pending Distribution Lists

Some **permission levels** have an associated **distribution list**. If a service is required to manipulate a distribution list but the operation it is performing requires approval, the service needs a mechanism for determining when that operation has been approved. This protocol provides a mechanism of accomplishing this by maintaining a collection of permission levels whose associated distribution lists have an operation pending.

1.4 Relationship to Other Protocols

This protocol relies on [MS-TDS] as its transport protocol to call **stored procedures** to manipulate **configuration objects** and files stored in the configuration database by way of **result sets** and **return codes**.

This relationship is illustrated in the following diagram.

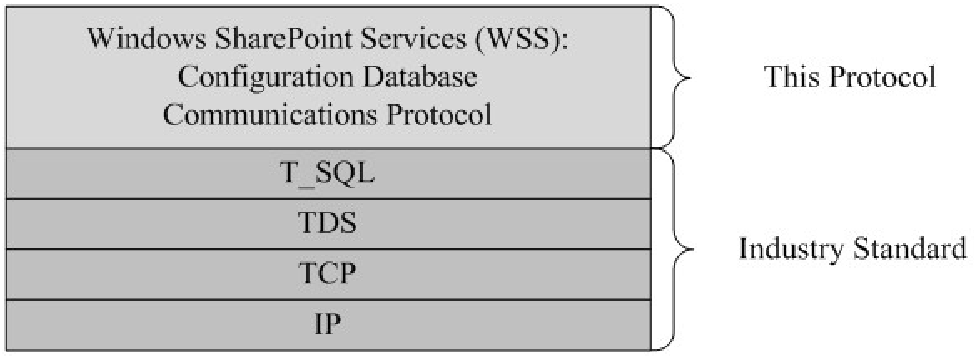
Windows SharePoint Services (WSS): Configuration Database Communications Protocol
T_SQL
TDS
TCP
IP
This Protocol
Industry Standard

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

The operations described by this protocol operate between a protocol client and a back-end database server. The protocol 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 only applicable to **application servers** when they communicate to the configuration database to manipulate configuration objects or files stored there.

1.7 Versioning and Capability Negotiation

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

1.8 Vendor-Extensible Fields

None.

1.9 Standards Assignments

None.

2 Messages

2.1 Transport

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

2.2 Common Data Types

This section contains common definitions used by this protocol.

2.2.1 Simple Data Types and Enumerations

2.2.2 Bit Fields and Flag Structures

2.2.2.1 Job Lock Type

The Job Lock Type is an integer value which indicates which type of locking a job definition uses. The value MUST be in the following table:

Value	Description
0	**None** The job definition does not use locking between protocol clients.
1	**content database** Only one job instance is permitted to process a content database at any one time.
2	**Job** Only one job instance for the job definition is permitted to execute at any one time.

2.2.2.2 Job Status Type

The Job Status Type is an integer value which indicates the execution status of a job instance. The value MUST be in the following table:

Value	Description
0	The job instance has been scheduled
1	The job instance has been initialized and is currently executing.
2	The job instance was successfully executed.
3	An error occurred while executing the job instance.
4	An error occurred during execution, but the job instance has been scheduled to retry the execution.
5	The job instance was interrupted before execution could complete and is not currently scheduled for another execution.

2.2.2.3 Lock Status Type

The Lock Status Type is an integer value which indicates the status of a **job lock** or **content database lock**. The value returned MUST be included in the following table:

Value	Description
0	The status of the lock is unknown.
1	The lock is currently held by another protocol client.
2	The lock is held by the specified protocol client.
3	An expired lock held by another protocol client was overwritten and the lock has been acquired by the specified protocol client.
4	An unexpected failure occurred while retrieving the state of the lock.

2.2.3 Binary Structures

None.

2.2.4 Result Sets

None.

2.2.5 Configuration Object Classes

The following configuration object class identifiers are used during the execution of this protocol in addition to the configuration object class identifiers specified in [MS-WSSFO] (Section 2.2.6.1.1).

Class	Class Identifier
Job Definition	3F9F635F-0036-42fe-9C2D-3284162732DB
Service	DACA2A15-B9B5-43da-BEA3-6B75FBE3A883
Shared Services Provider	9D95E78B-FA6F-4349-AD9A-43BD3EF44E99

2.2.6 Configuration Object Properties

The properties of the following configuration objects are used throughout this protocol.

2.2.6.1 Timer Job Definition

The timer job definition configuration object stores information needed to manage job instances. The parent of a timer job definition MUST be a configuration object with a class identifier of the service or the class derived from the service as specified in Section 2.2.5 or a **Web application** as specified in [MS-WSSFO] (Section 2.2.6.1.1).

Property	XPath Query	Description
Server	`/object/fld[attribute::name='m_Server']`	If the value returned by this XPath query is not null or empty, it MUST be the identifier of a configuration object. The configuration object MUST have

Property	XPath Query	Description
		the server class identifier specified in [MS-WSSFO] (Section 2.2.6.1.1). The execution of job instances for the job definition MUST execute only on the protocol client identified by the associated protocol server configuration object.
Lock Type	`/object/fld[attribute::name='m_LockType']`	A string value used to specify the job lock type used by a job definition.
Title	`/object/fld[attribute::name='m_Title']`	The string value used as a descriptive title for job definition.

2.2.6.2 Shared Services Provider

The Shared Services Provider configuration object stores the settings for resources shared by SharePoint services.

The parent of this configuration object MUST be a farm configuration object.

Property	XPath Query	Description
Shared Services Database Identifier (SSDI)	`/object/fld[@name='m_SharedServiceDatabase']`	The identifier of the Shared Services Database configuration object.

2.2.6.3 Shared Services Database

The shared services database configuration object stores the settings needed to connect to a shared services database.

Property	XPath Query	Description
Username	`/object/fld[@name='m_Username']`	The **SQL authentication** user name used to connect to the database. If this value is null or empty, the protocol client MUST use **trusted authentication**.
Password	`/object/fld[@name='m_Password']`	The SQL authentication password used to connect to the database.

2.2.6.4 E-mail-Enabled List

An e-mail enabled list maintains information about the assignment of an e-mail alias to a list and whether that e-mail alias has been marked for future deletion. An e-mail enabled list can also maintain information about a distribution list which has been marked for future deletion.

An e-mail enabled list is a complex type with the following fields, specified in **Transact-Structured Query Language (T-SQL)** format:

```
Alias                   nvarchar(128) NOT NULL
SiteId                  uniqueidentifier NOT NULL
WebId                   uniqueidentifier NOT NULL
```

```
ListId                 uniqueidentifier NOT NULL
Deleted                bit NOT NULL DEFAULT((0))
```

Alias: The e-mail alias of the list or distribution list. The alias of an e-mail enabled list is used to identify a single list or a single distribution list. As such, it MUST be different from the aliases of all other e-mail enabled lists in the configuration database.

SiteId: The site collection identifier of the site collection containing the list, if any, or the **empty GUID** if the alias belongs to a distribution list.

WebId: The **site identifier** of the **site (2)** containing the list, if any, or the empty GUID if the alias belongs to a distribution list.

ListId: The **list identifier** of the list, if any, or the empty GUID if the alias belongs to a distribution list.

2.2.6.5 Pending Distribution List

A pending distribution list encapsulates information about a distribution list which has an operation pending approval. It is a complex type with the following fields, specified in T-SQL format:

```
SiteId           uniqueidentifier NOT NULL
WebId            uniqueidentifier NOT NULL
GroupName        nvarchar(255) NOT NULL
ModifiedBy       nvarchar(255) NOT NULL
Version          rowversion
```

SiteId: The **site collection identifier** of the **site collection** containing the permission level.

WebId: The site identifier of the site containing the permission level.**GroupName:** A string containing the name of the permission level.**ModifiedBy:** A string containing the user name of the user who last performed an operation on the distribution list.**Version:** A **row version** used to identify when this distribution list last had an operation pending.

2.2.7 Tables and Views

None.

2.2.8 XML Structures

None.

3 Protocol Details

The client side of this protocol is simply a pass-through. That is, no additional timers or other state is required on the client side of this protocol. Calls made by the higher-layer protocol or application are passed directly to the transport, and the results returned by the transport are passed directly back to the higher-layer protocol or application.

3.1 Server Details

3.1.1 Abstract Data Model

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

3.1.1.1 Configuration Object Management

This protocol specifies the following types of operations used to manage configuration objects, which are specified in [MS-WSSFO] (Section 2.2.6.1).

Configuration Object Creation, Update and Deletion

The server MUST maintain a list of configuration objects. A service can store its settings on the protocol server by putting the settings in a new set of configuration objects. The configuration objects belonging to the service can then be passed to the stored procedures specified in this protocol to add them to the list on the protocol server. Later, if the settings for the service change, its configuration objects can be updated. Finally, a service can remove its configuration objects from the protocol server when they are no longer needed.

Class Registration

To enable different services to distinguish their configuration objects from those of other services, each configuration object has a class. Before a service adds a configuration object to the configuration database, it first ensures that the class of the configuration object has been added to the list of classes, which MUST be maintained by the server.

Dependency Tracking

The server MUST maintain a list of dependencies between configuration objects. By defining dependencies between configuration objects, a service can ensure that the object of the dependency is not deleted. In addition, this protocol provides a mechanism to quickly find all of a configuration object's dependencies.

File Storage

Most of the stored procedures used to retrieve configuration objects from the server retrieve the configuration objects' properties. Retrieving properties can reduce the efficiency of storing a large amount of data in configuration object properties, especially if those properties are rarely needed. To address this, the protocol server MUST maintain a list of large binary structures. This protocol specifies the stored procedures that can be used by a service to store and retrieve these large binary structures.

3.1.1.2 Timer Job Management

Like many services, the timer service stores settings in configuration objects on the protocol server. One of the primary classes of configuration object used by the timer service is a **job definition**. A job definition stores information about a task that the timer service is expected to perform on the protocol client. To deal with the complexity of coordinating job definitions executing on multiple protocol clients, this protocol specifies additional stored procedures used by the timer service to manage the creation, execution and removal of job definitions. These stored procedures can be categorized as follows:

Job Prerequisites

A job definition can optionally require a content database lock or Job Lock as specified by the **Lock Type** property. The protocol client is responsible for acquiring and maintaining the lease of these locks. Both types of locks are valid for any job instance executed on the protocol client which holds the lock. Using one of these locks ensures that a single job instance for a job definition processes the locked resources. A job lock is specific to one job definition. However, multiple job definitions can require the use of the same content database lock. In this case, multiple job definitions requiring a content database lock can execute concurrently on the same protocol client which holds the lock.

The second type of prerequisite a job definition may use is a **target instance**. A job instance can process several different resources during a single execution. A target instance provides a way to keep track of the job instance's execution progress.

Locks and target instances must be created before the job instance executes.

Job Execution

A job instance represents a single execution of a job definition on a protocol client. A single job definition may have multiple job instances executing concurrently, each on a different protocol client.

A content database job has one target instance for each content database. The job can be executed on multiple protocol clients concurrently if each protocol client has acquired one or more content database locks. For example, assume a protocol client has three content database locks. A single job instance is executed, and it handles each of the three content databases sequentially.

When a job is started using **proc_StartTimerRunningJob**, the **TargetCount** indicates how many target instances the job instance is scheduled to process. The **CurrentTarget** is a counter which indicates how many target instances have been processed.

Job Status

A protocol client may query the status of a job instance execution.

3.1.1.3 E-mail-Enabled Lists

The server MUST maintain a list of e-mail enabled lists. A service MAY use this by storing a mapping from an e-mail alias to a list identifier, site identifier and site collection identifier. Later, when the service receives incoming e-mail, it can look up the recipient's e-mail alias using the stored procedures specified in this protocol. If the e-mail alias is found, the service can update the associated list with the contents of the e-mail.

3.1.1.4 Pending Distribution Lists

The server MUST maintain a list of pending distribution lists. This MAY be used by a service which synchronizes the members of a permission level and a group maintained by an external user management system. When an operation occurs which requires the service to update the external user management system, the external system can reply that the operation has been accepted pending approval, in which case the service temporarily stores information about the operation on the protocol server. Later, the service iterates over all of the entries in the pending distribution lists list on the protocol server and queries the external system to see if the job has finished. Once the job has finished or been rejected, the service removes the operation entry from the list.

3.1.2 Timers

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

3.1.3 Initialization

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

3.1.4 Higher-Layer Triggered Events

None.

3.1.5 Message Processing Events and Sequencing Rules

This section describes the following **stored procedures**:

Procedure Name	Description
proc_AddTimerLockForJob	If the job definition uses a job lock it MUST be acquired prior to execution.
proc_AddTimerTargetInstance	If the job definition represents a one-time job and uses content database locks, then a target instance MUST be added for each content database contained in the Web application referenced by the job definition. If the job definition represents a one-time job and neither uses content database locks nor job locks, then a target instance MUST be added for each protocol server where the service associated with the job definition has been provisioned.
proc_CompleteTimerRunningJob	Complete a job instance.
proc_DeleteAllMarkedTimerLocks	Periodically delete any expired job locks.
proc_DeleteAllTimerLocksAndRunningJobs	Abort any job instances and release any acquired job locks for a specific client.
proc_DeleteTimerLockForJob	Mark for deletion or immediately delete a job lock for the specified job definition.
proc_DeleteTimerRunningJobs	Deletes all job instances for a particular client.

Procedure Name	Description
proc_DeleteTimerTargetInstance	Deletes a single target instance.
proc_DeleteTimerTargetInstances	Deletes all target instances for a job definition.
proc_dropEmailEnabledList	Removes an e-mail enabled list.
proc_dropEmailEnabledListByAlias	Removes the e-mail enabled list with the given e-mail alias.
proc_dropEmailEnabledListsByWeb	Removes all e-mail enabled lists within a specified site.
proc_DropObject	Removes a configuration object from the configuration database.
proc_dropPendingDistributionList	Removes a pending distribution list.
proc_DropSiteMap	Removes site collection lookup information from the configuration database
proc_getDeletedEmailAliases	Retrieves a list of aliases of e-mail enabled lists and distribution lists which have been marked for deletion.
proc_GetDependentObjectsByBaseClass	Retrieves a list of configuration objects which derive from a specified class and depend on a specified configuration object.
proc_getEmailEnabledListByAlias	Retrieves the site collection, site, and list identifiers of the list with a given alias.
proc_GetFile	Retrieves a file.
proc_GetFilePointer	Retrieves a TEXTPTR to a persisted file.
proc_GetNewObjects	Retrieves a list of configuration objects that have been recently created, modified or deleted.
proc_getPendingDistributionListsSinceVersion	Retrieves a list of the pending distribution lists since the specified version.
proc_GetSiteBestMatch	Retrieves information about site collections with specified properties.
proc_GetSiteCount	Retrieves an estimate of the number of site collections in a specified content database.
proc_GetSiteIdOfHostHeaderSite	Retrieves the site collection identifier of a host header site.
proc_GetSiteNames	Retrieves the **server-relative URLs** of site collections in a specified content database or Web application.
proc_GetSiteSubset	Retrieves site collection information about a limited set of site collections.
proc_GetTemplate	Retrieves the content of a **site template**.
proc_GetTimerJobLastRunTime	For job definitions that use a recurring schedule, this stored procedure returns the time of the last job execution.

Procedure Name	Description
proc_GetTimerRunningJobs	Returns the job status result set.
proc_GetTimerTargetInstance	When the job definition represents a one time job and uses a content database lock, then a target instance MUST exist.
proc_markForDeletionEmailEnabledList	Marks the specified list for deletion.
proc_markForDeletionEmailEnabledListsBySite	Marks all lists in a specified site collection for deletion.
proc_markForDeletionEmailEnabledListsByWeb	Marks all lists in a specified site for deletion.
proc_PutClass	Adds a new class to the configuration database.
proc_PutDependency	Adds a dependency between two configuration objects.
proc_putDistributionListToDelete	Adds the alias of a distribution list for future deletion.
proc_putEmailEnabledList	Adds a new e-mail enabled list.
proc_PutFileSegment	Adds a segment of a persisted file.
proc_PutObject	Adds or updates a configuration object.
proc_putPendingDistributionList	Adds a pending distribution list.
proc_PutSiteMap	Stores site collection metadata in the configuration database.
proc_RefreshAllTimerLocks	Periodically renew the lease on acquired job locks by the specified protocol client.
proc_RenameAllTimerLocksAndRunningJobs	Renames job locks and job instances in the configuration database.
proc_RenameSiteMap	Renames the **host header** for site collections in the configuration database.
proc_StartTimerRunningJob	Starts a job instance.
proc_UpdateTimerRunningJobProgress	Optionally called to update the progress of a running job instance.
proc_UpdateTimerRunningJobTarget	Used to update the current target instance being processed by a job instance.

The T-SQL syntax for each stored procedure and result set and the variables they are composed of, is specified in the [MSDN-TSQL-Ref] protocol. In the T-SQL syntax, the variable name is followed by the type of the variable which MAY optionally have a length value in brackets and MAY optionally have a default value indicated by an equals sign followed by the default value.

3.1.5.1 proc_AddTimerLockForJob

The **proc_AddTimerLockForJob** stored procedure is called to request a job lock for the specified protocol client and job definition. The stored procedure is specified using T-SQL syntax, as follows.

```
PROCEDURE proc_AddTimerLockForJob(
        @JobId                              uniqueidentifier,
        @ServerName                         nvarchar(128),
        @LockTimeout                        int,
        @LockStatus                         int OUTPUT,
        @LockExpiredServerName              nvarchar(128) OUTPUT
    );
```

@JobId: The configuration object identifier of the job definition for which a job lock is requested.

@ServerName: The name of the protocol client requesting the job lock.

@LockTimeout: The maximum age in minutes of an existing job lock before it is considered expired.

@LockStatus: A **Lock Status Type** which returns the status of the requested job lock. The valid values of this type are specified in Section 2.2.2.3. The output variable **@LockStatus** MUST be 2 if the requesting protocol client successfully acquires a lock for a new **JobId** or if the requesting protocol server already holds the lock for the specified **JobId**, whether it has expired or not.

@LockExpiredServerName: The name of the protocol client that holds an expired job lock which is overwritten. The output variable **@LockExpiredServerName** MUST be the name of the protocol client which previously held the lock if the return value in **@LockStatus** is equal to 3 and the lock is not marked for deletion. The output variable **@LockExpiredServerName** MUST be NULL if the return value in **@LockStatus** is equal to 3 and the lock is marked for deletion.

Return Values:

The **proc_AddTimerLockForJob** stored procedure returns an integer return code which MUST be in the following table:

Value	Description
0	Successful execution.
167	An error occurred while retrieving the state of the job lock.

The **proc_AddTimerLockForJob** stored procedure MUST NOT return a result set.

3.1.5.2 proc_AddTimerTargetInstance

The **proc_AddTimerTargetInstance** stored procedure is called to associate a job definition with a target instance. This allows the job definition to be executed on that target instance. The **proc_AddTimerTargetInstance** stored procedure is specified using T-SQL syntax, as follows:

```
    PROCEDURE proc_AddTimerTargetInstance (
        @JobId                        uniqueidentifier,
        @TargetInstanceID             uniqueidentifier
    );
```

@JobId: This MUST be the configuration object identifier of a job definition.

@TargetInstanceId: This MUST be the identifier of the configuration object associated with the target instance to be associated with the job definition.

Return Values: The **proc_AddTimerTargetInstance** stored procedure returns an integer return code which MUST be in the following table:

Value	Description
0	The specified job definition exists.
31	The specified job definition does not exist, and so wasn't associated with the specified target instance.

The **proc_AddTimerTargetInstance** stored procedure MUST NOT return a result set.

3.1.5.3 proc_CompleteTimerRunningJob

The **proc_CompleteTimerRunningJob** stored procedure is called by the protocol client after a job instance completes. The **proc_CompleteTimerRunningJob** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_CompleteTimerRunningJob (
    @ServiceId               uniqueidentifier,
    @VirtualServerId         uniqueidentifier,
    @JobId                   uniqueidentifier,
    @ServerName              nvarchar(128),
    @Status                  int
);
```

@ServiceId: The configuration object identifier of the service associated with the job definition.

@VirtualServerId: The configuration object identifier of the Web application associated with the job definition. This MUST be NULL when the job definition is associated only with a service.

@JobId: The configuration object identifier of the job definition.

@ServerName: The name of the protocol client where the job instance is executed.

@Status: The job status type of the job instance.

Return Values: The **proc_CompleteTimerRunningJob** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_CompleteTimerRunningJob** stored procedure MUST NOT return a result set.

3.1.5.4 proc_DeleteAllMarkedTimerLocks

The **proc_DeleteAllMarkedTimerLocks** stored procedure is called to delete all expired job locks marked for deletion. The stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_DeleteAllMarkedTimerLocks(
    @LockTimeout             int
);
```

@LockTimeout: The maximum age in minutes of an existing job lock before this procedure considers the lock to have expired.

Return Values:

The **proc_DeleteAllMarkedTimerLocks** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_DeleteAllMarkedTimerLocks** stored procedure MUST NOT return a result set.

3.1.5.5 proc_DeleteAllTimerLocksAndRunningJobs

The **proc_DeleteAllTimerLocksAndRunningJobs** stored procedure is called to delete all job locks marked for deletion and optionally delete all job instances for a specified protocol client. The stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_DeleteAllTimerLocksAndRunningJobs(
    @ServerName               nvarchar(128),
    @AbortRunningJobs         bit
);
```

@ServerName: The name of the protocol client holding the job lock(s) to be deleted.

@AbortRunningJobs: A flag which indicates whether to also delete job instances. The value is a bit and MUST be in the following table:

Value	Description
0	Delete job instances.
1	Mark job instances with current job status type equal to 1 to have job status type equal to 5, but do not delete them from the database.

Return Values:

The **proc_DeleteAllTimerLocksAndRunningJobs** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_DeleteAllTimerLocksAndRunningJobs** stored procedure MUST NOT return a result set.

3.1.5.6 proc_DeleteTimerLockForJob

The **proc_DeleteTimerLockForJob** stored procedure is called to delete a job lock for a specified job definition if the **@ServerName** parameter matches the protocol client which holds the lock or if the **@ServerName** parameter is NULL.

The stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_DeleteTimerLockForJob(
    @JobId                    uniqueidentifier,
    @ServerName               nvarchar(128),
    @MarkOnly                 bit = 1
);
```

@JobId: The configuration object identifier of the job definition for which a job lock is being deleted.

@ServerName: The name of the protocol client or NULL.

@MarkOnly: A flag which indicates whether to delete the job lock or to just mark it for deletion at a later time. The value is a bit and MUST be in the following table:

Value	Description
0	Delete job lock entry from the database.
1 (default value)	Mark the job lock for deletion and keep the entry in the database.

Return Values:

The **proc_DeleteTimerLockForJob** stored procedure returns an integer return code which MUST be in the following table:

Value	Description
0	One or more timer job locks were deleted.
31	No timer job locks were deleted.

The **proc_DeleteTimerLockForJob** stored procedure MUST NOT return a result set.

3.1.5.7 proc_DeleteTimerRunningJobs

The **proc_DeleteTimerRunningJobs** stored procedure is called to delete a set of job instances for the specified job definition. The **proc_DeleteTimerRunningJobs** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_DeleteTimerRunningJobs (
    @ServiceId               uniqueidentifier,
    @VirtualServerId         uniqueidentifier,
    @JobId                   uniqueidentifier,
    @ServerName              nvarchar(128) = NULL
);
```

@ServiceId: The configuration object identifier of the service associated with the job definition.

@VirtualServerId: The configuration object identifier of the Web application associated with the job definition. This parameter MUST be NULL when the specified job definition is associated only with a service.

@JobId: The configuration object identifier of the job definition.

@ServerName: The name of the protocol client where the job instance is executed. The default value is NULL, which indicates that the job instances for the specified job definition will be deleted for all protocol clients.

Return Values: The **proc_DeleteTimerRunningJobs** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_DeleteTimerRunningJobs** stored procedure MUST not return a result set.

3.1.5.8 proc_DeleteTimerTargetInstance

The **proc_DeleteTimerTargetInstance** stored procedure is called to disassociate a job definition from a target instance. This means that the job definition will no longer be executed on that target instance. The **proc_DeleteTimerTargetInstance** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_DeleteTimerTargetInstance (
    @JobId                    uniqueidentifier,
    @TargetInstanceID         uniqueidentifier,
    @Exists                   bit OUTPUT
);
```

@JobId: This MUST be the configuration object identifier of a job definition.

@TargetInstanceId: This MUST be the configuration object **Identifier** of a target instance associated with the job definition.

@Exists: This MUST be a bit value indicating whether the specified job definition has any target instance**s** still associated with it after the input target instance is disassociated. The value returned MUST be in the following table:

Value	Description
0	The job definition has no target instances associated with it.
1	The job definition still has some target instances associated with it.

Return Values: proc_DeleteTimerTargetInstance returns an integer return code which MUST be in the following table:

Value	Description
0	The job definition has no target instances associated with it.
1	The job definition still has some target instances associated with it.

The **proc_DeleteTimerTargetInstance** stored procedure MUST NOT return a result set.

3.1.5.9 proc_DeleteTimerTargetInstances

The **proc_DeleteTimerTargetInstances** stored procedure is called to disassociate a job definition from all of its target instances.

```
PROCEDURE proc_DeleteTimerTargetInstances (
    @JobId              uniqueidentifier
);
```

@JobId: This MUST be the configuration object identifier of a job definition.

Return Values:

The stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_DeleteTimerTargetInstances** stored procedure MUST NOT return a result set.

3.1.5.10 proc_dropEmailEnabledList

The proc_dropEmailEnabledList stored procedure is called to remove a specific list from the **e-mail enabled list** collection. The **proc_dropEmailEnabledList** stored procedure is specified using T-SQL syntax, as follows.

```
PROCEDURE proc_dropEmailEnabledList(
    @SiteId             uniqueidentifier,
    @WebId              uniqueidentifier,
    @ListId             uniqueidentifier
);
```

@**SiteId**: The site collection identifier of the site collection containing the site.

@**WebId**: The site identifier of the site which contains the list.

@**ListId**: The list identifier of the list to remove from the e-mail enabled list collection.

Return Values: The stored procedure returns an integer return code which MUST be 0.

The stored procedure MUST NOT return any result sets.

3.1.5.11 proc_dropEmailEnabledListByAlias

The **proc_dropEmailEnabledListByAlias** stored procedure is called to remove a specific list from the e-mail enabled list collection.

The **proc_dropEmailEnabledListByAlias** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_dropEmailEnabledListByAlias(
    @Alias               nvarchar(128)
);
```

@**Alias**: The e-mail alias of the e-mail enabled list.

Return Values: The stored procedure returns an integer return code which MUST be 0.

The stored procedure MUST NOT return any result sets.

3.1.5.12 proc_dropEmailEnabledListsByWeb

The **proc_dropEmailEnabledListsByWeb** stored procedure is called to remove all e-mail enabled list items with the specified site collection identifier and site identifier, as follows:

The proc_dropEmailEnabledListsByWeb stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_dropEmailEnabledListsByWeb(
    @SiteId             uniqueidentifier,
    @WebId              uniqueidentifier
);
```

@**SiteId**: The site collection identifier of the site collection containing the site.

@WebId: The site identifier of the site from which to remove all e-mail enabled lists in the e-mail enabled list collection.

Return Values: The stored procedure returns an integer return code which MUST be 0.

The stored procedure MUST NOT return any result sets.

3.1.5.13 proc_DropObject

The **proc_DropObject** stored procedure is called to remove a configuration object from the configuration database. The configuration database MUST prevent a configuration object from being deleted if another configuration object depends on it. The configuration database MUST remove a configuration object when its parent is removed.

The **proc_DropObject** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_DropObject(
    @Id               uniqueidentifier
);
```

@Id: Contains the configuration object identifier of the configuration object to be removed.

Return Values: The **proc_DropObject** stored procedure returns an integer return code which MUST be listed in the following table:

Value	Description
0	Successful execution.
1	The configuration object identified by **@Id** was not found in the configuration database.

The **proc_DropObject** stored procedure MUST NOT return a result set.

3.1.5.14 proc_dropPendingDistributionList

The **proc_dropPendingDistributionList** stored procedure is called to remove a distribution list belonging to a permission level from the set of distribution lists which have an operation pending. The stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_dropPendingDistributionList(
    @SiteId            uniqueidentifier,
    @WebId             uniqueidentifier,
    @GroupName         nvarchar(255)
);
```

@SiteId: The site collection identifier of the site collection to which the permission level belongs.

@WebId: The site identifier of the site to which the permission level belongs.

@GroupName: The string name of the permission level whose associated distribution list is to be removed from the set of distribution lists which have an operation pending.

Return Values: The stored procedure returns an integer return code which MUST be 0.

The stored procedure MUST NOT return any result sets.

3.1.5.15 proc_DropSiteMap

The **proc_DropSiteMap** stored procedure is called to delete a reference to a site collection. **proc_DropSiteMap** is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_DropSiteMap (
    @Id                uniqueidentifier
);
```

@ Id: The site collection identifier of the site collection **whose reference is being deleted**.

Return Values: The proc_DropSiteMap stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_DropSiteMap** stored procedure MUST NOT return a result set.

3.1.5.16 proc_getDeletedEmailAliases

The **proc_getDeletedEmailAliases** stored procedure is called to return the e-mail aliases of e-mail enabled lists and distribution lists which have been marked as deleted. The **proc_getDeletedEmailAliases** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_getDeletedEmailAliases();
```

Return Values: The stored procedure returns an integer return code which MUST be 0.

The stored procedure MUST return 1 result set:

3.1.5.16.1 DeletedEmailAliases Result Set

The **DeletedEmailAliases** result set MUST contain all elements in the collection of e-mail enabled list elements which have been marked for deletion. The **DeletedEmailAliases** result set is specified using T-SQL syntax, as follows:

```
Alias          nvarchar(128),
ListId         uniqueidentifier;
```

Alias: The e-mail alias of the e-mail enabled list.

ListId: The **list identifier** of the e-mail enabled list or the empty GUID if the alias belongs to a distribution list.

3.1.5.17 proc_GetDependentObjectsByBaseClass

The **proc_GetDependentObjectsByBaseClass** stored procedure is called to retrieve a list of configuration objects which are in the inheritance hierarchy of the specified class and which depend on a specified configuration object. The **proc_GetDependentObjectsByBaseClass** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_GetDependentObjectsByBaseClass (
    @BaseClassId              uniqueidentifier,
    @DependeeId               uniqueidentifier
);
```

@BaseClassId: Contains the **ClassId** of the class at the root of the inheritance hierarchy of the classes of the returned configuration objects.

@DependeeId: Contains the configuration object identifier of the configuration object whose dependents are to be retrieved.

Return Values: The **proc_GetDependentObjectsByBaseClass** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_GetDependentObjectsByBaseClass** stored procedure MUST return a single result set as follows:

3.1.5.17.1 Dependent Object Ids Result Set

The **Dependent Object Ids** result set returns a set of configuration object identifiers which depend on the specified configuration object and which have classes in the inheritance hierarchy of the specified class. The **Object Ids** result set MUST be returned. The **Object Ids** result set MUST return 1 or more rows if there are configuration objects that match the input parameters, Otherwise, it MUST return 0 rows. The **Object Ids** result set is specified using T-SQL syntax, as follows:

```
Id               uniqueidentifier;
```

Id: The configuration object identifier of a configuration object.

3.1.5.18 proc_getEmailEnabledListByAlias

The **proc_getEmailEnabledListByAlias** stored procedure is called to search the e-mail enabled list collection, across all site elements and site collection elements, and return a result set of all non-deleted e-mail enabled list elements with the specified e-mail alias.

The **proc_getEmailEnabledListByAlias** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_getEmailEnabledListByAlias (
    @Alias             nvarchar(128)
);
```

@**Alias**: The e-mail alias of the e-mail enabled list to retrieve.

Return Values: The stored procedure returns an integer return code which MUST be 0.

The stored procedure MUST return 1 result set:

3.1.5.18.1 EmailEnabledListByAlias Result Set

The **EmailEnabliedByAlias** result set MUST contain all rows in the collection of e-mail enabled list elements whose e-mail alias is equal to **@Alias**, and which have not been deleted. It MUST contain 0 or more rows each of which is specified using T-SQL syntax, as follows:

```
SiteId                 uniqueidentifier,
WebId                  uniqueidentifier,
ListId                 uniqueidentifier;
```

SiteId: The site collection identifier of the site collection containing the e-mail enabled list.

WebId: The site identifier of the site containing the e-mail enabled list.

ListId: The list identifier of the e-mail enabled list.

3.1.5.19 proc_getFile

The **proc_getFile** stored procedure is called to retrieve a file from the configuration database. The **proc_getFile** stored procedure is specified using T-SQL syntax, as follows:

```
  PROCEDURE proc_getFile(
      @ObjectId          uniqueidentifier
  );
```

@ObjectId: The configuration object identifier associated with the file to be retrieved.

Return Values: The **proc_getFile** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_getFile** stored procedure MUST return a single result set as follows:

3.1.5.19.1 File Result Set

File returns the file referred to by **@ObjectId**. The file result set MUST be returned and MUST contain only 1 row when **@ObjectId** corresponds to the **GUID** of a file in the database. When **@ObjectId** does not correspond to the GUID of a file in the database, the file result set MUST be returned and MUST contain 0 rows. The file result set is specified using T-SQL syntax, as follows:

```
  FileImage              image;
```

FileImage: Contains the file referred to by the **@ObjectId** parameter.

3.1.5.20 proc_getFilePointer

The **proc_getFilePointer** stored procedure is called to retrieve a TEXTPTR to a persisted file. The **proc_getFilePointer** stored procedure is specified using T-SQL syntax, as follows:

```
  PROCEDURE proc_getFilePointer(
      @ObjectId          uniqueidentifier,
      @Pointer           binary(16) output
  );
```

@ObjectId: The GUID of the file for which a TEXTPTR is being retrieved. If a file is being stored for the first time, **@ObjectId** MUST be a configuration object identifier.

@Pointer: Before returning, the protocol server MUST set this to the address of the location of the file referred to by **@ObjectId**. This value MUST be a TEXTPTR. If **@Pointer** is not NULL it MUST contain a valid handle of the type returned by the TEXTPTR T-SQL function specified in [MSDN-TSQL-Ref].

Return Values: The **proc_getFilePointer** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_getFilePointer** stored procedure MUST NOT return a result set.

3.1.5.21 proc_getNewObjects

The **proc_getNewObjects** stored procedure is called to retrieve new, changed, and deleted configuration objects whose configuration object version is greater than the value of **@NewestCachedVersion** as well as other configuration objects which depend on them. The **proc_getNewObjects** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_getNewObjects(
    @NewestCachedVersion      rowversion
);
```

@NewestCachedVersion: Contains the lowest, non-inclusive configuration object version of configuration objects sought.

Return Values: The **proc_getNewObjects** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

If no configuration objects have a configuration object version greater than **@NewestCachedVersion**, the **proc_getNewObjects** stored procedure MUST NOT return any result sets. Otherwise, the **proc_getNewObjects** stored procedure MUST return exactly 4 result sets in the order as specified in Section 3.1.5.21.1 through Section 3.1.5.21.4.

3.1.5.21.1 Last Update Result Set

The **LastUpdate** result set returns the maximum configuration object version. The **LastUpdate** result set MUST be returned and MUST contain 1 row specified using T-SQL syntax, as follows:

```
Version                rowversion;
```

Version: A row version value that MUST be greater than the maximum configuration object version used in the configuration database.

3.1.5.21.2 Modified Objects Result Set

The Modified Objects result set returns configuration objects created or modified after **@NewestCachedVersion**. If no configuration objects have been created or modified after **@NewestCachedVersion**, the **Objects** result set MUST NOT contain rows. If configuration objects have been modified after **@NewestCachedVersion**, the **Objects** result set MUST be returned and MUST contain 1 or more rows. The **Objects** result set is specified using T-SQL syntax, as follows:

```
Id                     uniqueidentifier,
ParentId               uniqueidentifier,
ClassId                uniqueidentifier,
Name                   nvarchar(128),
Status                 int,
Version                rowversion,
Properties             ntext;
```

Id: Contains the identifier of the configuration object.

ParentId: Contains the parent identifier of the configuration object.

ClassId: Contains the class identifier of the configuration object.

Name: Contains the name of the configuration object. It MUST NOT be NULL.

Status: Contains the status of the configuration object.

Version: Contains the version of the configuration object. This value MUST be greater than **@NewestCachedVersion**.

Properties: Contains the properties of the configuration object.

3.1.5.21.3 Dependencies Result Set

The **Dependencies** result set returns **Ids** of configuration objects which depend on configuration objects from the **Objects** result set. The **Dependencies** result set MUST be returned and MUST return 0 or more rows.

The **Dependencies** result set is specified using T-SQL syntax, as follows:

```
DependantId            uniqueidentifier;
```

DependantId: Contains the configuration object identifier of the dependent configuration object.

3.1.5.21.4 Tombstones Result Set

The **Tombstones** result set returns the **Ids** of configuration objects which have been deleted since **@NewestCachedVersion**. The **Tombstones** result set MUST be returned and MUST return 0 or more rows.

The **Tombstones** result set is specified using T-SQL syntax, as follows:

```
Id                    uniqueidentifier,
Version               rowversion;
```

Id: Contains the identifier of the deleted configuration object.

Version: Contains the version of the deleted configuration object. This value MUST be greater than **@NewestCachedVersion**.

3.1.5.22 proc_getPendingDistributionListsSinceVersion

The **proc_getPendingDistributionListsSinceVersion** is called to get all distribution lists which have an operation pending whose row version greater than **@Version** parameter. The stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_getPendingDistributionListsSinceVersion(
    @Version           rowversion
);
```

@Version: The row version which forms the exclusive lower bound of the result set.

Return Values: The stored procedure returns an integer return code which MUST be 0.

The stored procedure MUST return 1 result set:

3.1.5.22.1 PendingDistrubutionLists Result Set

The **PendingDistributionLists** result set returns a set of distribution lists which need approval. The **PendingDistributionLists** result set MUST contain all rows of distribution lists that have an operation pending whose row version is greater than the **@Version** parameter value. The **PendingDistributionLists** result set is specified using T-SQL syntax, as follows:

```
SiteId                  uniqueidentifier,
WebId                   uniqueidentifier,
GroupName               nvarchar(255),
{Version}               bigint;
```

SiteId: The site collection identifier of the site collection to which the distribution list belongs.

WebId: The site identifier of the site to which the distribution list belongs.

GroupName: The name of the permission level whose associated distribution list has an operation pending.

Version: The row version of the distribution list.

3.1.5.23 proc_getSiteBestMatch

The **proc_getSiteBestMatch** stored procedure is called to search for the best matched site collection from the specified criteria. The **proc_getSiteBestMatch** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_getSiteBestMatch(
    @DatabaseId                uniqueidentifier = NULL,
    @ApplicationId             uniqueidentifier = NULL,
    @PathSearch                nvarchar(128),
    @BestMatchOffsetScope      int = 0,
    @CollectionType            int,
    @BestMatchSiteId           uniqueidentifier output,
    @BestMatchDatabaseId       uniqueidentifier output,
    @BestMatchApplicationId    uniqueidentifier output,
    @BestMatchOffset           int output
);
```

@DatabaseId: Contains the configuration object identifier of the content database to search for the best matched site collection. The value of this parameter MUST be NULL or MUST correspond to an existing content database.

@ApplicationId: Contains the configuration object identifier of the Web application to search for the best matched site collection. The value of this parameter MUST be NULL or MUST correspond to an existing Web application.

@PathSearch: Contains the prefix of a server-relative URL to search for. This parameter MUST NOT be NULL.

@BestMatchOffsetScope: Specifies the scope when calculating the value of @**BestMatchOffset** for a site collection. The value MUST be listed in the following table:

Value	Description
0	No limit
1	Include site collections belonging to the same Web application as the best matched site collection.
2	Include site collections belonging to the same content database as the best matched site collections.

@CollectionType: Contains the type of site collections for the best matched site collection. The value MUST be specified and MUST be a value listed in the following table:

Value	Description
0	All site collections.
1	Only site collections which do Redirect to other site collections .
2	Only site collections which do not Redirect to other site collections.
3	Only site collections that have been upgraded.

@BestMatchSiteId: MUST return the site collection identifier of the best matched site collection with respect to the value of **@CollectionType** or MUST return NULL if no matches are found. If **@DatabaseId** is NULL, it MUST return the site collection identifier of the best matched site collection across all content databases. If **@DatabaseId** corresponds to the configuration object identifier of a content database, then it MUST return the site collection identifier of the best matched site collection within that content database. If **@ApplicationId** is NULL, it MUST return the site collection identifier of the best matched site collection across all Web applications. If **@ApplicationId** corresponds to the configuration object identifier of a content database, then it MUST return the site collection identifier of the best matched site collection within that Web application.

@BestMatchDatabaseId: MUST return the configuration object identifier of the content database containing the best matched site collection or MUST return NULL if no matches are found.

@BestMatchApplicationId: MUST return the configuration object identifier of the Web application containing the best matched site collection or MUST return NULL if no matches are found.

@BestMatchOffset: If a match is found, this parameter MUST return the total number of site collections that precede the best-matched site collection's server-relative URL with respect to the value of **@CollectionType** in alphabetical order and with respect to the collation of the content database. If **@BestMatchOffsetScope** is 0, @BestMatchOffset MUST include the number of site collections that precede the best-matched site collection's server-relative URL across all Web applications and all content databases**. If @BestMatchOffsetScope** is 1, **@BestMatchOffset** MUST include the number of site collections which precede the best-matched site collection's server-relative URL across the Web applications identified by **@BestMatchApplicationId**. If **@BestMatchOffsetScope** is 2, **@BestMatchOffset** MUST include the number of site collections that precede the best-matched site collection's server-relative URL across the content databases identified by **@BestMatchDatabaseId**. If matches are not found, this parameter MUST return 0.

Return Values: The **proc_getSiteBestMatch** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_getSiteBestMatch** stored procedure MUST NOT return a result set.

3.1.5.24 proc_getSiteCount

The **proc_getSiteCount** stored procedure is called to retrieve the number of site collections in the specified content database. The **proc_getSiteCount** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDUREproc_getSiteCount(
    @DatabaseId              uniqueidentifier = NULL
);
```

@DatabaseId: Contains the configuration object identifier of the content database whose sites are counted.

Return Values: The **proc_getSiteCount** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_getSiteCount** stored procedure MUST return a single result set as follows:

3.1.5.24.1 SiteCount Result Set

SiteCount MUST return the number of site collections in the content database identified by **@DatabaseId** and MUST return exactly 1 row. If **@DatabaseId** is NULL, it MUST return the total number of site collections in all content databases. The **SiteCount** result set is specified using T-SQL syntax, as follows:

```
{sitecount}             int;
```

sitecount: Contains the number of site collections in the content database(s).

3.1.5.25 proc_GetSiteIdOfHostHeaderSite

The **proc_GetSiteIdOfHostHeaderSite** stored procedure is called to get the site collection identifier of the site collection represented by the specified host header site identifier. The **proc_GetSiteIdOfHostHeaderSite** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_GetSiteIdOfHostHeaderSite (
    @HostHeader        nvarchar(128)
);
```

@HostHeader: The site identifier of the host header for the site collection to be returned.

Return Values: The **proc_GetSiteIdOfHostHeaderSite** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_GetSiteIdOfHostHeaderSite** stored procedure MUST return 1 result set as follows.

3.1.5.25.1 ID Result Set

The **ID** result set returns the site collection identifier of the site collection when the host header site identifier of that site collection matches the value of **@HostHeader**. The **ID** result set MUST be returned and MUST contain one row when the host header site identifier of a site collection matches the value of **@HostHeader**. When **@HostHeader** does not match any host header site identifiers, The **ID** result set MUST be returned and MUST NOT contain any rows. The **ID** result set is specified using T-SQL syntax, as follows:

```
Id              uniqueidentifier;
```

Id: Contains the site collection identifier with a host header site identifier matching **@HostHeader**.

3.1.5.26 proc_getSiteNames

The **proc_getSiteNames** stored procedure is called to retrieve server-relative URLs of all the site collections within a container. The **proc_getSiteNames** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_getSiteNames (
    @ContainerId              uniqueidentifier,
    @ContainerType            int,
    @CollectionType           int
);
```

@ContainerId: Contains the configuration object identifier of the container from which site collections are retrieved. This value MUST correspond to the configuration object identifier of a content database or Web application.

@ContainerType: Specifies the type of the container from which site collections are retrieved. The value MUST be listed in the following table.

Value	Description
0	Indicates that the protocol server MUST return the paths of all site collections in the Web application where **@ContainerId** matches the configuration object identifier of the configuration object**.**
1	Indicates that the protocol server MUST return the paths of all site collection in the content database where **@ContainerId** matches the configuration object identifier of the configuration object.

@CollectionType: Specifies the type of collection to retrieve the sites. The value MUST be listed in the following table:

Value	Description
0	All site collections.
1	Only site collections which do Redirect to other site collections .
2	Only site collections which do not Redirect to other site collections.

Return Values: The **proc_getSiteNames** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_getSiteNames** stored procedure MUST return a single result set as follows:

3.1.5.26.1 Path Result Set

The **Path** result set returns the server-relative URLs of all the site collections in a container. The **Path** result set MUST be returned if 1 or more site collections match the values of **@ContainerId**, **@ContainerType**, and **@CollectionType**. If no site collections are found that match the values of **@ContainerId**, **@ContainerType**, and **@CollectionType**, the **Path** result set MUST be returned and MUST NOT contain any rows.

The **Path** result set is specified using T-SQL syntax, as follows:

```
Path                    nvarchar(128);
```

Path: Contains the server-relative URL for the site collection.

3.1.5.27 proc_getSiteSubset

The **proc_getSiteSubset** stored procedure is called to retrieve the site collections within a Web application or within a content database. The **proc_getSiteSubset** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_getSiteSubset (
    @DatabaseId              uniqueidentifier = NULL,
    @ApplicationId           uniqueidentifier = NULL,
    @PageSize                int,
    @StartRow                int,
    @SortDirection           nvarchar(4),
    @CollectionType          int
);
```

@DatabaseId: Contains the configuration object identifier of a content database.

@ApplicationId: Contains the configuration object identifier of a Web application.

@PageSize: Contains the number of rows to be returned at a time.

@StartRow: Contains the index of the starting row from which the data will be retrieved.

@SortDirection: Contains the order in which the data is sorted. The value MUST be listed in the following table:

Value	Description
DESC	Data will be returned in descending order according to id.
ASC	Data will be returned in ascending order according to id.

@CollectionType: Contains the type of site collections. The value MUST be listed in the following table:

Value	Description
0	All site collections.
1	Only site collections which redirect to other site collections .
2	Only site collections which do not redirect to other site collections.
3	Only site collections which have been upgraded.

Return Values: The **proc_getSiteSubset** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_getSiteSubset** stored procedure MUST return a single result set as follows:

3.1.5.27.1 Site Result Set

Site returns the site collections identified by **@CollectionType**, **@DatabaseId**, and **@ApplicationId**. If **@DatabaseId** is NULL, the **Site** result set MUST contain one or more rows for the site collections in all **content databases**. If **@ApplicationId** is NULL, the **Site** result set MUST return all the site collections in all Web applications. The number of rows in the **Site** result set MUST NOT be greater than the value of **@PageSize**. If there are **no proc_getDeletedEmailAliases** site collections in the content database identified by **@DatabaseId**, the **Site** result set MUST be returned and MUST NOT contain any rows. If there are no site collections in the Web application identified by **@DatabaseId**, the **Site** result set MUST be returned and MUST NOT contain any rows.

The **Site** result set is specified using T-SQL syntax, as follows:

```
Id                      uniqueidentifier,
Path                    nvarchar(128);
```

Id: Contains the identifier of the site collection.

Path: Contains the server-relative URL for the site collection.

3.1.5.28 proc_getTemplate

The **proc_getTemplate** stored procedure is called to retrieve the content of a site template. The **proc_gettemplate** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_gettemplate(
    @ObjId              uniqueidentifier = NULL
);
```

@ObjId: The configuration object identifier of the site template whose content is being retrieved.

Return Values: The **proc_gettemplate** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_gettemplate** stored procedure MUST return a single result set as follows:

3.1.5.28.1 Template Result Set

The **Template** result set MUST be returned and it MUST contain 0 or more rows. When **@ObjId** is not specified or is NULL, the **Template** result set MUST be returned and it MUST contain 1 row for each site template stored in the database. When **@ObjId** is not NULL and matches the configuration object identifier of an existing site template, the **Template** result set MUST be returned and it MUST contain 1 row. When there are no site templates stored in the database or when **@ObjId** does not match the configuration object identifier of an existing site template, the **Template** result set MUST be returned and MUST contain 0 rows.

The **Template** result set is specified using T-SQL syntax, as follows:

```
FileImage              image;
```

FileImage: MUST contain the contents of the site template referred to by **@ObjId**.

3.1.5.29 proc_GetTimerJobLastRunTime

The **proc_GetTimerJobLastRunTime** stored procedure is called to get the last time a job instance was executed for the specified job definition**.** The **proc_GetTimerJobLastRunTime** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_GetTimerJobLastRunTime (
    @ServiceId                uniqueidentifier,
    @VirtualServerId          uniqueidentifier,
    @JobId                    uniqueidentifier,
    @LastRunTime              datetime = NULL OUTPUT
);
```

@ServiceId: The configuration object identifier of the service associated with the job definition.

@VirtualServerId: The configuration object identifier of the Web application associated with the job definition. This parameter MUST be NULL when the specified job definition is associated only with a service.

@JobId: The configuration object identifier of the job definition.

@LastRunTime: Output value. If the function succeeds, the value is the last time a job instance was executed for the specified job definition. Otherwise, the value MUST be ignored.

Return Values: The **proc_GetTimerJobLastRunTime** stored procedure returns an integer return code which MUST be in the following table.

Value	Description
0	Successful execution.
31	There is no record of a job instance execution for the specified job definition.

The **proc_GetTimerJobLastRunTime** stored procedure MUST not return a result set.

3.1.5.30 proc_GetTimerRunningJobs

The **proc_GetTimerRunningJobs** stored procedure returns a set of job status entries for the specified service or Web application. The **proc_GetTimerRunningJobs** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_GetTimerRunningJobs(
    @ServiceId               uniqueidentifier,
    @VirtualServerId         uniqueidentifier
);
```

@ServiceId: The configuration object identifier of a service.

@VirtualServerId: The configuration object identifier of a Web application. This parameter MUST be NULL when retrieving the Job Status result set for a service.

Return Values: The **proc_GetTimerRunningJobs** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The proc_GetTimerRunningJobs stored procedure MUST return a single **Job Status** result set that contains 0 or more rows and is specified as follows:

3.1.5.30.1 Job Status Result Set

The **Job Status** result set returns an unordered list of job status entries for the specified service or Web application. If the **@VirtualServerId** parameter is NULL, then the set of entries for the service MUST be returned. Otherwise, entries MUST be restricted to those associated with the specified WebApplication. The **Job Status** result set MUST contain one row for each job status entry and is specified using T-SQL syntax, as follows:

```
ServiceId                   uniqueidentifier,
VirtualServerId             uniqueidentifier,
JobId                       uniqueidentifier,
JobTitle                    nvarchar(255),
ServerName                  nvarchar(128),
Status                      int,
StartTime                   datetime,
CurrentTarget               int,
TargetCount                 int,
CurrentTargetPercentDone    int;
```

ServiceId: The configuration object identifier of the service.

VirtualServerId: The configuration object identifier of the Web application. This variable MUST be NULL if the job definition is not associated with a Web application.

JobId: The configuration object identifier of the job definition.

JobTitle: The title of the job definition.

ServerName: The name of the protocol client where the job instance is executed.

Status: The Job Status Type of the job instance.

StartTime: The datetime value when the job instance began execution.

CurrentTarget: The number of target instances that have been processed.

TargetCount: The total number of target instances the job instance is scheduled to process.

CurrentTargetPercentDone: The percentage of processing finished for the current target instance. If it is unknown NULL must be returned. Otherwise, the value returned MUST be an integer between "0" and "100" including the bounds.

3.1.5.31 proc_GetTimerTargetInstance

The **proc_GetTimerTargetInstance** stored procedure is called to determine if the specified target instance exists for a job definition. When no **TargetInstance** is specified, **proc_GetTimerTargetInstance** determines whether the input job definition has any **TargetInstances** associated with it. The stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_GetTimerTargetInstance (
    @JobId                  uniqueidentifier,
    @TargetInstanceId       uniqueidentifier,
    @Exists                 bit OUTPUT
);
```

@JobId: This is the configuration object identifier of a job definition.

@TargetInstanceId: This is either NULL or the GUID of a target instance associated with the job definition.

@Exists: This MUST be a bit indicating whether the specified target instance exists for the job definition. The value returned MUST be in the following table:

@TargetInstanceID	@Exists Value	Description
target instance GUID	0	The specified target instance does not exist for the job definition.
target instance GUID	1	The specified target instance exists for the job definition.
NULL	0	There are no target instances associated with the input job definition.
NULL	1	There are target instances associated with the input job definition.

Return Values: The proc_GetTimerTargetInstance stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The stored procedure MUST NOT return a result set.

3.1.5.32 proc_markForDeletionEmailEnabledList

The **proc_markForDeletionEmailEnabledList** stored procedure is called to mark as deleted the e-mail enabled list with the specified site collection identifier, site identifier, and list identifier. The stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_markForDeletionEmailEnabledList(
    @SiteId           uniqueidentifier,
    @WebId            uniqueidentifier,
    @ListId           uniqueidentifier
);
```

@SiteId: The site collection identifier of the site collection of the e-mail enabled list item to be marked for deletion.

@WebId: The site identifier of the site of the e-mail enabled list item to delete.

@ListId: The list identifier of the e-mail enabled list item to delete.

Return Values: The stored procedure returns an integer return code which MUST be 0.

The stored procedure MUST NOT return any result sets.

3.1.5.33 proc_markForDeletionEmailEnabledListsBySite

The proc_markForDeletionEmailEnabledListsBySite stored procedure is called to mark all e-mail enabled list items with the specified site collection identifier as deleted. The **proc_markForDeletionEmailEnabledListsBySite** is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_markForDeletionEmailEnabledListsBySite(
    @SiteId            uniqueidentifier
);
```

@SiteId: The site collection identifier of the site collection of the e-mail enabled list items to be marked for deletion.

Return Values: The stored procedure returns an integer return code which MUST be 0.

The stored procedure MUST NOT return any result sets.

3.1.5.34 proc_markForDeletionEmailEnabledListsByWeb

The **proc_markForDeletionEmailEnabledListsByWeb** stored procedure is called to mark all e-mail enabled list items with the specified site collection identifier and site identifier as deleted. The **proc_markForDeletionEmailEnabledsByWeb** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_markForDeletionEmailEnabledListsByWeb(
    @SiteId            uniqueidentifier,
    @WebId             uniqueidentifier
);
```

@SiteId: The site collection identifier of the site collection of the e-mail enabled list items to be marked for deletion.

@WebId: The site identifier of the site of the e-mail enabled list items to delete.

Return Values: The stored procedure returns an integer return code which MUST be 0.

The **proc_markForDeletionEmailEnabledListsByWeb** stored procedure MUST NOT return any result sets.

3.1.5.35 proc_putClass

The **proc_putClass** stored procedure is called to store a class. The **proc_putClass** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_putClass(
    @Id                uniqueidentifier,
    @BaseClassId       uniqueidentifier,
    @FullName          nvarchar(256)
);
```

@Id: Contains the ClassId of the class.

@BaseClassId: Contains the BaseClassId of the class.

@FullName: Contains an identifier that can be used by an application to associate a ClassId with a human- or machine-readable string. @FullName MUST be specified and MUST NOT be DBNull.

Return Values: The **proc_putClass** stored procedure returns an integer return code which MUST be listed in the following table:

Value	Description
0	Successful execution.
1	The class information of the GUID associated with **@BaseClassId** was not found.

The **proc_putClass** stored procedure MUST NOT return any result sets.

3.1.5.36 proc_putDependency

The **proc_putDependency** stored procedure is called to store a dependency between two configuration objects. The **proc_putDependency** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_putDependency(
    @ObjectId           uniqueidentifier,
    @DependantId        uniqueidentifier
);
```

@ObjectId: The identifier of the configuration object **that depends on another configuration object**.

@DependantId: The identifier of the configuration object on which the configuration object referenced by @ObjectId depends.

Return Values: The **proc_putDependency** stored procedure returns an integer return code which MUST be 0, which represents successful execution.

The **proc_putDependency** stored procedure MUST NOT return a result set.

3.1.5.37 proc_putDistributionListToDelete

The **proc_putDistributionListToDelete** stored procedure is called to mark a distribution list as deleted. The stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_putDistributionListToDelete(
    @Alias             nvarchar(128)
);
```

@Alias: The e-mail alias of the distribution list to be marked for deletion.

Return Values: The stored procedure returns an integer return code which MUST be 0.

The stored procedure MUST NOT return any result sets.

3.1.5.38 proc_putEmailEnabledList

The **proc_putEmailEnabledlist** stored procedure is called to add an existing list to the e-mail enabled list collection, or to update the e-mail alias of an existing element in the e-mail enabled list collection.

The **proc_putEmailEnabledList** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_putEmailEnabledList(
    @Alias                   nvarchar(128),
    @SiteId                  uniqueidentifier,
    @WebId                   uniqueidentifier,
    @ListId                  uniqueidentifier
);
```

@Alias: The e-mail alias of the e-mail enabled list.

@SiteId: The site collection identifier of the site collection containing the site.

@WebId: The site identifier of the site which contains the list.

@ListId: The list identifier of the list. If the list specified by the **@SiteId** parameter, **@WebId** parameter and **@ListId** parameter is already in the e-mail enabled list collection, this procedure updates the e-mail alias of this e-mail enabled list to match the **@Alias** parameter. If the list specified by the **@SiteId** parameter, **@WebId** parameter and **@ListId** parameter is not an e-mail enabled list, it will be added to the e-mail enabled list collection using the specified **@Alias** parameter for e-mail address.

Return Values: The **proc_putEmailEnabledList** stored procedure MUST return an integer return code which is specified in the following table.

Value	Description
0	Successful execution.
1	Error: The specified alias is already in use in the e-mail enabled list collection.

The stored procedure MUST NOT returns any result sets.

3.1.5.39 proc_putFileSegment

The **proc_putFileSegment** stored procedure is called to write a file in chunks. The **proc_putFileSegment** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_putFileSegment(
    @Pointer                 binary(16),
    @Offset                  int,
    @Bytes                   image
)
```

@Pointer: The address of the location of the file, being written. This value MUST be a TEXTPTR which is returned from a previous execution of the **proc_getFilePointer** stored procedure. If **@Pointer** is not NULL, it MUST contain a valid handle of the type returned by the TEXTPTR T-SQL function specified in [MSDN-TSQL-Ref].

@Offset: The value that determines the relative location in bytes, with respect to the location referenced by **@Pointer**.

@Bytes: The actual data being written at the location referenced by **@Offset** in the file referenced by **@Pointer**.

Return Values: The **proc_putFileSegment** stored procedure returns an integer return code which MUST be listed in the following table.

Value	Description
0	Successful execution.
1	Invalid data at location referenced by **@Pointer**.

The **proc_putFileSegment** stored procedure MUST NOT return a result set.

3.1.5.40 proc_putObject

The **proc_putObject** stored procedure is called to store a configuration object. The **proc_putObject** stored procedure is called to store new and update existing configuration objects. The **proc_putObject** stored procedure is specified using T-SQL syntax, as follows.

```
PROCEDURE proc_putObject (
    @Id                      uniqueidentifier,
    @ParentId                uniqueidentifier,
    @ClassId                 uniqueidentifier,
    @Name                    nvarchar(128),
    @Status                  int,
    @Version                 rowversion,
    @Properties              ntext,
    @ExistingObject          uniqueidentifier output,
    @NewVersion              rowversion output
);
```

@Id: The identifier of the configuration object **being stored**. If a new configuration object is being stored, the protocol client MUST generate a new configuration object identifier and pass its value in this parameter. If an existing configuration object is being updated, the value of this parameter MUST correspond to a configuration object which has already been stored in the database.

@ParentId: The **ParentId** of the configuration object. The value of this parameter MUST correspond to the identifier of a configuration object **which has been previously stored in the database.**

@ClassId: The **ClassId** of the configuration object.

@Name: The name of the configuration object.

@Status: The status of the configuration object.

@Version: The **version of** the configuration object. If a new configuration object **is being stored in this database for the first time, the protocol client MUST pass DBNull for the value of @Version.** Otherwise, the protocol client MUST pass the value of **version** returned from the previous call of the **proc_getObject** stored procedure as specified in [MS-WSSFO] (Section 3.1.4.31) or the **proc_getNewObjects** stored procedure which was used to obtain the configuration object **being updated.**

@Properties: The properties of the configuration object. This value MUST NOT be NULL.

@ExistingObject: If the protocol client passes **DBNull as the value of @Version** and the configuration database already contains a configuration object with the specified name, parent, and **ClassId** but with a different identifier, the protocol server MUST set **@ExistingObject** to the configuration object identifier of the configuration object**.**

@NewVersion: Upon successful execution of the proc_putObject stored procedure, the server MUST set @Version to a new, higher configuration object version.

Return Values: The **proc_putObject** stored procedure returns an integer return code which MUST be listed in the following table:

Value	Description
0	Successful execution.
1	Returned if the protocol client passed in value other than DBNull for **@Version**, but a configuration object with the specified identifier was not found.
2	Returned if a GUID value is passed to **@ClassId** without having been previously passed to the **@Id** parameter of the **proc_putClass** stored procedure.
3	Returned if a failure occurs when trying to create a new configuration object.
4	Returned when the caller tries to update an existing configuration object with a **version value that is different than what is referenced by @Version**.
5	Returned if a failure occurs when updating an existing configuration object.
6	Returned if a failure occurs when setting the value of **@NewVersion**.
7	Returned if the class referenced by **@ClassId** does not refer to the same class, described in the [XML Snippet] referenced by **@Properties**.
8	Returned when the **proc_putObject** stored procedure is called with **@Version** set to DbNull, **@Id** set to a new GUID, and with **@ClassId, @ParentId, and @Name set to values matching the ClassId, ParentId, and Name values of a** configuration object that is already in the configuration database.

Value	Description
9	Returned when **@Version** is **DBNull and @Id matches the identifier of an existing** configuration object.

The **proc_putObject** stored procedure MUST NOT return a result set.

3.1.5.41 proc_putPendingDistributionList

The **proc_putPendingDistributionList** stored procedure is called to add a new distribution list to the set of distribution lists which have an operation pending. The stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_putPendingDistributionList(
    @SiteId           uniqueidentifier,
    @WebId            uniqueidentifier,
    @GroupName        nvarchar(255),
    @ModifiedBy       nvarchar(255)
);
```

@SiteId: The site collection identifier of the site collection to which the distribution list belongs.

@WebId: The site identifier of the site to which the distribution list belongs.

@GroupName: The string name of the permission level to which the distribution list belongs.

@ModifiedBy: The string name containing the login name of the user whose distribution list operation is pending.

Return Values: The stored procedure returns an integer return code which MUST be 0.

The **proc_putPendingDistributionList** stored procedure MUST return a single result set as follows:

3.1.5.41.1 PutPendingDistributionList Result Set

The **PutPendingDistributionList** result set MUST contain only 1 row when **@SiteId @WebId** and **@GroupName** corresponds to a distribution list which has not been approved yet. If no such list exists, the PutPendingDistributionList result set MUST be returned and MUST contain 0 rows. The PutPendingDistributionList result set is specified using T-SQL syntax, as follows:

```
{PendingListRegistered}       int;
```

PendingListRegistered: Must contain the value 0 if the result set is returned.

3.1.5.42 proc_putSiteMap

The **proc_putSiteMap** stored procedure is called to create a new reference to a site collection. The **proc_putSiteMap** stored procedure is specified using T-SQL syntax, as follows.

```
PROCEDUREproc_putSiteMap(
    @ApplicationId                uniqueidentifier,
    @DatabaseId                   uniqueidentifier,
    @SiteId                       uniqueidentifier,
      @Path                           nvarchar(128),
      @Pairing                        tinyint,
      @RedirectUrl                    nvarchar(512),
      @HostHeaderIsSiteName           bit,
      @CurrentDatabaseSiteCount       int output
  );
```

@ApplicationId*:* Contains the configuration object identifier of the Web application which contains the site collection with the site collection identifier equal to **@SiteId***.*

@DatabaseId: Contains the configuration object identifier of the content database which contains the site collection identifier equal to **@SiteId***.*

@SiteID:

@Path: If the site collection referenced by **@SiteID** uses a host header site identifier, then **@Path** is the name of the site collection. Otherwise, **@Path** contains new server-relative URL for the site collection. This parameter MUST NOT be NULL.

@Pairing: Indicates whether the site collection was upgraded from previous version. This parameter MUST be one of the values in the following table.

Value	Description
0	This site collection was NOT upgraded from the previous version.
1	This site collection was upgraded from the previous version.

@RedirectUrl: Contains the server-relative URL for the site collection to redirect to. This parameter MUST be specified when **@Pairing** is set to 1.

@HostHeaderIsSiteName: Indicates whether the site collection being stored uses a host header site identifier. This parameter MUST be one of the values in the following table.

Value	Description
0	This site collection does NOT use a host header site identifier.
1	This site collection does use a host header site identifier.

@CurrentDatabaseSiteCount: Returns the number of the site collections in the same content database as identified by **@DatabaseId**. If the **proc_putSiteMap** stored procedure executes successfully, **@CurrentDatabaseSiteCount** MUST contain the total number of site collections in the content database. If the **proc_putSiteMap** stored procedure executes unsuccessfully, **@CurrentDatabaseSiteCount** MUST contain the input value.

Return Values: The **proc_putSiteMap** stored procedure returns an integer return code which MUST be listed in the following table.

Value	Description
0	Successful execution.
1	Failed to create the site collection

If the **proc_putSiteMap** stored procedure fails, it MUST NOT return a result set. If **proc_putSiteMap** executes successfully, it MUST return a single result set as follows:

3.1.5.42.1 SiteId Result Set

SiteId result set returns the GUID of the new site collection. The **SiteId** result set is specified using T-SQL syntax, as follows:

```
{SiteId}              uniqueidentifier;
```

SiteId: Contains the value of **@SiteId**.

3.1.5.43 proc_RefreshAllTimerLocks

The **proc_RefreshAllTimerLocks** stored procedure is called to refresh all job locks held by a specified protocol client. The stored procedure is specified using T-SQL syntax, as follows.

```
PROCEDURE proc_RefreshAllTimerLocks(
    @ServerName        nvarchar(128)
);
```

@ServerName: The name of the protocol client holding the job lock(s).

Return Values:

The **proc_RefreshAllTimerLocks** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_RefreshAllTimerLocks** stored procedure MUST NOT return a result set.

3.1.5.44 proc_RenameAllTimerLocksAndRunningJobs

The **proc_RenameAllTimerLocksAndRunningJobs** stored procedure is called to update the **Name** of a specified protocol client in the database tables that store timer job locks and job instances. The stored procedure is specified using T-SQL syntax, as follows.

```
PROCEDURE proc_RenameAllTimerLocksAndRunningJobs(
    @OldServerName            nvarchar(128),
    @NewServerName            nvarchar(128)
);
```

@OldServerName: The old name of the protocol client being renamed.

@ NewServerName: The new name of the protocol client being renamed.

Return Values:

The **proc_RenameAllTimerLocksAndRunningJobs** stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

The **proc_RenameAllTimerLocksAndRunningJobs** stored procedure MUST NOT return a result set.

3.1.5.45 proc_renameSiteMap

The **proc_renameSiteMap** stored procedure is called to update the server-relative URL for the specified site collection. The **proc_renameSiteMap** stored procedure is specified using T-SQL syntax, as follows.

```
PROCEDUREproc_renameSiteMap(
    @SiteId                 uniqueidentifier,
    @Path                   nvarchar(128)
);
```

@SiteId: Contains the GUID identifying the site collection to be updated.

@Path: If the site collection referenced by @SiteID uses a host header site identifier, then **@Path** is the name of the site collection. Otherwise, **@Path** contains new server-relative URL for the site collection. This parameter MUST NOT be NULL.

Return Values: The **proc_renameSiteMap** stored procedure MUST return an integer return code which MUST be 0.

The **proc_renameSiteMap** stored procedure MUST NOT return a result set.

3.1.5.46 proc_startTimerRunningJob

The **proc_StartTimerRunningJob** stored procedure is called to start a job instance. The **proc_StartTimerRunningJob** stored procedure is specified using T-SQL syntax, as follows.

```
PROCEDURE proc_StartTimerRunningJob (
    @ServiceId              uniqueidentifier,
    @VirtualServerId        uniqueidentifier,
    @JobId                  uniqueidentifier,
    @JobTitle               nvarchar(255),
    @ServerName             nvarchar(128),
    @TargetCount            int
);
```

@ServiceId: The configuration object identifier of the service associated with the job definition**.**

@VirtualServerId: The configuration object identifier of the Web application associated with the job definition. This MUST be NULL when the job definition is associated only with a service.

@JobId: The configuration object identifier of the job definition.

@JobTitle: The title of the job definition.

@ServerName: The name of the client where the job instance is executed.

@TargetCount: The number of target instances the job instance is scheduled to process.

Return Values: The **proc_StartTimerRunningJob** stored procedure returns an integer return code which MUST be in the following table.

Value	Description
0	Successful execution.

Value	Description
31	If the specified job instance cannot be started

The **proc_StartTimerRunningJob** stored procedure MUST not return a result set.

3.1.5.47 proc_UpdateTimerRunningJobProgress

The **proc_UpdateTimerRunningJobProgress** stored procedure is called to update the progress of the specified job instance**.** The **proc_UpdateTimerRunningJobProgress** stored procedure is specified using T-SQL syntax, as follows:

```
PROCEDURE proc_UpdateTimerRunningJobProgress (
    @ServiceId                      uniqueidentifier,
    @VirtualServerId                uniqueidentifier,
    @JobId                          uniqueidentifier,
    @ServerName                     nvarchar(128),
    @CurrentTargetPercentDone       int
);
```

@ServiceId: The configuration object identifier of the service associated with the job definition.

@VirtualServerId: The configuration object identifier of the Web application associated with the job definition. This MUST be NULL when the job definition is associated only with a service.

@JobId: The configuration object identifier of the job definition.

@ServerName: The name of the protocol client where the job instance is executed.

@CurrentTargetPercentDone: The percentage of processing that has been finished by the job instance for the current target instance. The scope of the value should be 0-100.

Return Values: The **proc_UpdateTimerRunningJobProgress** stored procedure returns an integer return code which MUST be in the following table.

Value	Description
0	Successful execution.
31	If there are no updates to the progress of the specified job instance**.**

The **proc_UpdateTimerRunningJobProgress** stored procedure MUST not return a result set.

3.1.5.48 proc_UpdateTimerRunningJobTarget

The **proc_UpdateTimerRunningJobTarget** stored procedure is called to update the target instance for the specified job instance. The **proc_UpdateTimerRunningJobTarget** stored procedure is specified using T-SQL syntax, as follows.

```
PROCEDURE proc_UpdateTimerRunningJobTarget (
    @ServiceId                uniqueidentifier,
    @VirtualServerId          uniqueidentifier,
    @JobId                    uniqueidentifier,
    @ServerName               nvarchar(128),
    @CurrentTarget            int
);
```

@ServiceId: The configuration object identifier of the service associated with the job definition.

@VirtualServerId: The configuration object identifier of the Web application associated with the job definition. This MUST be NULL when the job definition is associated only with a service.

@JobId: The configuration object identifier of the job definition.

@ServerName: The name of the client where the job instance is executed.

@CurrentTarget: The number of target instances that have been processed.

Return Values: The **proc_UpdateTimerRunningJobTarget** stored procedure returns an integer return code which MUST be in the following table.

Value	Description
0	Successful execution.
31	If there are no updates to the target instance of the specified job instance.

The **proc_UpdateTimerRunningJobTarget** stored procedure MUST not return a result set.

3.1.6 Timer Events

None.

3.1.7 Other Local Events

None.

4 Protocol Examples

4.1 Delete E-mail-Enabled Lists from a Site Collection

If SharePoint® adds e-mail aliases to an **Active Directory** when they become associated with SharePoint lists, then it needs to remove those e-mail aliases when they are no longer in use. When many e-mail aliases are involved, this can be a time consuming operation, so the protocol server tracks e-mail aliases which are no longer mapped to lists, but which still need to be removed from the Active Directory. This example illustrates the protocol operations needed to remove all of the e-mail aliases used by e-mail enabled lists within a specified site collection. An existing connection to the configuration database using lower-level protocols is assumed.

4.1.1 Mark E-mail-Enabled Lists as Deleted

The example begins by marking all of the e-mail enabled lists for a specified site collection as deleted by calling the **proc_markForDeletionEmailEnabledListsBySite** stored procedure with the specified site collection identifier.

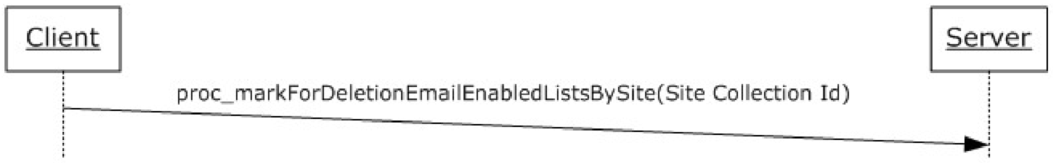
Client
Server
proc_markForDeletionEmailEnabledListsBySite(Site Collection Id)

Figure 2: Marking the e-mail enabled lists as deleted

4.1.2 Retrieve E-mail Aliases Marked as Deleted

The e-mail aliases marked for deletion are retrieved at a later time by calling the **proc_getDeletedEmailAliases** stored procedure.

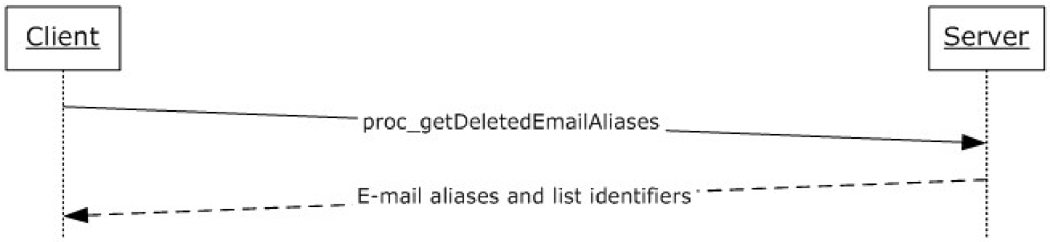
Client
Server
proc_getDeletedEmailAliases
E-mail aliases and list identifiers

Figure 3: Retrieving the e-mail aliases marked as deleted

This call returns e-mail aliases and list identifiers for all of the e-mailed aliases which have been marked for deletion. The protocol client can then perform any actions needed to remove these e-mail aliases from the Active Directory.

4.1.3 Remove E-mail-Enabled Lists

Once a specified e-mail alias has been deleted from the Active Directory, the e-mail enabled list is removed from the configuration database by calling the **proc_dropEmailEnabledListByAlias** stored procedure with its e-mail alias. Each e-mail alias removed requires a separate call.

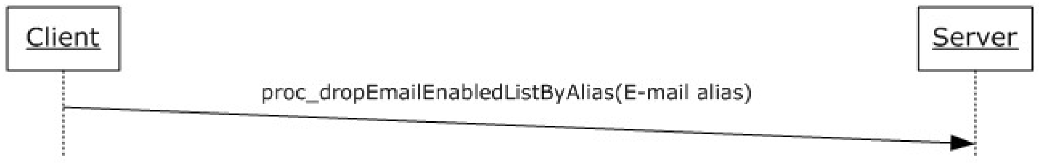

Figure 4: Removing the e-mail enabled lists

4.2 Pending Distribution Lists

Creation of distribution lists associated with permission levels can require approval, and the protocol server stores the list of permission levels whose associated distribution lists are still pending approval so that it can periodically check for that approval. An existing connection to the configuration database using lower-level protocols is assumed.

4.2.1 Add a Pending Distribution List

When creation of a distribution list associated with a permission level requires approval, the client adds it to the list of distribution lists with an operation pending by calling **proc_putPendingDistributionList** with the site collection id, site id, and name of the permission level along with the name of the user who has requested creation of the distribution list.

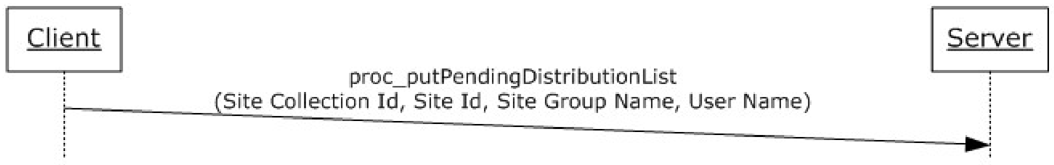
Client
Server
proc_putPendingDistributionList
(Site Collection Id, Site Id, Site Group Name, User Name)

Figure 5: Adding a pending distribution list

4.2.2 Retrieve Pending Distribution Lists

When the protocol client is ready to check for approval of its pending distribution lists, it retrieves the list of permission levels whose distribution lists require approval by calling the **proc_getPendingDistributionListsSinceVersion** stored procedure with row version equal to 0. This returns all of the distribution lists pending approval, along with their row versions.

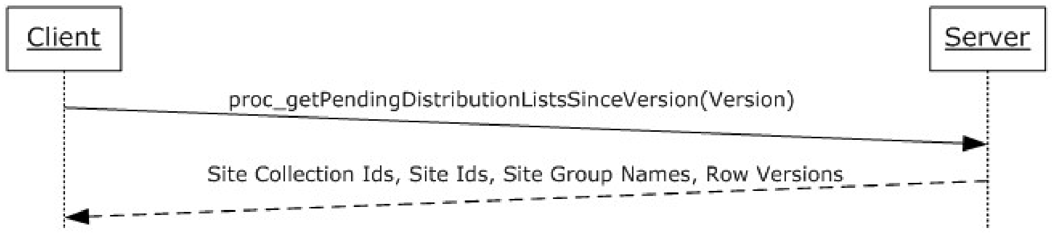
Client
Server
proc_getPendingDistributionListsSinceVersion(Version)
Site Collection Ids, Site Ids, Site Group Names, Row Versions

Figure 6: Retrieving the pending distribution lists

If the groups returned by this call have not yet been given approval, the protocol client can continue to check for approval, and it can request pending distribution lists which were added since the last time it retrieved the list by calling the **proc_getPendingDistributionListsSinceVersion** stored procedure with the largest row version previously returned to it.

4.2.3 Remove Pending Distribution Lists

Once creation of the distribution list associated with a permission level is no longer pending, the protocol client removes the distribution list from the list of pending distribution lists by calling the **proc_dropPendingDistribution** stored procedure.

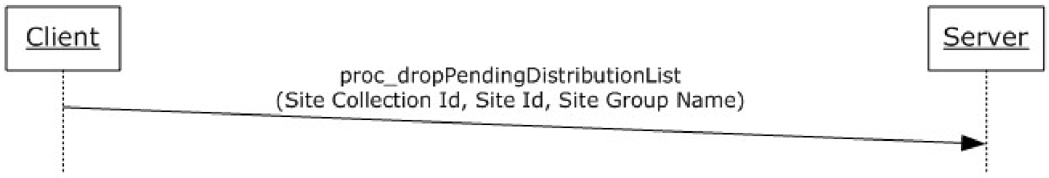
Client
Server
proc_dropPendingDistributionList
(Site Collection Id, Site Id, Site Group Name)

Figure 7: Removing the pending distribution lists

4.3 Run a Job Instance

This example illustrates the sequence of operations performed for a job instance, which is the execution of a job definition on one protocol client. In this example, the parent of the job definition is a Web application with three content databases.

4.3.1 Acquire a Database Lock

Our example job definition requires content database locks to ensure that only a single protocol client processes a content database at any one time. Multiple job instances for a job definition may execute concurrently on different clients if each protocol client has obtained different content database locks. A protocol client may attempt to obtain a content database lock by calling the **proc_GetTimerLock** stored procedure, as specified in [MS-WSSCADM].

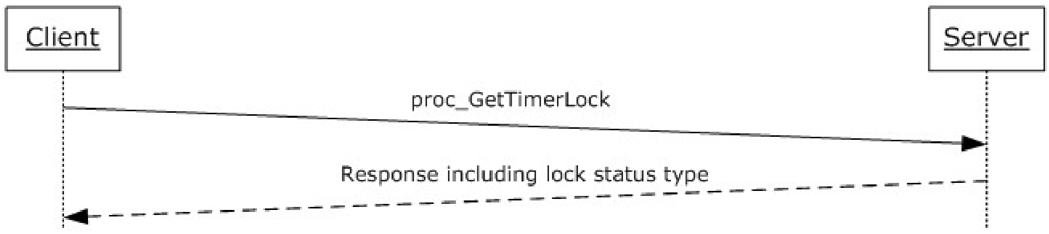

Figure 8: Acquiring locks

The lock status type indicates whether the protocol client successfully acquires the content database lock. For this example, the protocol client was able to successfully obtain locks for two of the three content databases specified by the Web application of the parent.

4.3.2 Create a Target Instance

A job instance can process several different resources during a single execution. A target instance provides a way to keep track of the job instance's execution progress. In our example, the job definition targets three content databases and the protocol client has obtained content database locks for two of those content databases. The job instance processes those two content databases during a single execution. A target instance for each content database can be created by calling the **proc_AddTimerTargetInstance** stored procedure.

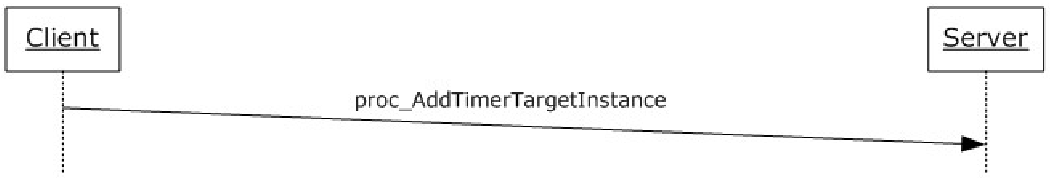

Figure 9: Target instances

4.3.3 Start a Job Instance

When a job is started using the **proc_StartTimerRunningJob** stored procedure, the **TargetCount** indicates how many target instances the job instance is scheduled to process. The **CurrentTarget** is a counter which indicates how many target instances have been processed. In our example, the protocol client has acquired two content database locks and processes two content databases represented by target instances. So initially the proc_StartTimerRunningJob stored procedure is called with **TargetCount** = 2.

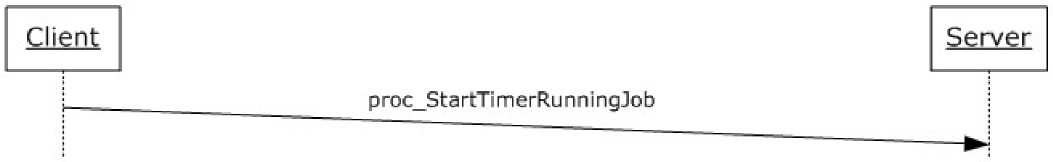

Figure 10: Starting a job instance

4.3.4 Update Job Progress

While the protocol client is processing each content database, progress can be reported as a percentage of the current target that has been processed. To update the current progress, the protocol client can periodically call the **Proc_UpdateTimerRunningJobProgress** stored procedure.

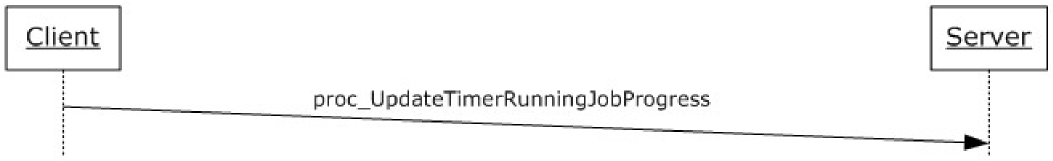

Figure 11: Updating job progress

4.3.5 Process Additional Target Instances

In our example, the protocol client has acquired two content database locks. Once the protocol client has finished processing a content database, processing may begin on the subsequent content database. To begin processing the next content database, the protocol client can call the **proc_UpdateTimerRunningJobTarget** stored procedure.

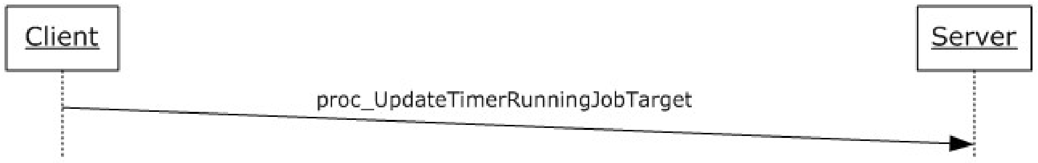

Figure 12: Processing additional target instances

4.3.6 Complete a Job Instance

Once the protocol client has finished processing the two content databases, the job instance may be completed by calling the **proc_CompleteTimerRunningJob** stored procedure.

Client
Server
proc_CompleteTimerRunningJob

Figure 13: Completing a job instance

4.4 File Storage and Retrieval

This example follows the sequence of calls need to store and retrieve a file from the configuration database. This is useful for files that are very large and might benefit from the ability to stream a segment of the file into or out of the database without needed to hold the entire file contents in memory.

4.4.1 File Storage

File Storage is a multi-step operation which begins when the protocol client passes the GUID of an existing configuration object to the **proc_GetFilePointer** stored procedure.

The **proc_GetFilePointer** stored procedure returns a TEXTPTR which is then immediately passed to the **proc_putFileSegment** stored procedure, along with some of the contents of the file and the offset of the location in the file to which are writing. In this example, the entire file is updated, so the **@Offset** parameter for the first call to **proc_putFileSegment** is 0. This file being stored in this example is 75000 bytes. To illustrate the chunking behavior, the value of the **@Bytes** parameter in the first call to **proc_putFileSegment** contains the first 50000 bytes of the file.

To store the remaining 25000 bytes of the file, the **proc_putFileSegment** stored procedure is called again. This time, the last 25000 bytes of the file are passed in **@Bytes** and the **@Offse**t is 50000.

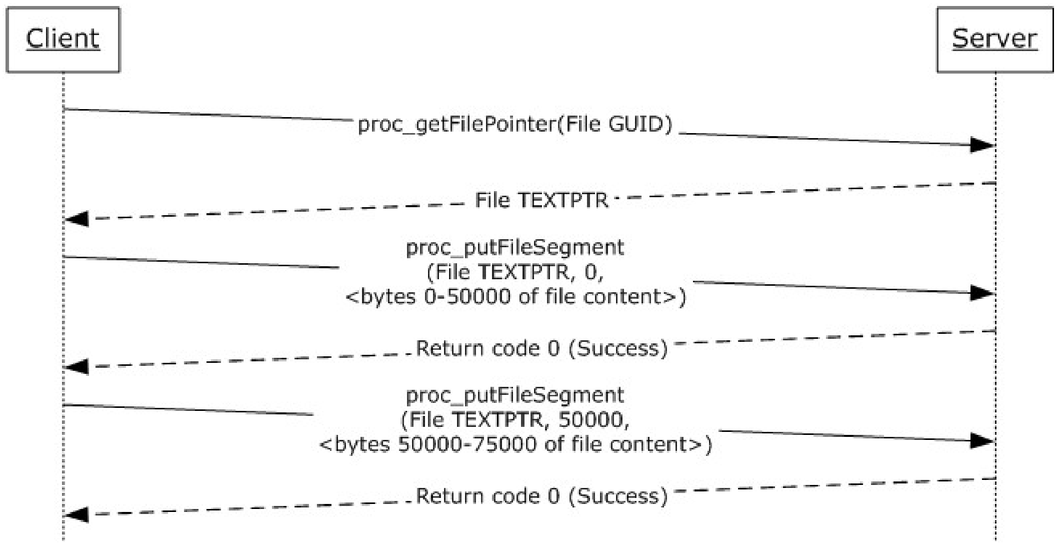
Client
Server
proc_getFilePointer(File GUID)
File TEXTPTR
proc_putFileSegment (File TEXTPTR, 0, <bytes 0-50000 of file content>)
Return code 0 (Success)
proc_putFileSegment (File TEXTPTR, 50000, <bytes 50000-75000 of file content>)
Return code 0 (Success)

Figure 14: File storage

4.4.2 File Retrieval

Once a file has been stored in the configuration database, its contents can be returned by a simple call to the **proc_getFile** stored procedure. The same GUID used during the call to the **proc_GetFilePointer** stored procedure is used here to retrieve the contents of that file.

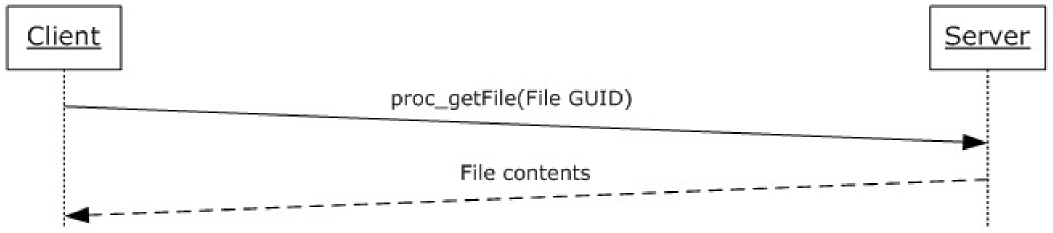

Figure 15: File retrieval

4.5 Shared Services Provider Connection String Lookup

Retrieve a shared services provider **connection string** from the configuration database as follows.

1. Retrieve the farm identifier as specified in the example in [MS-WSSFO] (Section 4.7.1).

2. Call the **proc_getObjectsByClass** stored procedure as specified in [MS-WSSFO] (Section 3.1.4.33) to retrieve the shared services provider configuration object.

   - Set the **@ParentId** parameter to the Farm Id returned in the result set from Step 1.

   - Set the **@ClassId** parameter to the Shared Services Provider **ClassId** value specified in Section 2.2.6.2.

   - Set the **@Name** parameter to the name of the Shared Services Provider to be retrieved.

3. Call the **proc_getObject** stored procedure as specified in [MS-WSSFO] (Section 3.1.4.31).

- Set the **@ Id** parameter to the value of the Shared Services Provider **SharedDatabaseId** property of the result set from Step 2 as specified in Section 2.2.6.2.

4. Call the **proc_getObject** stored procedure as specified in [MS-WSSFO] (Section 3.1.4.31).
   - Set the **@ Id** parameter to the value of the **ParentId** property of the result set from Step 3.
5. Call the **proc_getObject** stored procedure as specified in [MS-WSSFO] (Section 3.1.4.31).
   - Set the **@ Id** parameter to the value of the **ParentId** property of the result set from Step 4.

Compose the connection string as follows:

- Set the **Database Name** property to the value of **Name** in the result set from Step 3.
- Set the **Server** property to the value of **Name** in the result set from Step 5.
- Set the **Instance** property to the value of **Name** in the result set from Step 4, if specified.
- Set the **User Name** property to the value of **Username** in the result set from Step 3 as specified in Section 2.2.6.3 if **Username** is not null or empty, otherwise, set the **Trusted Connection** property to **true**.
- Set the **Password** property to the value of **Password** in the result set from Step 3 as specified in Section 2.2.6.3 if **Password** is not null or empty.

5 Security

5.1 Security Considerations for Implementers

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

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

The information in this specification is applicable to the following Microsoft products or supplemental software. References to product versions include released service packs:

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

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

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

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