Intellectual Property Rights Notice for Open Specifications Documentation

- **Technical Documentation.** Microsoft publishes Open Specifications documentation for protocols, file formats, languages, standards as well as overviews of the interaction among each of these technologies.

- **Copyrights.** This documentation is covered by Microsoft copyrights. Regardless of any other terms that are contained in the terms of use for the Microsoft website that hosts this documentation, you may make copies of it in order to develop implementations of the technologies described in the Open Specifications and may distribute portions of it in your implementations using these technologies or your documentation as necessary to properly document the implementation. You may also distribute in your implementation, with or without modification, any schema, IDL’s, or code samples that are included in the documentation. This permission also applies to any documents that are referenced in the Open Specifications.

- **No Trade Secrets.** Microsoft does not claim any trade secret rights in this documentation.

- **Patents.** Microsoft has patents that may cover your implementations of the technologies described in the Open Specifications. Neither this notice nor Microsoft’s delivery of the documentation grants any licenses under those or any other Microsoft patents. However, a given Open Specification may be covered by Microsoft Open Specification Promise or the Community Promise. If you would prefer a written license, or if the technologies described in the Open Specifications are not covered by the Open Specifications Promise or Community Promise, as applicable, patent licenses are available by contacting iplg@microsoft.com.

- **Trademarks.** The names of companies and products contained in this documentation may be covered by trademarks or similar intellectual property rights. This notice does not grant any licenses under those rights.

- **Fictitious Names.** The example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted in this documentation are fictitious. No association with any real company, organization, product, domain name, email address, logo, person, place, or event is intended or should be inferred.

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

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

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

## 1 Introduction

- Glossary ................................................................. 7
- References .............................................................. 7
  - Normative References ............................................... 8
  - Informative References ............................................ 8
- Protocol Overview (Synopsis) ...................................... 9
  - Configuration Object Management .............................. 9
  - File Storage ......................................................... 9
  - Timer Job Management ............................................ 9
  - E-mail-Enabled Lists .............................................. 9
  - Pending Distribution Lists ..................................... 9
- Relationship to Other Protocols .................................. 10
- Prerequisites/Preconditions ....................................... 10
- Applicability Statement ............................................ 10
- Versioning and Capability Negotiation .......................... 10
- Vendor-Extensible Fields .......................................... 10
- Standards Assignments ............................................. 10

## 2 Messages

- Transport ........................................................................ 11
- Common Data Types ................................................... 11
  - Simple Data Types and Enumerations .......................... 11
  - Bit Fields and Flag Structures .................................. 11
    - Job Lock Type ..................................................... 11
    - Job Status Type .................................................. 11
    - Lock Status Type ................................................ 12
  - Binary Structures .................................................. 12
  - Result Sets ................................................................ 12
  - Configuration Object Classes .................................... 12
  - Configuration Object Properties ................................. 12
    - Timer Job Definition ............................................ 12
    - E-Mail-Enabled List .............................................. 13
    - Pending Distribution List ...................................... 13
  - Tables and Views .................................................... 14
  - XML Structures ..................................................... 14

## 3 Protocol Details

- Server Details .......................................................... 15
  - Abstract Data Model ............................................... 15
    - Configuration Object Management ............................ 15
    - Timer Job Management .......................................... 16
    - E-Mail-Enabled Lists ............................................. 17
    - Pending Distribution Lists .................................... 17
  - Timers ........................................................................ 17
  - Initialization .......................................................... 17
  - Higher-Layer Triggered Events .................................... 17
  - Message Processing Events and Sequencing Rules .......... 17
    - proc_AddTimerJobHistory ...................................... 17
    - proc_AddTimerLockForJob ..................................... 18
    - proc_AddTimerTargetInstance ................................ 19
3.1.5.4 proc_ClearProductVersions

3.1.5.5 proc_CompleteTimerRunningJob

3.1.5.6 proc_DeleteAllMarkedTimerLocks

3.1.5.7 proc_DeleteAllTimerLocksAndRunningJobs

3.1.5.8 proc_DeleteOldTimerJobHistory

3.1.5.9 proc_DeleteScheduledJob

3.1.5.10 proc_DeleteTimerLockForJob

3.1.5.11 proc_DeleteTimerRunningJobs

3.1.5.12 proc_DeleteTimerTargetInstance

3.1.5.13 proc_DeleteTimerTargetInstances

3.1.5.14 proc_dropClass

3.1.5.14.1 DropClass Result Set

3.1.5.15 proc_dropEmailEnabledList

3.1.5.16 proc_dropEmailEnabledListByAlias

3.1.5.17 proc_dropEmailEnabledListsByWeb

3.1.5.18 proc_DropObject

3.1.5.19 proc_dropPendingDistributionList

3.1.5.20 proc_DropSiteMap

3.1.5.21 proc_getDeletedEmailAliases

3.1.5.21.1 DeletedEmailAliases Result Set

3.1.5.22 proc_GetDependentObjectsByBaseClass

3.1.5.22.1 Dependent Object Ids Result Set

3.1.5.23 proc_getEmailAddressListByAlias

3.1.5.23.1 EmailAddressListByAlias Result Set

3.1.5.24 proc_getFile

3.1.5.24.1 File Result Set

3.1.5.25 proc_getNewObjects

3.1.5.25.1 Last Update Result Set

3.1.5.25.2 Modified Objects Result Set

3.1.5.25.3 Dependencies Result Set

3.1.5.25.4 Tombstones Result Set

3.1.5.26 proc_getPendingDistributionListsSinceVersion

3.1.5.26.1 PendingDistributionList Result Set

3.1.5.27 proc_GetProductVersions

3.1.5.27.1 Get Product Versions Result Set

3.1.5.28 proc_getSiteBestMatch

3.1.5.29 proc_getSiteCount

3.1.5.29.1 SiteCount Result Set

3.1.5.30 proc_getSiteIdOfHostHeaderSite

3.1.5.30.1 ID Result Set

3.1.5.31 proc_getSiteIds

3.1.5.31.1 Get Site Ids Result Set

3.1.5.32 proc_getSiteNames

3.1.5.32.1 Path Result Set

3.1.5.33 proc_getSiteSubscriptionsInContentDB

3.1.5.33.1 Get Site Subscriptions In Content DB Result Set

3.1.5.34 proc_getSiteSubscriptionsInWebApplication

3.1.5.34.1 Get Site Subscriptions in Web Application Result Set

3.1.5.35 proc_getSiteSubscriptionSiteIdsInContentDatabase

3.1.5.35.1 Get Site Subscription Site Ids in Content Database Result Set

3.1.5.36 proc_getSiteSubscriptionSiteIdsInWebApplication

3.1.5.36.1 Get Site Subscription Site Ids In Web Application Result Set

3.1.5.37 proc_getSubscriptionIds
3.1.5.37.1 Get Subscription Ids Result Set ........................................ 44
3.1.5.38 proc_getSiteSubset ................................................................. 44
3.1.5.38.1 Site Result Set ...................................................................... 45
3.1.5.39 proc_getTemplate ................................................................. 45
3.1.5.39.1 Template Result Set ......................................................... 46
3.1.5.40 proc_GetTimerJobHistory ....................................................... 46
3.1.5.40.1 Timer Job History Result Set ........................................... 47
3.1.5.41 proc_GetTimerJobLastRunTime ................................................ 48
3.1.5.42 proc_GetTimerRunningJobs .................................................... 48
3.1.5.42.1 Job Status Result Set ...................................................... 49
3.1.5.43 proc_GetTimerScheduledJobs ................................................ 50
3.1.5.43.1 Timer Scheduled Job Result Set ....................................... 50
3.1.5.44 proc_GetTimerTargetInstance ................................................ 51
3.1.5.45 proc_GetTimerTargetInstanceState ......................................... 51
3.1.5.45.1 Timer Target Instance State Result Set ......................... 52
3.1.5.46 proc_markForDeletionEmailEnabledList ........................................ 52
3.1.5.47 proc_markForDeletionEmailEnabledListsBySite .......................... 52
3.1.5.48 proc_markForDeletionEmailEnabledListsByWeb ....................... 53
3.1.5.49 proc_putClass ................................................................... 53
3.1.5.50 proc_putDependency .............................................................. 54
3.1.5.51 proc_putDistributionListToDelete ........................................ 54
3.1.5.52 proc_putEmailEnabledList ...................................................... 55
3.1.5.53 proc_putFileSegment ............................................................. 55
3.1.5.54 proc_putObject .................................................................. 56
3.1.5.55 proc_putPendingDistributionList ......................................... 58
3.1.5.55.1 PutPendingDistributionList Result Set .............................. 58
3.1.5.56 proc_putSiteMap ................................................................. 58
3.1.5.56.1 SiteId Result Set ............................................................... 60
3.1.5.57 procRefreshAllTimerLocks ..................................................... 60
3.1.5.58 proc_RegisterProductVersion .............................................. 60
3.1.5.59 proc_renameSiteMap ............................................................ 61
3.1.5.60 proc_ScheduleTimerJob ......................................................... 62
3.1.5.61 proc_setSubscription .......................................................... 63
3.1.5.62 proc_SiteExists ................................................................. 63
3.1.5.63 proc_StartTimerRunningJob ................................................. 63
3.1.5.64 proc_UpdateTimerRunningJobProgress ................................... 64
3.1.5.65 proc_UpdateTimerRunningJobStatus .................................... 65
3.1.5.66 proc_UpdateTimerRunningJobTarget .................................... 66
3.1.5.67 proc_UpdateTimerTargetInstanceState ................................. 67
3.1.5.68 proc_hideSiteMap ............................................................... 67
3.1.5.69 proc_unhideSiteMap ............................................................ 67
3.1.5.70 proc_dropSiteMap2 ............................................................. 68
3.1.5.71 proc_getDeletedSitesByUrl .................................................. 68
3.1.5.71.1 Site Result Set ................................................................. 69
3.1.5.72 proc_getDeletedSitesByQuery .............................................. 70
3.1.5.72.1 Site Result Set ................................................................. 70
3.1.5.73 proc_getDeletedSitesBySiteId .............................................. 71
3.1.5.73.1 Site Result Set ................................................................. 72
3.1.5.74 proc_getDeletedSite ............................................................ 73
3.1.5.74.1 Site Result Set ................................................................. 73
3.1.5.75 proc_putDeletedSiteMap ....................................................... 74
3.1.6 Timer Events ........................................................................... 75
3.1.7 Other Local Events .................................................................. 75
1 Introduction

The Windows SharePoint Services: Configuration Database Communications 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
class
GUID
Hypertext Transfer Protocol (HTTP)
object
Security Support Provider Interface (SSPI)
service

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

application server
back-end database server
class identifier (CLSID)
configuration database
configuration object
configuration object identifier
content database
content database lock
datetime
distribution list
e-mail alias
e-mail enabled list
empty GUID
farm
front-end Web server
host header
host header site identifier
job definition
job instance
job lock
list
list identifier
permission level
request identifier
result set
return code
row version
server-relative URL
site
site collection
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.


#### 1.2.2 Informative References

- [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

When 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 by using a timer service that runs on all protocol clients connected to a configuration database.

1.3.4 E-mail-Enabled Lists

Some services need to store data in a list that 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:

![Diagram showing relationship to other protocols]

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 Security Support Provider Interface (SSPI) and SQL authentication with the protocol server role specified 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

None.

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>None</td>
</tr>
<tr>
<td>1</td>
<td>content database</td>
</tr>
<tr>
<td>2</td>
<td>Job</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The job definition does not use locking between protocol clients.</td>
</tr>
<tr>
<td></td>
<td>Only one job instance is permitted to process a content database at any one time.</td>
</tr>
<tr>
<td></td>
<td>Only one job instance for the job definition is permitted to execute at any one time.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The job instance has been scheduled.</td>
</tr>
<tr>
<td>1</td>
<td>The job instance has been initialized and is currently executing.</td>
</tr>
<tr>
<td>2</td>
<td>The job instance was successfully executed.</td>
</tr>
<tr>
<td>3</td>
<td>An error occurred while executing the job instance.</td>
</tr>
<tr>
<td>4</td>
<td>An error occurred during execution, but the job instance has been scheduled to retry the execution.</td>
</tr>
<tr>
<td>5</td>
<td>The job instance was interrupted before execution could complete and is not currently scheduled for another execution.</td>
</tr>
<tr>
<td>6</td>
<td>The job instance is in the process of pausing.</td>
</tr>
</tbody>
</table>
### 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:

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

### 2.2.3 Binary Structures

None.

### 2.2.4 Result Sets

None.

### 2.2.5 Configuration Object Classes

The following configuration object class identifiers (CLSID) are used during the execution of this protocol in addition to the configuration object class identifiers (CLSID) specified in [MS-WSSFO2] section 2.2.6.1.1 [MS-WSSFO2].

<table>
<thead>
<tr>
<th>Class</th>
<th>Class Identifier (CLSID)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job Definition</td>
<td>3F9F635F-0036-42fe-9C2D-3284162732DB</td>
</tr>
<tr>
<td>Service</td>
<td>DACA2A15-B9B5-43da-BEA3-6B75FBE3A883</td>
</tr>
</tbody>
</table>

### 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 (CLSID) of the service or the class derived from the service as specified in Section 2.2.5 or a Web application as specified in in [MS-WSSFO2] section 2.2.6.1.1.
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 T-SQL format:

<table>
<thead>
<tr>
<th>Alias</th>
<th>nvarchar(128) NOT NULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>SiteId</td>
<td>uniqueidentifier NOT NULL</td>
</tr>
<tr>
<td>WebId</td>
<td>uniqueidentifier NOT NULL</td>
</tr>
<tr>
<td>ListId</td>
<td>uniqueidentifier NOT NULL</td>
</tr>
<tr>
<td>Deleted</td>
<td>bit NOT NULL DEFAULT((0))</td>
</tr>
</tbody>
</table>

**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 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.

**Deleted:** This value is 1 if the e-mail alias has been marked for future deletion, and 0 otherwise.

### 2.2.6.3 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:

<table>
<thead>
<tr>
<th>SiteId</th>
<th>uniqueidentifier NOT NULL</th>
</tr>
</thead>
<tbody>
<tr>
<td>WebId</td>
<td>uniqueidentifier NOT NULL</td>
</tr>
</tbody>
</table>

---

[MS-WSSCFGD2] — v20120630
Windows SharePoint Services: Configuration Database Communications Version 2 Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
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-WSSFD2] section 2.2.5.1.

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

As in 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 Scheduling

A scheduled job is a record that indicates when a job instance will be run by a protocol client. The protocol client creates a scheduled job entry for each job instance when it is scheduled for execution.

Job Prerequisites

A job definition can optionally require a content database lock or job lock. 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 the protocol client indicates how many target instances the job instance is scheduled to process and keeps a counter which indicates how many target instances have been finished.

Job Status

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

Job History

After a job instance has been run, the protocol client creates a record of the execution. The job history entries provide information about the execution of a job instance. A list of job instances run by a protocol client in the past can be obtained by retrieving these job history entries.
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 such as renaming a permission level, modifying its description or creating an external group 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. There can be only one operation that’s pending per permission level. 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 determine whether 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

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 can optionally have a length value in brackets and can optionally have a default value indicated by an equals sign followed by the default value.

3.1.5.1 proc_AddTimerJobHistory

The proc_AddTimerJobHistory stored procedure is called to record the processing of a target instance during the execution of a job instance. The stored procedure is specified by the following T-SQL syntax.

```
PROCEDURE proc_AddTimerJobHistory (  
@ServiceId            uniqueidentifier,  
@WebApplicationId     uniqueidentifier,  
@WebApplicationName   nvarchar(255),  
@JobId                uniqueidentifier,  
@JobTitle             nvarchar(255),
```

[MS-WSSCFGD2] — v20120630
Windows SharePoint Services: Configuration Database Communications Version 2 Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
@ServerId: The configuration object identifier of the service associated with the job definition.

@WebApplicationId: 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.

@WebApplicationName: The name 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.

@ServerId: The configuration object identifier of the protocol client where the job instance was run.

@ServerName: The name of the protocol client where the job instance was run.

@Status: The Job Status Type of the job instance.

@StartTime: The datetime value when the job instance began running.

@EndTime: The datetime value when the job instance was finished.

@DatabaseName: The name of the processed database when a job instance uses a content database lock, otherwise NULL.

@ErrorMessage: The error message associated with an error that occurred while running the job instance, otherwise NULL.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_AddTimerJobHistory stored procedure MUST NOT return a result set.

3.1.5.2 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 by the following T-SQL syntax.

PROCEDURE proc_AddTimerLockForJob(
    @JobId                  uniqueidentifier,
    @ServerId               uniqueidentifier,
    @LockTimeout            int,
    @LockStatus             int OUTPUT,
    @LockExpiredServerId    uniqueidentifier OUTPUT,
    @RequestGuid            uniqueidentifier = null OUTPUT
);
@JobId: The configuration object identifier of the job definition for which a job lock is requested.

@ServerId: The configuration object identifier 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 client already holds the lock for the specified JobId, whether it has expired or not.

@LockExpiredServerId: The configuration object identifier of the protocol client that holds an expired job lock which is overwritten. The output variable @LockExpiredServerId MUST be the identifier 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. Otherwise the output variable @LockExpiredServerId MUST be NULL.

@RequestGuid: The optional request identifier for the current request.

Return Code Values:

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>167</td>
<td>An error occurred while retrieving the state of the job lock.</td>
</tr>
</tbody>
</table>

Result Set: The proc_AddTimerLockForJob stored procedure MUST NOT return a result set.

3.1.5.3 proc_AddTargetInstance

The proc_AddTargetInstance 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 type of target instances depends on the Job Lock Type specified by a job definition:

<table>
<thead>
<tr>
<th>Job Lock Type</th>
<th>Target Instance</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>A target instance for each protocol server where the service associated with the job definition has been provisioned.</td>
</tr>
<tr>
<td>Job</td>
<td>One target instance for the job definition.</td>
</tr>
<tr>
<td>Content Database</td>
<td>A target instance for each content database contained in the Web application referenced by the job definition.</td>
</tr>
</tbody>
</table>

The proc_AddTargetInstance stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_AddTargetInstance (  
```
@JobId: The configuration object identifier of the job definition for which a job lock is requested.

@TargetInstanceId: The configuration object identifier of a target instance associated with the job definition.

@RequestGuid: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The specified job definition exists.</td>
</tr>
<tr>
<td>31</td>
<td>The specified job definition does not exist, and so wasn’t associated with the specified target instance.</td>
</tr>
</tbody>
</table>

Result Set: The proc_AddTimerTargetInstance stored procedure MUST NOT return a result set.

3.1.5.4 proc_ClearProductVersions

The proc_ClearProductVersions stored procedure is called to remove all saved information about the installations on a server in the farm. The stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_ClearProductVersions (  
    @ServerId uniqueidentifier  
);  

@ServerId: Remove all installation information for the protocol client with this configuration object identifier.

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

Result Set: The proc_ClearProductVersions stored procedure MUST NOT return a result set.

3.1.5.5 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 by the following T-SQL syntax.

PROCEDURE proc_CompleteTimerRunningJob (  
    @ServiceId uniqueidentifier,  
    @VirtualServerId uniqueidentifier,  
    @JobId uniqueidentifier,  
    @ServerId uniqueidentifier,  
    @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.

@ServerId: The configuration object identifier of the protocol client where the job instance was executed.

@Status: The Job Status Type of the job instance. If the @Status parameter equals 6 or 7 then the stored procedure MUST update the Job Status Type for the job instance. Otherwise the running job instance MUST be removed.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_CompleteTimerRunningJob stored procedure MUST NOT return a result set.

### 3.1.5.6 proc_DeleteAllMarkedTimerLocks

The proc_DeleteAllMarkedTimerLocks stored procedure is called to delete all expired job locks marked for deletion. The stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_DeleteAllMarkedTimerLocks(
    @LockTimeout  int,
    @RequestGuid  uniqueidentifier = NULL OUTPUT
);
```

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

@RequestGuid: The optional request identifier for the current request.

Return Code Values:

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

Result Set: The proc_DeleteAllMarkedTimerLocks stored procedure MUST NOT return a result set.

### 3.1.5.7 proc_DeleteAllTimerLocksAndRunningJobs

The proc_DeleteAllTimerLocksAndRunningJobs stored procedure is called to delete job locks marked for deletion and job instances for a specified protocol client. The stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_DeleteAllTimerLocksAndRunningJobs(
);
```
@ServerId: The configuration object identifier of the protocol client holding the job lock(s) to be deleted.

@KeepPauseJobs: A flag which indicates whether to delete job instances that are pausing or have been paused as indicated by the Job Status Type. The value is a bit and MUST be in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Delete all job instances for the specified protocol client.</td>
</tr>
</tbody>
</table>
| 1     | Job instances with current Job Status Type equal to 6 MUST be set to Job Status Type 7 and MUST not be deleted.  
All remaining Job instances for the specified protocol client MUST be deleted. |

RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_DeleteAllTimerLocksAndRunningJobs stored procedure MUST NOT return a result set.

3.1.5.8 proc_DeleteOldTimerJobHistory

The proc_DeleteOldTimerJobHistory stored procedure is called to delete the record of job instances run before the specified date and time. The stored procedure is specified using T-SQL syntax, as specified by the following T-SQL syntax.

PROCEDURE proc_DeleteOldTimerJobHistory (  
    @DeleteBeforeDate      datetime,  
    @RequestGuid           uniqueidentifier = NULL OUTPUT
);

@DeleteBeforeDate: Job instances older than this datetime value MUST be deleted.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_DeleteOldTimerJobHistory stored procedure MUST NOT return a result set.
3.1.5.9 proc_DeleteScheduledJob

The proc_DeleteScheduledJob stored procedure is called to delete the records of job instances scheduled to run in the future. The stored procedure is specified by the following T-SQL syntax.

```
PROCEDURE proc_DeleteScheduledJob(
    @ServerId      uniqueidentifier,
    @JobId         uniqueidentifier,
    @RequestGuid   uniqueidentifier = NULL OUTPUT
);
```

@ServerId: The optional configuration object identifier of the protocol client where the job instance is executed.

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

The scheduled job instances deleted vary depending on the @ServerId and @JobId parameters as follows:

<table>
<thead>
<tr>
<th>@ServerId</th>
<th>@JobId</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>NULL</td>
<td>NULL</td>
<td>All records for scheduled job instances MUST be deleted.</td>
</tr>
<tr>
<td>Valid identifier</td>
<td>NULL</td>
<td>All records for scheduled job instances matching the @ServerId parameter MUST be deleted.</td>
</tr>
<tr>
<td>NULL</td>
<td>Valid identifier</td>
<td>All records for scheduled job instances matching the @JobId parameter MUST be deleted.</td>
</tr>
<tr>
<td>Valid identifier</td>
<td>Valid identifier</td>
<td>The scheduled job instance for the specified @JobId and @ServerId MUST be deleted.</td>
</tr>
</tbody>
</table>

@RequestGuid: The optional request identifier for the current request.

Return Values: The proc_DeleteScheduledJob stored procedure MUST return an integer result code of 0.

Result Set: The proc_DeleteScheduledJob stored procedure MUST NOT return a result set.

3.1.5.10 proc_DeleteTimerLockForJob

The proc_DeleteTimerLockForJob stored procedure is called to delete a job lock for a specified job definition if the @ServerId parameter matches the protocol client which holds the lock or if the @ServerId parameter is NULL. The stored procedure is specified by the following T-SQL syntax.

```
PROCEDURE proc_DeleteTimerLockForJob(
    @JobId          uniqueidentifier,
    @ServerId       uniqueidentifier,
    @MarkOnly       bit = 1,
    @RequestGuid    uniqueidentifier = NULL OUTPUT
);
```

@JobId: The configuration object identifier of the job definition for which a job lock is being deleted.
@ServerId: The optional configuration object identifier of a protocol client. If NULL, every job lock matching the specified job definition MUST be deleted. Otherwise only the job lock for the specified protocol client MUST be deleted.

@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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Delete job lock entry from the database.</td>
</tr>
<tr>
<td>1 (default value)</td>
<td>Mark the job lock for deletion and keep the entry in the database.</td>
</tr>
</tbody>
</table>

@RequestGuid: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A job lock was deleted.</td>
</tr>
<tr>
<td>31</td>
<td>No job locks were deleted.</td>
</tr>
</tbody>
</table>

Result Set: The proc_DeleteTimerLockForJob stored procedure MUST NOT return a result set.

3.1.5.11 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 by the following T-SQL syntax.

```
PROCEDURE proc_DeleteTimerRunningJobs (  
  @ServiceId          uniqueidentifier,  
  @VirtualServerId    uniqueidentifier,  
  @JobId              uniqueidentifier,  
  @ServerId           uniqueidentifier = NULL,  
  @RequestGuid        uniqueidentifier = 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.

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

@RequestGuid: The optional request identifier for the current request.
Return Values: The proc_DeleteTimerRunningJobs stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

Result Set: The proc_DeleteTimerRunningJobs stored procedure MUST not return a result set.

3.1.5.12 proc_DeleteTimerTargetInstance

The proc_DeleteTimerTargetInstance stored procedure is called to disassociate job definitions from a target instance. Disassociation means that a job definition will no longer be executed on a target instance. The proc_DeleteTimerTargetInstance stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_DeleteTimerTargetInstance (
    @JobId uniqueidentifier,
    @TargetInstanceId uniqueidentifier,
    @Exists bit OUTPUT,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

@JobId: The configuration object identifier of a job definition. If the @JobId parameter is NULL then all job definitions MUST be disassociated with the target instance specified by the @TargetInstanceId parameter.

@TargetInstanceId: The configuration object identifier of a target instance associated with the job definition.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The job definition has no target instances associated with it.</td>
</tr>
<tr>
<td>1</td>
<td>The job definition still has some target instances associated with it.</td>
</tr>
</tbody>
</table>

@RequestGuid: The optional request identifier for the current request.

Return Values: proc_DeleteTimerTargetInstance returns an integer return code which MUST be 0, which indicates successful execution.

Result Set: The proc_DeleteTimerTargetInstance stored procedure MUST NOT return a result set.

3.1.5.13 proc_DeleteTimerTargetInstances

The proc_DeleteTimerTargetInstances stored procedure is called to disassociate a job definition from all of its target instances. The proc_DeleteTimerTargetInstances stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_DeleteTimerTargetInstances (
    @JobId uniqueidentifier,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```
@JobId: The configuration object identifier of a job definition.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_DeleteTimerTargetInstances stored procedure MUST NOT return a result set.

3.1.5.14  proc_dropClass

The proc_dropClass stored procedure is called to remove a class and all configuration objects related to that class from the configuration database. The configuration database MUST remove any classes that have a class identifier (CLSID) equal to @ClassId or a BaseClassId equal to the class identifier (CLSID) of a class removed by proc_dropClass. The configuration database MUST remove any configuration objects with a class identifier (CLSID) equal to @ClassId.

The stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_dropClass (
    @ClassId uniqueidentifier,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

@ClassId: The class identifier (CLSID) of the class.

@RequestGuid: The optional request identifier for the current request.

Return Values:
The stored procedure MUST return an integer result code of 0.

Result Set: The proc_dropClass stored procedure MUST return a single result set, as specified in section 3.1.5.14.1.

3.1.5.14.1  DropClass Result Set

The DropClass result set MUST contain the statistics about the classes and objects removed by the stored procedure. The DropClass result set is specified by the following T-SQL syntax.

```sql
NumClasses           int,
NumObjectsBefore     int,
NumObjectsAfter      int;
```

NumClasses: The count of all the classes removed, including the classes that have a class identifier (CLSID) equal to @ClassId and all classes that have a BaseClassId equal to the class identifier (CLSID) of the class removed by proc_dropClass.

NumObjectsBefore: The count of all objects in the configuration database before the execution of the stored procedure.
NumObjectsAfter: The count of all objects in the configuration database after the execution of the stored procedure.

3.1.5.15 proc_dropEmailEnabledList

The proc_dropEmailEnabledList stored procedure is called to remove an e-mail enabled list with the specified site collection identifier, site identifier and list identifier, which is not marked for deletion, from the e-mail enabled list collection. The proc_dropEmailEnabledList stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_dropEmailEnabledList(
    @SiteId uniqueidentifier,
    @WebId uniqueidentifier,
    @ListId uniqueidentifier,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

@SiteId: The site collection identifier of the site collection which contains the list.

@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.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_dropEmailEnabledList stored procedure MUST NOT return any result sets.

3.1.5.16 proc_dropEmailEnabledListByAlias

The proc_dropEmailEnabledListByAlias stored procedure is called to remove an e-mail enabled list with the specified alias from the e-mail enabled list collection. The proc_dropEmailEnabledListByAlias stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_dropEmailEnabledListByAlias(
    @Alias nvarchar(128),
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

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

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_dropEmailEnabledListByAlias stored procedure MUST NOT return any result sets.

3.1.5.17 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 and not marked for deletion. The proc_dropEmailEnabledListsByWeb stored procedure is specified by the following T-SQL syntax.
PROCEDURE proc_dropEmailEnabledListsByWeb(
    @SiteId         uniqueidentifier,
    @WebId          uniqueidentifier,
    @RequestGuid    uniqueidentifier = NULL OUTPUT
);

@SiteId: The site collection identifier of the site collection which contains 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.

@RequestGuid: The optional request identifier for the current request.

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

The stored procedure MUST NOT return any result sets.

3.1.5.18 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 by returning an Error with Number 547 as specified in [MS-TDS] section 2.2.7.9. The configuration database MUST remove a configuration object when its parent is removed.

The proc_DropObject stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_DropObject(
    @Id           uniqueidentifier,
    @RequestGuid  uniqueidentifier = NULL OUTPUT
);

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

@RequestGuid: The optional request identifier for the current request.

Return Values: The proc_DropObject stored procedure MUST return an integer return code that is listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>1</td>
<td>The configuration object identified by @Id was not found in the configuration database.</td>
</tr>
</tbody>
</table>

Result Set: The proc_DropObject stored procedure MUST NOT return a result set.

3.1.5.19 proc_dropPendingDistributionList

The proc_dropPendingDistributionList stored procedure is called to remove a distribution list belonging to a permission level from the pending distribution lists set (section 1.3.5). The stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_dropPendingDistributionList();
@SiteId          uniqueidentifier,
@WebId           uniqueidentifier,
@GroupName       nvarchar(255),
@RequestGuid     uniqueidentifier = NULL OUTPUT
)

@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 name of the permission level whose associated distribution list is to be removed from the pending distribution lists set (section 1.3.5).

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The stored procedure proc_dropPendingDistributionList MUST NOT return any result sets.

3.1.5.20  proc_DropSiteMap

The proc_DropSiteMap stored procedure is called to delete a reference to a site collection. The proc_DropSiteMap stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_DropSiteMap (  
    @Id               uniqueidentifier,  
    @RequestGuid     uniqueidentifier = NULL OUTPUT
  )

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

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_DropSiteMap stored procedure MUST NOT return a result set.

3.1.5.21  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 by the following T-SQL syntax.

PROCEDURE proc_getDeletedEmailAliases(  
    @RequestGuid     uniqueidentifier = NULL OUTPUT
  )

@RequestGuid: The optional request identifier for the current request.

Return Values: The stored procedure returns an integer return code which MUST be 0.
Result Set: The `proc_getDeletedEmailAliases` stored procedure MUST return a single result set, as specified in section 3.1.5.21.1.

### 3.1.5.21.1 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 by the following T-SQL syntax.

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

**Alias**: The 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.

### 3.1.5.22 proc_GetDependentObjectsByBaseClass

The `proc_GetDependentObjectsByBaseClass` stored procedure is called to retrieve a list of configuration objects which depend on a specified configuration object and are in the inheritance hierarchy of the specified class. The `proc_GetDependentObjectsByBaseClass` stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_GetDependentObjectsByBaseClass (  
    @BaseClassId      uniqueidentifier,  
    @DependeeId       uniqueidentifier,  
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);
```

**@BaseClassId**: The configuration object identifier of the class at the root of the inheritance hierarchy of the classes of the returned configuration objects.

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

**@RequestGuid**: The optional request identifier for the current request

**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 specified in section 3.1.5.22.1.

### 3.1.5.22.1 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 by the following T-SQL syntax.

```sql
Id     uniqueidentifier;
```
Id: The configuration object identifier of a configuration object.

3.1.5.23 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 by the following T-SQL syntax.

```sql
PROCEDURE proc_getEmailEnabledListByAlias(
    @Alias nvarchar(128),
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

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

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The stored procedure proc_getEmailEnabledListByAlias MUST return 1 result set as specified in section 3.1.5.23.1.

3.1.5.23.1 EmailEnabledListByAlias Result Set

The EmailEnabledListByAlias result set MUST contain all rows in the e-mail enabled list collection with e-mail alias equal to @Alias, and which have not been deleted. The EmailEnabledListByAlias result set is specified by the following T-SQL syntax.

```sql
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.

3.1.5.24 proc_getFile

The proc_getFile stored procedure is called to retrieve a file from the configuration database. The proc_getFile stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_getFile(
    @ObjectId uniqueidentifier,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

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

@RequestGuid: The optional request identifier for the current request.
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 specified in section 3.1.5.24.1.

3.1.5.24.1 File Result Set

File Result Set 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 by the following T-SQL syntax.

```
FileImage        image
```

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

3.1.5.25 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 by the following T-SQL syntax.

```
PROCEDURE proc_getNewObjects(
    @NewestCachedVersion    rowversion,
    @RequestGuid            uniqueidentifier = NULL OUTPUT
);
```

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

@RequestGuid: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>1</td>
<td>@NewestCachedVersion is greater than the highest configuration object version.</td>
</tr>
</tbody>
</table>

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 3.1.5.25.1 through section 3.1.5.25.4.

3.1.5.25.1 Last Update Result Set

The LastUpdate result set returns a version number greater than the maximum configuration object version used in the configuration database. The LastUpdate result set MUST contain 1 row specified by the following T-SQL syntax.
**Version**: A row version value that MUST be greater than the maximum configuration object version used in the configuration database.

### 3.1.5.25.2 Modified Objects Result Set

The Modified Objects result set returns configuration objects with configuration object version greater than @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 contain 1 or more rows. The Objects result set is specified by the following T-SQL syntax.

```sql
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 (CLSID) 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. Properties MUST NOT be NULL.

### 3.1.5.25.3 Dependencies Result Set

The Dependencies result set returns identifiers of configuration objects which depend on configuration objects from the Modified Objects result set. The Dependencies result set MUST return zero or more rows.

The Dependencies result set is specified by the following T-SQL syntax.

```sql
DependantId    uniqueidentifier;
```

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

### 3.1.5.25.4 Tombstones Result Set

The Tombstones result set returns the identifiers of configuration objects which have been deleted after @NewestCachedVersion. The Tombstones result set MUST return 0 or more rows.
The **Tombstones result set** is specified by the following T-SQL syntax.

```sql
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.

### 3.1.5.26 proc_getPendingDistributionListsSinceVersion

The **proc_getPendingDistributionListsSinceVersion** stored procedure is called to get all distribution lists which have an operation pending whose row version is greater than @Version parameter. The stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_getPendingDistributionListsSinceVersion(
  @Version        rowversion,
  @RequestGuid    uniqueidentifier = NULL OUTPUT
);
```

**@Version:** The row version which forms the exclusive lower bound of the result set. If @Version is equal to 0, the server will return all the distribution lists pending approval.

**@RequestGuid:** The optional request identifier for the current request.

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

The stored procedure MUST return 1 result set:

### 3.1.5.26.1 PendingDistrubutionLists Result Set

The **PendingDistributionLists** result set returns a set of distribution lists marked as requiring approval. The **PendingDistributionLists** result set MUST contain each distribution list that has an operation pending whose row version is greater than the @Version parameter value. The **PendingDistributionLists** result set is specified by the following T-SQL syntax.

```sql
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.
3.1.5.27 proc_GetProductVersions

The proc_GetProductVersions stored procedure is called to retrieve information about all installations on the farm. The stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_GetProductVersions ( );

Return values:

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

Result Sets:

The proc_GetProductVersions stored procedure MUST return a single result set as specified in section 3.1.5.27.1.

3.1.5.27.1 Get Product Versions Result Set

The Get Product Versions Result Set returns all the saved information about the installed products on the farm. The Get Product Versions Result Set MUST contain one row for each update installation, as well as one row for each installation that can be patched.

  ServerId          uniqueidentifier,
  Product           nvarchar(1023),
  Version           nvarchar(64),
  PatchableUnit     nvarchar(1023),
  PatchableUnitName nvarchar(1023),
  PatchName         nvarchar(1023),
  PatchUrl          nvarchar(260),
  Flags             int,
  CustomData        ntext;

ServerId: The configuration object identifier of the front-end Web server or application server where the installation was performed.

Product: The name of the product that includes this installation.

Version: The version of the installation. The version number is in the following format:

  major.minor[.build[.revision]]

Where major, minor, build, and revision fields are integers and build and revision are optional.

PatchableUnit: If this row represents a Windows Installer (.msi) file installation, this MUST contain the ProductCode stored in the Windows Installer (.msi) file. If this row represents a third party product that can be subdivided into individual units for patching, this MUST contain a unique identifier for the individual unit. If this row represents a third party product that cannot be subdivided into individual units for patching, this MUST be NULL.

PatchableUnitName: If this row represents a Windows Installer (.msi) file installation, this MUST contain the ProductName stored in the Windows Installer (.msi) file. If this row represents a piece of software that can be subdivided into individual units for patching, this MUST contain a display
name for the individual unit. If this row represents a piece of software that cannot be subdivided into individual units for patching, this MUST be NULL.

**PatchName**: If this row represents an installed software patch, this MUST contain a display name for the software patch. Otherwise, this MUST be NULL.

**PatchUrl**: If this row represents installed software patch and there is a URL associated with the software patch, this MUST contain the URL. Otherwise, this MUST be NULL.

**Flags**: Bit flags containing information about the state of the installation. It MUST contain the sum of zero or more values from the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>This installation is required on all servers in the farm.</td>
</tr>
<tr>
<td>2</td>
<td>An optional upgrade is available for this server because of the installation state detected.</td>
</tr>
<tr>
<td>4</td>
<td>A required upgrade must be performed on this server because of the installation state detected.</td>
</tr>
</tbody>
</table>

**CustomData**: This column MUST be NULL.

### 3.1.5.28 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 by the following T-SQL syntax.

```sql
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,
    @RequestGuid               uniqueidentifier = NULL
) OUTPUT
```

**@DatabaseId**: Contains the configuration object identifier of the content database to search for the best matched site collection.

**@ApplicationId**: Contains the configuration object identifier of the Web application to search for the best matched site collection.

**@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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No limit.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>1</td>
<td>Include site collections belonging to the same Web application as the best matched site collection.</td>
</tr>
<tr>
<td>2</td>
<td>Include site collections belonging to the same content database as the best matched site collections.</td>
</tr>
</tbody>
</table>

@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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All site collections.</td>
</tr>
<tr>
<td>1</td>
<td>Only site collections which do redirect to other site collections.</td>
</tr>
<tr>
<td>2</td>
<td>Only site collections which do not redirect to other site collections.</td>
</tr>
<tr>
<td>3</td>
<td>Only site collections that have been upgraded.</td>
</tr>
</tbody>
</table>

@BestMatchSiteId: Return the site collection identifier of the best matched site collection with respect to the value of @CollectionType or 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: The configuration object identifier of the content database containing the best matched site collection.

@BestMatchApplicationId: The configuration object identifier of the Web application containing the best matched site collection.

@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 that 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.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_getSiteBestMatch stored procedure MUST NOT return a result set.
3.1.5.29 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 by the following T-SQL syntax.

```sql
PROCEDURE proc_getSiteCount(

    @DatabaseId     uniqueidentifier = NULL,
    @RequestGuid    uniqueidentifier = NULL OUTPUT

) ;
```

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

@RequestGuid: The optional request identifier for the current request.

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 specified in section 3.1.5.29.1.

3.1.5.29.1 SiteCount Result Set

The SiteCount result set 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 by the following T-SQL syntax.

```sql
sitecount        int;
```

sitecount: Contains the count of the number of site collections in the content database(s) identified by @DatabaseId. If no content database with a configuration object identifier matching @DatabaseId is found in the configuration database, the value of sitecount MUST be 0.

3.1.5.30 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 by the following T-SQL syntax.

```sql
PROCEDURE proc_GetSiteIdOfHostHeaderSite (

    @HostHeader  nvarchar(128),
    @RequestGuid  uniqueidentifier = NULL OUTPUT

) ;
```

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

@RequestGuid: The optional request identifier for the current request.

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 specified in section 3.1.5.30.1.

### 3.1.5.30.1 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 contain 0 rows. The **ID** result set is specified by the following T-SQL syntax.

```
Id       uniqueidentifier
```

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

### 3.1.5.31 proc_getSiteIds

The **proc_getSiteIds** stored procedure is called to get a list of site collection identifiers that correspond to site collections in a **site subscription**. The **proc_getSiteIds** stored procedure is specified by the following T-SQL syntax,

```
PROCEDURE proc_getSiteIds (  
  @SubscriptionId   uniqueidentifier,  
  @RequestGuid      uniqueidentifier = NULL OUTPUT
);
```

**@SubscriptionId:** Contains the identifier of the site subscription. This value MUST NOT be NULL.

**@RequestGuid:** The optional request identifier for the current request.

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

**Result Sets:** This stored procedure MUST return 1 result set as specified in section 3.1.5.31.1.

### 3.1.5.31.1 Get Site Ids Result Set

The **Site Ids** result set returns the set of site collection identifiers that correspond to site collections in a single site subscription. The **Site Ids** result set MUST be returned, and MUST contain one result for each site collection contained in the site subscription identified by the value of **@SubscriptionId**.

```
Id     uniqueidentifier;
```

**Id:** Contains the site collection identifier. This value MUST NOT be NULL.

### 3.1.5.32 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 by the following T-SQL syntax.
PROCEDURE proc_getSiteNames(
    @ContainerId uniqueidentifier,
    @ContainerType int,
    @CollectionType int,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);

@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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Indicates that the protocol server MUST return the server-relative URLs of all site collections in the Web application where @ContainerId matches the configuration object identifier of the configuration object.</td>
</tr>
<tr>
<td>1</td>
<td>Indicates that the protocol server MUST return the server-relative URLs of all site collections in the content database where @ContainerId matches the configuration object identifier of the configuration object.</td>
</tr>
</tbody>
</table>

@CollectionType: Specifies the types of site collections to be retrieved from the container. The value MUST be listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All site collections.</td>
</tr>
<tr>
<td>1</td>
<td>Only site collections which redirect to other site collections.</td>
</tr>
<tr>
<td>2</td>
<td>Only site collections which do not redirect to other site collections.</td>
</tr>
</tbody>
</table>

@RequestGuid: The optional request identifier for the current request.

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 specified in section 3.1.5.32.1.

3.1.5.32.1 Path Result Set

The Get Site Names 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 contain 0 rows.

The Path result set is specified by the following T-SQL syntax.

Path nvarchar(128),
HostHeaderIsSiteName  bit;

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

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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This site collection does NOT use a host header site identifier.</td>
</tr>
<tr>
<td>1</td>
<td>This site collection does use a host header site identifier.</td>
</tr>
</tbody>
</table>

3.1.5.33  proc_getSiteSubscriptionsInContentDB

The proc_getSiteSubscriptionsInContentDB stored procedure is called to get a list of site subscriptions that have site collections in a single content database. The proc_getSiteSubscriptionsInContentDB stored procedure is specified by the following T-SQL syntax.

```
PROCEDURE proc_getSiteSubscriptionsInContentDB (    
    @ContentDatabaseId   uniqueidentifier,    
    @RequestGuid         uniqueidentifier = NULL OUTPUT
);
```

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

@RequestGuid: The optional request identifier for the current request.

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

Result Sets:
This stored procedure MUST return 1 result set as specified in section 3.1.5.33.1.

3.1.5.33.1  Get Site Subscriptions In Content DB Result Set

The Get Site Subscriptions In Content DB result set returns a set of unique site subscription identifiers. The result set MUST contain one result for each unique site subscription identifier present on a site collection in the content database referenced by @ContentDatabaseId.

SubscriptionId uniqueidentifier

SubscriptionId: The unique identifier of the site subscription. This value MUST NOT be NULL.

3.1.5.34  proc_getSiteSubscriptionsInWebApplication

The proc_getSiteSubscriptionsInWebApplication is called to get a list of site subscriptions that have site collections in a single Web application. The proc_getSiteSubscriptionsInWebApplication stored procedure is specified by the following T-SQL syntax.
PROCEDURE proc_getSiteSubscriptionsInWebApplication (  
    @WebApplicationId    uniqueidentifier,  
    @RequestGuid        uniqueidentifier = NULL OUTPUT  
);  

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

@RequestGuid: The optional request identifier for the current request.

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

Result Sets:
This stored procedure MUST return 1 result set as specified in section 3.1.5.34.1.

3.1.5.34.1 Get Site Subscriptions in Web Application Result Set
The Get Site Subscriptions in Web Application result set returns a set of unique site subscription identifiers. The result set MUST contain one result for each unique site subscription identifier present on a site collection in the Web application referenced by @WebApplicationId.

SubscriptionId uniqueidentifier

SubscriptionId: The unique identifier of the site subscription. This value MUST NOT be NULL.

3.1.5.35 proc_getSiteSubscriptionSiteIdsInContentDatabase
The proc_getSiteSubscriptionSiteIdsInContentDatabase stored procedure is called to get a list of site collection identifiers that represent site collections contained in a single site subscription and content database. The proc_getSiteSubscriptionSiteIdsInContentDatabase stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_getSiteSubscriptionSiteIdsInContentDatabase (  
    @ContentDatabaseId   uniqueidentifier,  
    @SubscriptionId      uniqueidentifier,  
    @RequestGuid         uniqueidentifier = NULL OUTPUT  
);  

@ContentDatabaseId: Contains the configuration object identifier of a content database. This value MUST NOT be NULL.

@SubscriptionId: Contains the unique identifier of a site subscription. This value MUST NOT be NULL.

@RequestGuid: The optional request identifier for the current request.

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

Result Sets: This stored procedure MUST return 1 result set as specified in section 3.1.5.35.1.
3.1.5.35.1  Get Site Subscription Site IDs in content Database Result Set

The Get Site Subscription Site IDs in Content Database result set returns a set of unique site collection identifiers. The result set MUST contain one result for each site collection in the content database referenced by @ContentDatabaseId and whose site subscription identifier is @SubscriptionId.

Id: Contains the site collection identifier. This value MUST NOT be NULL.

3.1.5.36  proc_getSiteSubscriptionSiteIdsInWebApplication

The proc_getSiteSubscriptionSiteIdsInWebApplication stored procedure is called to get a list of site collection identifiers that represent site collections contained in a single site subscription and Web application. The proc_getSiteSubscriptionSiteIdsInWebApplication stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_getSiteSubscriptionSiteIdsInWebApplication (  
  @WebApplicationId   uniqueidentifier,  
  @SubscriptionId    uniqueidentifier,  
  @RequestGuid        uniqueidentifier = NULL OUTPUT  
);  

@WebApplicationId: Contains the configuration object identifier of a Web application. This value MUST NOT be NULL.

@SubscriptionId: Contains the unique identifier of a site subscription. This value MUST NOT be NULL.

@RequestGuid: The optional request identifier for the current request.

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

Result Sets:

This stored procedure MUST return 1 result set as specified in section 3.1.5.36.1.

3.1.5.36.1  Get Site Subscription Site Ids In Web Application Result Set

The Get Site Subscription Site Ids In Web Application result set returns a set of unique site collection identifiers. The result set MUST contain one result for each site collection in the Web application referenced by @WebApplicationId and whose site subscription identifier is @SubscriptionId.

Id: Contains the site collection identifier. This value MUST NOT be NULL.
3.1.5.37 proc_getSubscriptionIds

The proc_getSubscriptionIds stored procedure is called to get a list of site subscription identifiers in the farm. The proc_getSubscriptionIds stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_getSubscriptionIds (  
    @RequestGuid uniqueidentifier = NULL OUTPUT  
);  
```

@RequestGuid: The optional request identifier for the current request.

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

Result Sets:

This stored procedure MUST return 1 result set as specified in section 3.1.5.37.1.

3.1.5.37.1 Get Subscription Ids Result Set

The Get Subscription Ids result set returns a list of site subscription identifiers in the farm. The result set MUST contain one result per site subscription identifier.

    SubscriptionId uniqueidentifier

SubscriptionId: Contains the site subscription identifier. This value MUST NOT be NULL.

3.1.5.38 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 by the following T-SQL syntax.

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

@DatabaseId: The configuration object identifier of a content database. If @DatabaseId is NULL, it refers all the site collections in all content databases.

@ApplicationId: The configuration object identifier of a Web application. If @ApplicationId is NULL, it refers all the site collections in all Web applications.

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

@StartRow: The index of the starting row from which the data will be retrieved.
@SortDirection: The order in which the data is sorted. The value MUST be listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESC</td>
<td>Data will be returned in descending order according to identifier.</td>
</tr>
<tr>
<td>ASC</td>
<td>Data will be returned in ascending order according to identifier.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>All site collections.</td>
</tr>
<tr>
<td>1</td>
<td>Only site collections which redirect to other site collections.</td>
</tr>
<tr>
<td>2</td>
<td>Only site collections which do not redirect to other site collections.</td>
</tr>
<tr>
<td>3</td>
<td>Only site collections which have been upgraded.</td>
</tr>
</tbody>
</table>

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

Result Set: The proc_getSiteSubset stored procedure MUST return a single result set as specified in section 3.1.5.38.1.

3.1.5.38.1 Site Result Set

The Site result set returns the site collections identified by @CollectionType, @DatabaseId, and @ApplicationId. The number of rows in the Site result set MUST NOT be greater than the value of @PageSize. If there are no site collections in the content database identified by CollectionType @DatabaseId and @ApplicationId, the Site result set MUST be returned and MUST NOT contain any rows.

The Site result set is specified by the following T-SQL syntax.

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

Id: The identifier of the site collection.

Path: The store-relative form URL for the site collection.

3.1.5.39 proc_getTemplate

The proc_getTemplate stored procedure is called to retrieve the content of a site template. The proc_gettemplate stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_gettemplate(
    @ObjId          uniqueidentifier = NULL,
    @RequestGuid    uniqueidentifier = NULL OUTPUT
);
```
@ObjId: The configuration object identifier of the site template whose content is being retrieved.

@RequestGuid: The optional request identifier for the current request.

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 specified in section 3.1.5.39.1.

3.1.5.39.1 Template Result Set

The Template result set MUST be returned and it MUST contain 0 or more rows. When @ObjId is NULL, the Template result set MUST be returned and it MUST contain 1 row for each site template stored in the configuration 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 @ObjId is not NULL and 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 by the following T-SQL syntax.

```
FILEimage image

FILEimage: MUST contain the contents of the site template referred to by @ObjId.
```

3.1.5.40 proc_GetTimerJobHistory

The proc_GetTimerJobHistory stored procedure returns a set of job history entries. The parameters @ServiceId, @WebApplicationId, @ServerId, @JobId, and @JobStatus act as filters. If any one of these parameters is NULL, then the proc_GetTimerJobHistory stored procedure ignores that parameter when building the result set of job history entries. The proc_GetTimerJobHistory stored procedure is specified by the following T-SQL syntax.

```
PROCEDURE proc_GetTimerJobHistory (  
    @ServiceId          uniqueidentifier,  
    @WebApplicationId   uniqueidentifier,  
    @ServerId           uniqueidentifier,  
    @JobId              uniqueidentifier,  
    @JobStatus          int = NULL,  
    @MinimumId          bigint,  
    @MaximumRows        int,  
    @RequestGuid        uniqueidentifier = NULL OUTPUT  
);  

@ServiceId: The optional configuration object identifier of a service. If not NULL, the result set entries MUST be associated with the specified service.

@WebApplicationId: The optional configuration object identifier of a Web application. If not NULL, the result set entries MUST be associated with the specified Web application.

@ServerId: The optional configuration object identifier of a protocol client. If not NULL, the result set entries MUST be associated with the specified protocol client.
@JobId: The optional configuration object identifier of a job definition. If not NULL, the result set entries MUST be associated with the specified job definition.

@JobStatus: The optional Job Status Type of a finished job instance. If not NULL, the result set entries MUST match the specified Job Status Type.

@MaximumId: The maximum value of a job history entry identifier to be returned in the Timer Job History result set.

@MinimumId: The minimum value of a job history entry identifier to be returned in the Timer Job History result set.

@MaximumRows: The maximum number of rows to be returned in the Timer Job History result set.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_GetTimerJobHistory stored procedure MUST return a single Job History result set. The result set contains 0 or more rows and is specified in section 3.1.5.40.1.

3.1.5.40.1 Timer Job History Result Set

The Job History result set returns a list of job history entries. The Job History result set MUST contain one row for each job history entry and is specified by the following T-SQL syntax.

```sql
Id bigint,
JobId uniqueidentifier,
JobTitle nvarchar(255),
WebApplicationId uniqueidentifier,
WebApplicationName nvarchar(255),
ServiceId uniqueidentifier,
ServerId uniqueidentifier,
ServerName nvarchar(128),
Status int,
StartTime datetime,
EndTime datetime,
DatabaseName nvarchar(128),
ErrorMessage nvarchar(1000);
```

Id: The SQL identity column value created when the job history entry was added.

JobId: The configuration object identifier of the job definition.

JobTitle: The title of the job definition.

WebApplicationId: 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.

WebApplicationName: The name of the Web application associated with the job definition. This MUST be NULL when the job definition is associated only with a service.

ServiceId: The configuration object identifier of the service associated with the job definition.
ServerId: The configuration object identifier of the protocol client where the job instance was run.

ServerName: The name of the protocol client where the job instance was run.

Status: The Job Status Type of the job instance.

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

EndTime: The datetime value when the job instance stopped running.

DatabaseName: The name of the database processed when a job instance uses a content database lock, otherwise NULL.

ErrorMessage: The error message associated with an error that occurred while running the job instance, otherwise NULL.

3.1.5.41 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 by the following T-SQL syntax.

PROCEDURE proc_GetTimerJobLastRunTime (  
  @JobId uniqueidentifier,  
  @LastRunTime datetime = NULL OUTPUT,  
  @RequestGuid uniqueidentifier = NULL OUTPUT  
);

@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 and is NULL.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_GetTimerJobLastRunTime stored procedure MUST not return a result set.

3.1.5.42 proc_GetTimerRunningJobs

The proc_GetTimerRunningJobs stored procedure returns a set of running job status entries for a service, Web application, protocol client, or job definition. The proc_GetTimerRunningJobs stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_GetTimerRunningJobs(  
  @ServiceId uniqueidentifier,  
  @WebApplicationId uniqueidentifier,  
  @ServerId uniqueidentifier,  
  @JobId uniqueidentifier,  
  @RequestGuid uniqueidentifier = NULL OUTPUT  
);

@ServiceId: The optional configuration object identifier of a service. If not NULL, the result set entries MUST be associated with the specified service.
@WebApplicationId: The optional configuration object identifier of a Web application. If not NULL, the result set entries MUST be associated with the specified Web application.

@ServerId: The optional configuration object identifier of a protocol client. If not NULL, the result set entries MUST be associated with the specified protocol client.

@JobId: The optional configuration object identifier of a job definition. If not NULL, the result set entries MUST be associated with the specified job definition.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_GetTimerRunningJobs stored procedure MUST return a single Job Status result set. The result set contains 0 or more rows and is specified in section 3.1.5.42.1.

3.1.5.42.1 Job Status Result Set

The Job Status result set returns an unordered list of job status entries. The Job Status result set MUST contain one row for each job status entry and is specified by the following T-SQL syntax.

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

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

VirtualServerId: The configuration object identifier of the Web application. This parameter 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.

ServerId: The configuration object identifier 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.
3.1.5.43 proc_GetTimerScheduledJobs

The proc_GetTimerScheduledJobs stored procedure returns a set of scheduled job entries for a service, Web application, protocol client or job definition. The proc_GetTimerScheduledJobs stored procedure is specified by the following T-SQL syntax.

```
PROCEDURE proc_GetTimerScheduledJobs (
    @ServiceId          uniqueidentifier,
    @WebApplicationId   uniqueidentifier,
    @ServerId           uniqueidentifier,
    @JobId              uniqueidentifier,
    @StartRow           int,
    @MaximumRows        int,
    @RequestGuid        uniqueidentifier = NULL OUTPUT
);
```

@ServiceId: The optional configuration object identifier of a service. If not NULL, the result set entries MUST be associated with the specified service.

@WebApplicationId: The optional configuration object identifier of a Web application. If not NULL, the result set entries MUST be associated with the specified Web application.

@ServerId: The optional configuration object identifier of a protocol client. If not NULL, the result set entries MUST be associated with the specified protocol client.

@JobId: The optional configuration object identifier of a job definition. If not NULL, the result set entries MUST be associated with the specified job definition.

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

@MaximumRows: The maximum number of scheduled job entries that can be included in the Timer Scheduled Job result set.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_GetTimerScheduledJobs stored procedure MUST return a single Timer Scheduled Job result set. The result set contains 0 or more rows and is specified in section 3.1.5.43.1.

3.1.5.43.1 Timer Scheduled Job Result Set

The Scheduled Job result set returns a list of scheduled job entries. The list MUST be sorted by the StartTime in ascending order. The Scheduled Job result set MUST contain one row for each scheduled job and is specified by the following T-SQL syntax.

```
ServiceId            uniqueidentifier,
WebApplicationId     uniqueidentifier,
ServerId             uniqueidentifier,
JobId                uniqueidentifier,
StartTime            datetime;
```

ServiceId: The configuration object identifier of the service.
**WebApplicationId:** The configuration object identifier of the Web application. This MUST be NULL if the job definition is not associated with a Web application.

**ServerId:** The configuration object identifier of the protocol client where the job instance will be run.

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

**StartTime:** The datetime value when the job instance is scheduled to run.

### 3.1.5.44 proc_GetTimerTargetInstance

The **proc_GetTimerTargetInstance** stored procedure is called to determine if the specified target instance exists for a job definition. The stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_GetTimerTargetInstance (  
    @JobId               uniqueidentifier,
    @TargetInstanceId   uniqueidentifier,
    @Exists             bit     OUTPUT,
    @RequestGuid        uniqueidentifier = NULL OUTPUT
);
```

**@JobId:** The optional configuration object identifier of a job definition.

**@TargetInstanceId:** The optional GUID of a target instance.

**@Exists:** A bit indicating whether the specified target instance exists.

The value returned MUST be in the following table:

<table>
<thead>
<tr>
<th>@Exists Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A target instance does not exist.</td>
</tr>
<tr>
<td>1</td>
<td>A target instance exists.</td>
</tr>
</tbody>
</table>

**@RequestGuid:** The optional request identifier for the current request.

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

**Result Set:** The stored procedure MUST NOT return a result set.

### 3.1.5.45 proc_GetTimerTargetInstanceState

The **proc_GetTimerTargetInstanceState** stored procedure returns the target instance state for the specified target instance and job definition. The **proc_GetTimerTargetInstanceState** stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_GetTimerTargetInstanceState (  
    @JobId       uniqueidentifier,
    @TargetInstanceId   uniqueidentifier,
    @RequestGuid       uniqueidentifier = NULL OUTPUT
);
```

[MS-WSSCFGD2] — v20120630

Windows SharePoint Services: Configuration Database Communications Version 2 Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
@JobId: The configuration object identifier of a job definition.

@TargetInstanceId: The GUID of a target instance.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: This stored procedure MUST return Timer Target Instance State result set as defined in section 3.1.5.45.1.

3.1.5.45.1 Timer Target Instance State Result Set

Timer Target Instance State result set returns the target instance state for a target instance and job definition. The result set MUST contain 1 row if a target instance for the specified job definition exists, otherwise the result set MUST contain 0 rows. The Timer Target Instance State result set is specified by the following T-SQL syntax.

```
State varbinary(max);
```

State: Binary state data for a target instance and job definition.

3.1.5.46 proc_markForDeletionEmailEnabledList

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

```
PROCEDURE proc_markForDeletionEmailEnabledList(
    @SiteId        uniqueidentifier,
    @WebId         uniqueidentifier,
    @ListId        uniqueidentifier,
    @RequestGuid   uniqueidentifier = NULL OUTPUT
);
```

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

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

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

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_markForDeletionEmailEnabledList stored procedure MUST NOT return any result sets.

3.1.5.47 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 by the following T-SQL syntax.

---

[MS-WSSCFGD2] — v20120630
Windows SharePoint Services: Configuration Database Communications Version 2 Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
PROCEDURE proc_markForDeletionEmailEnabledListsBySite(
    @SiteId           uniqueidentifier,
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);

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

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_markForDeletionEmailEnabledListsBySite stored procedure MUST NOT return any result sets.

3.1.5.48 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_markForDeletionEmailEnabledListsByWeb stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_markForDeletionEmailEnabledListsByWeb(
    @SiteId           uniqueidentifier,
    @WebId            uniqueidentifier,
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);

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

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

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_markForDeletionEmailEnabledListsByWeb stored procedure MUST NOT return any result sets.

3.1.5.49 proc_putClass

The proc_putClass stored procedure is called to store a class. The proc_putClass stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_putClass(
    @Id             uniqueidentifier,
    @BaseClassId    uniqueidentifier,
    @FullName       nvarchar(256),
    @RequestGuid    uniqueidentifier = NULL OUTPUT
);

@Id: Contains the class identifier (CLSID) of the class.
@BaseClassId: Contains the BaseClassId of the class.

@FullName: Contains an identifier that can be used by an application to associate a class identifier (CLSID) with a human- or computer-readable string. @FullName MUST be specified and MUST NOT be DBNull.

@RequestGuid: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>1</td>
<td>The class information of the GUID associated with @BaseClassId was not found.</td>
</tr>
</tbody>
</table>

Result Set: The proc_putClass stored procedure MUST NOT return any result sets.

3.1.5.50 proc_putDependency

The proc_putDependency stored procedure is called to store a dependency between two configuration objects. The proc_putDependency stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_putDependency(
    @ObjectId uniqueidentifier,
    @DependantId uniqueidentifier,
    @RequestGuid uniqueidentifier = NULL OUTPUT
)
```

@ObjectId: The identifier of the configuration object that depends on the configuration object that is referenced by @DependantId. The value MUST be defined in the configuration database, or the server MUST return an exception.

@DependantId: The identifier of the configuration object on which the configuration object referenced by @ObjectId depends. The value MUST be defined in the configuration database, or the server MUST return an exception.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_putDependency stored procedure MUST NOT return a result set.

3.1.5.51 proc_putDistributionListToDelete

The proc_putDistributionListToDelete stored procedure is called to mark a distribution list as deleted. The stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_putDistributionListToDelete(
    @Alias nvarchar(128),
    @RequestGuid uniqueidentifier = NULL OUTPUT
)
```
@Alias: The e-mail alias of the distribution list to be marked as deleted.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_putDistributionListToDelete stored procedure MUST NOT return any result sets.

3.1.5.52 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 list in the e-mail enabled list collection. The proc_putEmailEnabledList stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_putEmailEnabledList(
    @Alias nvarchar(128),
    @SiteId uniqueidentifier,
    @WebId uniqueidentifier,
    @ListId uniqueidentifier,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

@Alias: The new or existing e-mail alias of the e-mail enabled list. It MUST NOT be NULL.

@SiteId: The site collection identifier of the site collection containing the site. It MUST NOT be NULL.

@WebId: The site identifier of the site containing the list. It MUST NOT be NULL.

@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 stored 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. It MUST NOT be NULL.

@RequestGuid: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>1</td>
<td>Error: The specified e-mail alias is already in use in the e-mail enabled list collection.</td>
</tr>
</tbody>
</table>

Result Set: The proc_putEmailEnabledList stored procedure MUST NOT return any result sets.

3.1.5.53 proc_putFileSegment

The proc_putFileSegment stored procedure is called to write a byte range of a file. The proc_putFileSegment stored procedure is specified by the following T-SQL syntax.
PROCEDURE proc_putFileSegment(
    @ObjectId uniqueidentifier,
    @Bytes varbinary(max),
    @Offset int,
    @RequestGuid uniqueidentifier = NULL OUTPUT
)

@ObjectId: The identifier of the file being stored.

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

@Offset: The value that determines the relative location of the start of the file in bytes. This MUST be set to 0 the first time the stored procedure is called with a new @ObjectId value.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>1</td>
<td>The file identified by @ObjectId was not found and @Offset was greater than 0.</td>
</tr>
<tr>
<td>2</td>
<td>The file write operation failed.</td>
</tr>
</tbody>
</table>

The proc_putFileSegment stored procedure MUST NOT return a result set.

3.1.5.54 proc_putObject

The proc_putObject stored procedure is called to store new or update existing configuration objects. The proc_putObject stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_putObject (  
    @Id uniqueidentifier,  
    @ParentId uniqueidentifier,  
    @ClassId uniqueidentifier,  
    @Name nvarchar(128),  
    @Status int,  
    @Version rowversion,  
    @Properties ntext,  
    @AutoResolveMissingTypes bit,  
    @ExistingObject uniqueidentifier output,  
    @NewVersion rowversion output,  
    @RequestGuid uniqueidentifier = NULL 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, as defined in [MS-WSSFO2] section 2.2.6.1.4, 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 class identifier (CLSID) 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-WSSFO2] section 3.1.5.35 or the proc_getNewObjects stored procedure which was used to obtain the configuration object being updated.

@Properties: The properties of the configuration object.

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

@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 class identifier (CLSID) but with a different identifier, the protocol server MUST set @ExistingObject to the identifier of the configuration object. Otherwise, the protocol server MUST NOT change the value of @ExistingObject.

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

@RequestGuid: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>1</td>
<td>Returned if the protocol client passed in value other than DBNull for @Version, but a configuration object with the specified identifier was not found.</td>
</tr>
<tr>
<td>2</td>
<td>Returned if a GUID value is passed to @ClassId without having been previously passed to the @Id parameter of the proc_putClass stored procedure.</td>
</tr>
<tr>
<td>3</td>
<td>Returned if a failure occurs when trying to create a new configuration object.</td>
</tr>
<tr>
<td>4</td>
<td>Returned when the caller tries to update an existing configuration object with a version value that is different than what is referenced by @Version.</td>
</tr>
<tr>
<td>5</td>
<td>Returned if a failure occurs when updating an existing configuration object.</td>
</tr>
<tr>
<td>6</td>
<td>Returned if a failure occurs when setting the value of @NewVersion.</td>
</tr>
<tr>
<td>8</td>
<td>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 class identifier (CLSID), ParentId, and name values of a configuration object that is already in the configuration database.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>9</td>
<td>Returned when @Version is DBNull and @Id matches the identifier of an existing configuration object.</td>
</tr>
</tbody>
</table>

**Result Set:** The `proc_putObject` stored procedure MUST NOT return a result set.

### 3.1.5.55 proc_putPendingDistributionList

The `proc_putPendingDistributionList` stored procedure is called to add a distribution list which has an operation pending to the pending distribution lists set (section 1.3.5). The stored procedure is specified by the following T-SQL syntax.

```tsql
PROCEDURE proc_putPendingDistributionList(
    @SiteId uniqueidentifier,
    @WebId uniqueidentifier,
    @GroupName nvarchar(255),
    @ModifiedBy nvarchar(255),
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

**@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 of the distribution list.

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

**@RequestGuid:** The optional request identifier for the current request.

**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 specified in section 3.1.5.55.1.

#### 3.1.5.55.1 PutPendingDistributionList Result Set

The `PutPendingDistributionList` result set MUST be returned and MUST contain only 1 row when `@SiteId`, `@WebId` and `@GroupName` corresponds to a distribution list which has an operation pending. If no such list exists, the `PutPendingDistributionList` result set MUST contain 0 rows.

The `PutPendingDistributionList` result set is specified by the following T-SQL syntax.

```tsql
{PendingListRegistered} int
```

**PendingListRegistered:** MUST contain the value 0 if the result set has 1 row.

### 3.1.5.56 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 by the following T-SQL syntax.
PROCEDURE proc_putSiteMap(
    @ApplicationId uniqueidentifier,
    @DatabaseId uniqueidentifier,
    @SiteId uniqueidentifier,
    @SubscriptionId uniqueidentifier,
    @Path nvarchar(128),
    @Pairing tinyint,
    @RedirectUrl nvarchar(512),
    @HostHeaderIsSiteName bit,
    @CurrentDatabaseSiteCount int OUTPUT
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
Return Values: The proc_putSiteMap stored procedure returns an integer return code which MUST be 0, which indicates successful execution.

Result Set: If the proc_putSiteMap stored procedure fails, it MUST NOT return a result set. If proc_putSiteMap executes successfully, it MUST return a result set with a single row as specified in section 3.1.5.56.1.

3.1.5.56.1 SiteId Result Set

SiteId result set returns the GUID of the new site collection. The SiteId result set is specified by the following T-SQL syntax.

{SiteId} uniqueidentifier;

SiteId: Contains the value of @SiteId.

3.1.5.57 proc_RefreshAllTimerLocks

The proc_RefreshAllTimerLocks stored procedure is called to renew the lease on all job locks held by a specified protocol client. The stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_RefreshAllTimerLocks(
    @ServerId       uniqueidentifier,
    @RequestGuid    uniqueidentifier = NULL OUTPUT
);

@ServerId: The configuration object identifier of the protocol client holding the job lock(s).

@RequestGuid: The optional request identifier for the current request.

Return Code Values:

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

Result Set: The proc_RefreshAllTimerLocks stored procedure MUST NOT return a result set.

3.1.5.58 proc_RegisterProductVersion

The proc_RegisterProductVersion stored procedure is called to save information about an installation on the farm. The stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_RegisterProductVersion ( 
    @ServerId             uniqueidentifier,
    @Product              nvarchar(1023),
    @Version              nvarchar(64),
    @PatchableUnit        nvarchar(1023) = NULL,
    @PatchableUnitName    nvarchar(1023) = NULL,
    @PatchName            nvarchar(1023) = NULL,
    @PatchUrl             nvarchar(260) = NULL,
    @Flags                int            = NULL,
    @CustomData           nvarchar(max)  = NULL
);
@ServerId: The configuration object identifier of the client where the installation was performed.

@Product: The name of the product that includes this installation.

@Version: The version of the installation. The version number is in the following format:

major.minor[.build[.revision]]

Where major, minor, build, and revision fields are integers and build and revision are optional.

@PatchableUnit: If this stored procedure is called for Windows Installer (.msi) file installations, this MUST contain the ProductCode stored in the Windows Installer (.msi) file. If this stored procedure is called for third party products that can be subdivided into individual units for patching, this MUST contain a unique identifier for the individual unit.

@PatchableUnitName: If this stored procedure is called for Windows Installer (.msi) file installations, this MUST contain the ProductName stored in the Windows Installer (.msi) file. If this stored procedure is called for third party products that can be subdivided into individual units for patching, this MUST contain a display name for the individual unit.

@PatchName: If this stored procedure is being called for an installed patch, this MUST contain a display name for the patch.

@PatchUrl: If this stored procedure is being called for an installed patch and there is a URL associated with the patch, this MUST contain the URL.

@Flags: Bit flags containing information about the state of the installation. It MUST contain the sum of zero or more values from the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>This installation is required on all servers in the farm.</td>
</tr>
<tr>
<td>2</td>
<td>An optional upgrade is available for this server because the installation state detected.</td>
</tr>
<tr>
<td>4</td>
<td>A required upgrade must be performed on this server because the installation state detected.</td>
</tr>
</tbody>
</table>

@CustomData: This is reserved and MUST be NULL.

Return values:

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

Result Set: The proc_RegisterProductVersion stored procedure MUST NOT return a result set.

3.1.5.59 proc_renameSiteMap

The proc_renameSiteMap stored procedure is called to update the host header site identifier for the specified site collection. The proc_renameSiteMap stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_renameSiteMap(
    @SiteId    uniqueidentifier,
    @Path      nvarchar(128)
)``,
```
@SiteId: Contains the site collection identifier of the site collection to be updated. The site collection referenced by @SiteId MUST use a host header site identifier. This parameter MUST NOT be NULL.

@Path: Contains the host header site identifier 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.

Result Set: The proc_renameSiteMap stored procedure MUST NOT return a result set.

3.1.5.60 proc_ScheduleTimerJob

The proc_ScheduleTimerJob stored procedure is called to create a scheduled job for the specified protocol client. The stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_ScheduleTimerJob (  @ServiceId          uniqueidentifier,  @WebApplicationId   uniqueidentifier,  @ServerId           uniqueidentifier,  @JobId              uniqueidentifier,  @StartTime          datetime,  @RequestGuid        uniqueidentifier = NULL OUTPUT  
    );
```

@ServiceId: The configuration object identifier of the service associated with the job definition. This parameter MUST NOT be NULL.

@WebApplicationId: 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.

@ServerId: The configuration object identifier of the protocol client where the job instance is scheduled to run.

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

@StartTime: The datetime value when the job instance is scheduled to run.

@RequestGuid: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>31</td>
<td>If the specified job instance cannot be scheduled.</td>
</tr>
</tbody>
</table>
3.1.5.61 proc_setSubscription

The proc_setSubscription stored procedure is called to set the site subscription identifier of a site collection.

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

```sql
PROCEDURE proc_setSubscription (
    @SiteId           uniqueidentifier,
    @SubscriptionId   uniqueidentifier,
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);
```

@SiteId: The site collection identifier of the site collection. This value MUST NOT be NULL.

@SubscriptionId: The identifier of the site subscription.

@RequestGuid: The optional request identifier for the current request.

Return values: The proc_setSubscription stored procedure returns an integer return code that MUST be 0.

3.1.5.62 proc_SiteExists

The proc_SiteExists stored procedure is called to determine if the configuration database contains a reference to a site collection with a given site collection identifier.

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

```sql
PROCEDURE proc_SiteExists (
    @SiteId        uniqueidentifier,
    @RequestGuid   uniqueidentifier = NULL OUTPUT
);
```

@SiteId: The site collection identifier of the site collection for which the configuration database will be searched.

@RequestGuid: The optional request identifier for the current request.

Return Code Values: An integer which MUST be one of the following values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The configuration database contains a reference to a site collection with a site collection identifier equal to @SiteId.</td>
</tr>
<tr>
<td>0</td>
<td>The configuration database does not contain a reference to a site collection with a site collection identifier equal to @SiteId.</td>
</tr>
</tbody>
</table>

Result Set: The proc_SiteExists stored procedure MUST NOT return a result set.

3.1.5.63 proc_StartTimerRunningJob

The proc_StartTimerRunningJob stored procedure is called to mark a job instance as started. The proc_StartTimerRunningJob stored procedure is specified by the following T-SQL syntax.
PROCEDURE proc_StartTimerRunningJob (
    @ServiceId           uniqueidentifier,
    @VirtualServerId     uniqueidentifier,
    @JobId               uniqueidentifier,
    @JobTitle            nvarchar(255),
    @ServerId            uniqueidentifier,
    @TargetCount         int,
    @CurrentTarget       int OUTPUT,
    @RequestGuid         uniqueidentifier = 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 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.

@ServerId: The configuration object identifier of the protocol client where the job instance will be executed.

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

@CurrentTarget: The number of target instances that have been processed. An existing running job can be retrieved using the proc_GetTimerRunningJobs stored procedure and specifying the provided @ServiceId, @VirtualServerId, @JobId, @ServerId, and @RequestGuid parameters. If a running job exists and the Status property from the Job Status Result Set is '7' and the TargetCount equals the @TargetCount parameter then @CurrentTarget MUST return the CurrentTarget value from the Job Status result set. Otherwise it MUST return 0.

@RequestGuid: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>31</td>
<td>If the specified job instance cannot be started</td>
</tr>
</tbody>
</table>

Result Set: The proc_StartTimerRunningJob stored procedure MUST not return a result set.

3.1.5.64 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 by the following T-SQL syntax.

PROCEDURE proc_UpdateTimerRunningJobProgress (
    @ServiceId           uniqueidentifier,
    @VirtualServerId     uniqueidentifier,
    @JobId               uniqueidentifier,
    @ServerId            uniqueidentifier,
    @CurrentTarget       int OUTPUT,
    @RequestGuid         uniqueidentifier = 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 MUST be NULL when the job definition is associated only with a service.

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

@ServerId: The configuration object identifier of the protocol client where the job instance is running.

@CurrentTargetPercentDone: The percentage of processing that has been finished by the job instance. If it is unknown, the value MUST be NULL. Otherwise, the value MUST be between "0" and "100" including the bounds.

@RequestGuid: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>31</td>
<td>If there are no updates to the progress of the specified job instance.</td>
</tr>
</tbody>
</table>

Result Set: The proc_UpdateTimerRunningJobProgress stored procedure MUST not return a result set.

3.1.5.65 proc_UpdateTimerRunningJobStatus

The proc_UpdateTimerRunningJobStatus stored procedure is called to update the status of the specified running job instance. The proc_UpdateTimerRunningJobStatus stored procedure is specified by the following T-SQL syntax.

```
PROCEDURE proc_UpdateTimerRunningJobStatus (  
    @JobId         uniqueidentifier,  
    @ServerId      uniqueidentifier,  
    @NewStatus     int,  
    @OldStatus     int,  
    @RequestGuid   uniqueidentifier = NULL OUTPUT  
);  
```

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

@ServerId: The configuration object identifier of the protocol client where the job instance is running.

@NewStatus: The new Job Status Type for the job instance.

@OldStatus: The existing Job Status Type of the running job instance. If NULL, the existing Job Status Type of the running job instance MUST be updated to the value specified in the @NewStatus
parameter. Otherwise, the existing Job Status Type of the running job MUST only be updated when the Job Status Type matches the value of the @OldStatus parameter.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: The proc_UpdateTimerRunningJobStatus stored procedure MUST NOT return a result set.

3.1.5.66 proc_UpdateTimerRunningJobTarget

The proc_UpdateTimerRunningJobTarget stored procedure is called to update the running target instance for the specified job instance. The proc_UpdateTimerRunningJobTarget stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_UpdateTimerRunningJobTarget (  
    @ServiceId          uniqueidentifier,  
    @VirtualServerId    uniqueidentifier,  
    @JobId              uniqueidentifier,  
    @ServerId           uniqueidentifier,  
    @CurrentTarget      int,  
    @RequestGuid        uniqueidentifier = 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 MUST be NULL when the job definition is associated only with a service.

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

@ServerId: The configuration object identifier of the protocol client where the job instance is being executed.

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

@RequestGuid: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>31</td>
<td>If there are no updates to the target instance of the specified job instance.</td>
</tr>
</tbody>
</table>

Result Set: The proc_UpdateTimerRunningJobTarget stored procedure MUST not return a result set.
3.1.5.67 proc_UpdateTimerTargetInstanceState

The proc_UpdateTimerTargetInstanceState stored procedure is called to update the target instance state for the specified job definition and target instance. The stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_UpdateTimerTargetInstanceState (  
    @JobId              uniqueidentifier,
    @TargetInstanceId   uniqueidentifier,
    @State              varbinary(max),
    @RequestGuid        uniqueidentifier = NULL OUTPUT
 );
```

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

@TargetInstanceId: The configuration object identifier of a target instance.

@State: The binary state data for a target instance and job definition.

@RequestGuid: The optional request identifier for the current request.

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

Result Set: This stored procedure MUST not return a result set.

3.1.5.68 proc_hideSiteMap

The 3.1.5.68 proc_hideSiteMap stored procedure is called to mark a reference to a site collection proc_hideSiteMap for deletion, as specified by the following T-SQL syntax.

```sql
PROCEDURE proc_hideSiteMap (  
    @Id               uniqueidentifier,  
    @DatabaseId       uniqueidentifier,  
    @RequestGuid      uniqueidentifier = NULL OUTPUT
 );
```

@Id: The site collection identifier of the site collection whose reference is being marked for deletion.

@DatabaseId: Contains the configuration object identifier of the content database that contains the site collection.

@RequestGuid: The optional request identifier for the current request.

Return Values: The proc_hideSiteMap stored procedure returns an integer return code that MUST be 0, indicating a successful execution.

Result Set: The proc_hideSiteMap stored procedure MUST NOT return a result set.

3.1.5.69 proc_unhideSiteMap

The proc_unhideSiteMap stored procedure is called to unmark a reference to a site collection. proc_unhideSiteMap for deletion. The stored procedure is specified by the following T-SQL syntax.
PROCEDURE proc_unhideSiteMap (  
  @Id               uniqueidentifier,  
  @DatabaseId       uniqueidentifier,  
  @RequestGuid      uniqueidentifier = NULL OUTPUT  
);  

@Id: The site collection identifier of the site collection whose reference is being unmarked for deletion.

@DatabaseId: Contains the configuration object identifier of the content database that contains the site collection.

@RequestGuid: The optional request identifier for the current request.

Return Values: The proc_unhideSiteMap stored procedure returns an integer return code that MUST be 0, indicating a successful execution.

Result Set: The proc_unhideSiteMap stored procedure MUST NOT return a result set.

3.1.5.70 proc_dropSiteMap2

The proc_dropSiteMap2 stored procedure is called to delete a reference to a site collection.

proc_dropSiteMap2. The stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_dropSiteMap2 (  
  @Id               uniqueidentifier,  
  @DatabaseId       uniqueidentifier,  
  @RequestGuid      uniqueidentifier = NULL OUTPUT  
);  

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

@DatabaseId: Contains the configuration object identifier of the content database that contains the site collection.

@RequestGuid: The optional request identifier for the current request.

Return Values: The proc_dropSiteMap2 stored procedure returns an integer return code that MUST be 0, indicating a successful execution.

Result Set: The proc_dropSiteMap2 stored procedure MUST NOT return a result set.

3.1.5.71 proc_getDeletedSitesByUrl

The proc_getDeletedSitesByUrl stored procedure is called to return information about site collections that have been marked for deletion. The proc_getDeletedSitesByUrl stored procedure is specified by the following T-SQL syntax.

PROCEDURE proc_unhideSiteMap (  
  @ApplicationId    uniqueidentifier,  
  @Path             nvarchar(128),  
  @RowLimit         int,  
  @RequestGuid      uniqueidentifier = NULL OUTPUT  
);
@ApplicationId: Contains the configuration object identifier of the Web application.

@Path: Contains the prefix of a server-relative URL of the site collection. This parameter MUST NOT be NULL.

@RowLimit: The maximum number of rows to return.

@RequestGuid: The optional request identifier for the current request.

Return Values: The proc_getDeletedSitesByUrl stored procedure returns an integer return code that MUST be 0, indicating a successful execution.

Result Set: The proc_getDeletedSitesByUrl stored procedure MUST return a single result set as specified in section 3.1.5.71.1.

3.1.5.71.1 Site Result Set

The Site result set returns the site collections identified by @ApplicationId and @Path. The number of rows in the Site result set MUST NOT be greater than the value of @RowLimit. If there are no site collections in the content database identified by @ApplicationId and @Path, the Site result set MUST be returned and MUST NOT contain any rows.

The Site result set is specified by the following T-SQL syntax.

```
Id uniqueidentifier,
SubscriptionId uniqueidentifier,
DatabaseId uniqueidentifier,
DeletionTime datetime,
HostHeaderIsSiteName bit,
Pairing tinyint;
```

Id: The identifier of the site collection.

SubscriptionId: The identifier of the site subscription.

DatabaseId: The configuration object identifier of the content database.

DeletionTime: The date and time that the site collection was marked for deletion.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This site collection does NOT use a host header site identifier.</td>
</tr>
<tr>
<td>1</td>
<td>This site collection does use a host header site identifier.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This site collection was NOT upgraded from the previous version.</td>
</tr>
</tbody>
</table>
### Value | Description
---|---
1 | This site collection was upgraded from the previous version.

#### 3.1.5.72 proc_getDeletedSitesByQuery

The `proc_getDeletedSitesByQuery` stored procedure is called to return information about a set of site collections that have been marked for deletion. The members of the set are determined by the query parameters. The `proc_getDeletedSitesByQuery` stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_unhideSiteMap (  
  @ApplicationId     uniqueidentifier,  
  @SubscriptionId    uniqueidentifier,  
  @PathLike          nvarchar(256),  
  @DeletionTimeFrom  datetime,  
  @DeletionTimeTo    datetime,  
  @SiteId            uniqueidentifier,  
  @RowLimit          int,  
  @RequestGuid       uniqueidentifier = NULL OUTPUT  
);  
```

- **@ApplicationId**: Contains the configuration object identifier of the Web application.
- **@SubscriptionId**: The identifier of the site subscription. If this is a null value, do not use this parameter for selecting the result set of site collections.
- **@PathLike**: Contains the prefix of a server-relative URL of the site collections being queried for. If this is a null value, do not use this parameter for selecting the result set of site collections.
- **@DeletionTimeFrom**: One end of the range for the dates and times of the site collections that are marked for deletion and being queried.
- **@DeletionTimeTo**: The other end of the range for the dates and times of the site collections that are marked for deletion and being queried.
- **@SiteId**: The site collection identifier of the site collection being queried.
- **@RowLimit**: The maximum number of rows to return.
- **@RequestGuid**: The optional request identifier for the current request.

**Return Values**: The `proc_getDeletedSitesByQuery` stored procedure returns an integer return code which MUST be 0, indicating a successful execution.

**Result Set**: The `proc_getDeletedSitesByQuery` stored procedure MUST return a single result set specified in section 3.1.5.72.1.

#### 3.1.5.72.1 Site Result Set

The Site result set returns the site collections identified by **@ApplicationId**, **@SubscriptionId**, **@PathLike**, **@DeletionTimeFrom**, **@DeletionTimeTo**, and **@SiteId**. If **@SubscriptionId** is null, it must not be used to identify the site collections. If **@PathLike** is null, it must not be used to identify the site collections. The number of rows in the Site result set MUST NOT be greater than
the value of @RowLimit. If there are no site collections in the content database identified by the query parameters, the Site result set MUST be returned and MUST NOT contain any rows.

The Site result set is specified by the following T-SQL syntax.

```
Id                   uniqueidentifier,
SubscriptionId       uniqueidentifier,
DatabaseId           uniqueidentifier,
DeletionTime         datetime,
Path                 nvarchar(128),
HostHeaderIsSiteName bit,
Pairing              tinyint;
```

**Id:** The identifier of the site collection.

**SubscriptionId:** The identifier of the site subscription.

**DatabaseId:** the configuration object identifier of the content database.

**DeletionTime:** the date and time that the site collection was marked for deletion.

**Path:** the server-relative URL of the site collection.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This site collection does NOT use a host header site identifier.</td>
</tr>
<tr>
<td>1</td>
<td>This site collection does use a host header site identifier.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This site collection was NOT upgraded from the previous version.</td>
</tr>
<tr>
<td>1</td>
<td>This site collection was upgraded from the previous version.</td>
</tr>
</tbody>
</table>

### 3.1.5.73 proc_getDeletedSitesBySiteId

The proc_getDeletedSitesBySiteId stored procedure is called to return information about a set of site collections that have been marked for deletion. The members of the set are determined by the query parameters. The proc_getDeletedSitesBySiteId stored procedure is specified by the following T-SQL syntax.

```
PROCEDURE proc_getDeletedSitesBySiteId (  
    @ApplicationId     uniqueidentifier,  
    @SiteId            uniqueidentifier,  
    @RowLimit          int,  
    @RequestGuid       uniqueidentifier = NULL OUTPUT
);
```
@ApplicationId: Contains the configuration object identifier of the Web application.

@SiteId: The site collection identifier of the site collection being queried.

@RowLimit: The maximum number of rows to return.

@RequestGuid: The optional request identifier for the current request.

Return Values: The proc_getDeletedSitesBySiteId stored procedure returns an integer return code that MUST be 0, indicating a successful execution.

Result Set: The proc_getDeletedSitesBySiteId stored procedure MUST return a single result set specified in section 3.1.5.73.1.

3.1.5.73.1 Site Result Set

The Site result set returns the site collections identified by @ApplicationId and @SiteId. The number of rows in the Site result set MUST NOT be greater than the value of @RowLimit. If there are no site collections in the content database identified by @ApplicationId and @SiteId, the Site result set MUST be returned and MUST NOT contain any rows.

The Site result set is specified by the following T-SQL syntax.

```sql
Id                   uniqueidentifier,
SubscriptionId      uniqueidentifier,
DatabaseId           uniqueidentifier,
DeletionTime         datetime,
Path                 nvarchar(128),
HostHeaderIsSiteName bit,
Pairing              tinyint;
```

Id: The identifier of the site collection.

SubscriptionId: The identifier of the site subscription.

DatabaseId: the configuration object identifier of the content database.

DeletionTime: the date and time that the site collection was marked for deletion.

Path: the server-relative URL of the site collection.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This site collection does NOT use a host header site identifier.</td>
</tr>
<tr>
<td>1</td>
<td>This site collection does use a host header site identifier.</td>
</tr>
</tbody>
</table>

Pairing: Indicates whether the site collection was upgraded from previous version. This parameter MUST be one of the values in the following table.
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This site collection was NOT upgraded from the previous version.</td>
</tr>
<tr>
<td>1</td>
<td>This site collection was upgraded from the previous version.</td>
</tr>
</tbody>
</table>

### 3.1.5.74 proc_getDeletedSite

The **proc_getDeletedSite** stored procedure is called to return information about a site collection which have been marked for deletion. The **proc_getDeletedSite** stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_getDeletedSite (  
    @ApplicationId     uniqueidentifier,  
    @DatabaseId        uniqueidentifier,  
    @SiteId            uniqueidentifier,  
    @RequestGuid       uniqueidentifier = NULL OUTPUT  
);  
```

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

**@DatabaseId**: the configuration object identifier of the content database.

**@SiteId**: The site collection identifier of the site collection being queried.

**@RequestGuid**: The optional request identifier for the current request.

**Return Values**: The **proc_getDeletedSite** stored procedure returns an integer return code that MUST be 0, indicating a successful execution.

**Result Set**: The **proc_getDeletedSite** stored procedure MUST return a single result set specified in section 3.1.5.74.1.

### 3.1.5.74.1 Site Result Set

The **Site** result set returns the site collection identified by **@ApplicationId**, **@DatabaseId** and **@SiteId**. The number of rows in the **Site** result set MUST be 1 or 0. If there are no site collections in the content database identified by **@ApplicationId**, **@DatabaseId** and **@SiteId**, the **Site** result set MUST be returned and MUST NOT contain any rows.

The **Site** result set is specified by the following T-SQL syntax.

```sql
Id                   uniqueidentifier,  
SubscriptionId       uniqueidentifier,  
DatabaseId           uniqueidentifier,  
DeletionTime         datetime,  
Path                 nvarchar(128),  
HostHeaderIsSiteName bit,  
Pairing              tinyint;  
```

**Id**: The identifier of the site collection.

**SubscriptionId**: The identifier of the site subscription.
**DatabaseId**: the configuration object identifier of the content database.

**DeletionTime**: the date and time that the site collection was marked for deletion.

**Path**: the server-relative URL of the site collection.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This site collection does NOT use a host header site identifier.</td>
</tr>
<tr>
<td>1</td>
<td>This site collection does use a host header site identifier.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This site collection was NOT upgraded from the previous version.</td>
</tr>
<tr>
<td>1</td>
<td>This site collection was upgraded from the previous version.</td>
</tr>
</tbody>
</table>

### 3.1.5.75 proc_putDeletedSiteMap

The **proc_putDeletedSiteMap** stored procedure is called to create a new reference to a site collection that is marked for deletion. The **proc_putDeletedSiteMap** stored procedure is specified by the following T-SQL syntax.

```sql
PROCEDURE proc_putDeletedSiteMap (  
    @ApplicationId               uniqueidentifier,  
    @DatabaseId                  uniqueidentifier,  
    @SiteId                      uniqueidentifier,  
    @SubscriptionId              uniqueidentifier,  
    @Path                        nvarchar(128),  
    @Pairing                     tinyint,  
    @RedirectUrl                 nvarchar(512),  
    @HostHeaderIsSiteName        bit,  
    @DeletionTime                datetime,  
    @RequestGuid                 uniqueidentifier = NULL OUTPUT
);  
```

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

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

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

**@SubscriptionId**: Specifies the site subscription identifier.

**@Path**: If **@HostHeaderIsSiteName** has a value of 1, then **@Path** MUST contain the host header site identifier for the site collection. Otherwise, **@Path** MUST contain the 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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This site collection was NOT upgraded from the previous version.</td>
</tr>
<tr>
<td>1</td>
<td>This site collection was upgraded from the previous version.</td>
</tr>
</tbody>
</table>

**@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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>This site collection does NOT use a host header site identifier.</td>
</tr>
<tr>
<td>1</td>
<td>This site collection does use a host header site identifier.</td>
</tr>
</tbody>
</table>

**@DeletionTime:** the date and time the site collection was marked for deletion.

**@RequestGuid:** The optional request identifier for the current request.

**Return Values:** The `proc_putDeletedSiteMap` stored procedure returns an integer return code that MUST be a value listed in the following table.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>1</td>
<td>New marked for deletion Site Collection reference was not added successfully.</td>
</tr>
</tbody>
</table>

**Result Set:** The `proc_putDeletedSiteMap` 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 protocol client 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.

![Figure 2: Marking the e-mail enabled lists as deleted](image1)

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.

![Figure 3: Retrieving the e-mail aliases marked as deleted](image2)

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.
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 the protocol client 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 identifier, site identifier, name of the permission level, and name of the user who has requested creation of the distribution list, along with an optional request identifier.

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.

Figure 4: Removing the e-mail enabled lists

Figure 5: Adding a pending distribution list

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 a Pending Distribution List

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_dropPendingDistributionList stored procedure.

![Diagram](proc_dropPendingDistributionList)

*Figure 7: Removing the pending distribution lists*

4.3 Execution of 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 Content 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 with server identifier and time in minutes of the lock before it will be considered expired, as described in [MS-WSSCADM2].

![Diagram](proc_GetTimerLock)

*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 associated with the job definition.

4.3.2 Associate a Target Instance with a Job Definition

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 associated with a job definition by calling the proc_AddTimerTargetInstance stored procedure with the job identifier and target instance identifier.
4.3.3 Schedule a Job Instance

A schedule job instance is created to indicate when the protocol client will start a job instance. To create the scheduled job instance, the protocol client calls the `proc_ScheduleTimerJob` stored procedure with the service identifier, web application identifier, server identifier, job definition identifier and scheduled start time.

4.3.4 Start a Job Instance

When a job is marked as 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`.

If the job instance will not be run again in the future, then the protocol client deletes the scheduled job instance by calling `proc_DeleteScheduledJob` stored procedure with server identifier and job definition identifier:
4.3.5 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 13: Updating job progress](image)

4.3.6 Add a Job History Entry

After each target instance has been processed by the job instance, a job history entry should be created by calling `proc_AddTimerJobHistory` stored procedure.

![Figure 14: Adding job instance history](image)

4.3.7 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 15: Processing additional target instances](image)

4.3.8 Complete a Job Instance

Once the protocol client has finished processing the two content databases, the job instance may be finished by calling the `proc_CompleteTimerRunningJob` stored procedure with service identifier, web application identifier associated with the job definition, job definition identifier, server identifier and job status type of the 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 Store a File

File Storage is a multi-step operation which begins when the protocol client passes the GUID of an existing configuration object 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. The file being stored in this example is 75000 bytes. To illustrate the 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 `@Offset` is 50000.

4.4.2 Retrieve a File

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_putFileSegment` stored procedure to store the file is used here to retrieve the contents of that file.
4.5 Site Subscription Retrieval

This example follows the sequence of calls needed to retrieve site subscriptions on a farm, or in a given Web application or content database on a protocol server. Using the retrieved site subscription identifier this example follows with the calls needed to find all site collections in a farm, or given Web application or content database that belong to the specified site subscription. An existing site subscription with one or more associated site collections is assumed.

4.5.1 Retrieve Site Subscriptions

The protocol client retrieves a list of site subscriptions in a farm by calling the `proc_getSubscriptionIds` stored procedure. This returns the list of site subscriptions in a farm.

The protocol client can also retrieve a list of site subscriptions in a Web application on the protocol server by calling the `proc_getSiteSubscriptionsInWebApplication` stored procedure. This returns a list of site subscriptions that have site collections in the Web application.

The protocol client can also retrieve a list of site subscriptions in a content database on the protocol server by calling the `proc_getSiteSubscriptionsInContentDatabase` stored procedure. This returns a list of site subscriptions with site collections in the content database.
4.5.2 Retrieve Site Collections

Once the list of site subscription identifiers are retrieved for a Farm, they can be used to retrieve all site collections associated with a site subscription in a farm by calling the `proc_getSiteIds` stored procedure. This returns a list of site collection identifiers for the given site subscription in the entire farm.

The list of site subscription identifiers can also be used to retrieve all site collections in a site subscription contained in a Web application by calling the `proc_getSiteSubscriptionSiteIdsInWebApplication` stored procedure. This returns a list of site collection identifiers for the given site subscription in the given Web application.

The list of site subscription identifiers can also be used to retrieve all site collections in a site subscription contained in a content database by calling the `proc_getSiteSubscriptionSiteIdsInContentDatabase` stored procedures. This returns a list of site collection identifiers for the given site subscription in the given content database.
Figure 24: proc_getSiteSubscriptionSiteIdsInContentDatabase
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® SharePoint® Foundation 2010
- Microsoft® SQL Server® 2005
- Microsoft® SQL Server® 2008
- Microsoft® SQL Server® 2008 R2

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

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

No table of changes is available. The document is either new or has had no changes since its last release.
Index

A
Abstract data model  
server 15  
Acquire a content database lock example 78  
Add a job history entry example 80  
Add a pending distribution list example 77  
Applicability 10  
Associate a target instance with a job definition example 78

B
Binary structures - overview 12  
Bit fields  
job lock type 11  
job status type 11  
lock status type 12

C
Capability negotiation 10  
Change tracking 87  
Classes  
configuration object 12  
Client  
overview 15  
Common data types  
overview 11  
Complete a job instance example 80  
Configuration object classes 12  
Configuration object management 9  
server 15  
Configuration object properties 12

D
Data model - abstract  
server 15  
Data types  
common 11  
Data types - simple  
overview 11  
Deletion of e-mail-enabled lists from a site collection example  
mark lists as deleted 76  
overview 76  
remove e-mail enabled lists 76  
retrieve e-mail aliases marked as deleted 76

E
E-mail-enabled list 13  
E-mail-enabled lists 9  
server 17  
Events  
local - server 75  
timer - server 75  
Examples  
acquire a content database lock 78  
add a job history entry 80  
add a pending distribution list 77  
associate a target instance with a job definition 78  
complete a job instance 80  
delete e-mail-enabled lists from a site collection 76  
execute a job instance 78  
mark e-mail enabled lists as deleted 76  
monitor pending distribution lists 77  
process additional target instances 80  
remove a pending distribution list 78  
remove e-mail-enabled lists 76  
retrieve a file 81  
retrieve e-mail aliases marked as deleted 76  
retrieve pending distribution lists 77  
retrieve site collections 83  
retrieve site subscriptions (section 4.5, section 4.5.1 82)  
schedule a job instance 79  
start a job instance 79  
store a file 81  
store and retrieve files 81  
update job progress 80  
Execution of a job instance example  
acquire a content database lock 78  
add a job history entry 80  
associate a target instance with a job definition 78  
complete a job instance 80  
overview 78  
process additional target instances 80  
schedule a job instance 79  
start a job instance 79  
update job progress 80

F
Fields - vendor-extensible 10  
File retrieval example 81  
File storage 9  
File storage and retrieval example  
overview 81  
retrieve a file 81  
store a file 81  
File storage example 81  
Flag structures  
job lock type bit field 11  
job status type bit field 11  
lock status type bit field 12

G
Glossary 7

H
Higher-layer triggered events
Index of security parameters

- Implementer - security considerations: 85
- Index of security parameters: 85
- Informative references: 8

Initialization

- server: 17
- Introduction: 7

J

- Job lock type bit field: 11
- Job status type bit field: 11

L

- Local events: server: 75
- Lock status type bit field: 12

M

- Mark e-mail-enabled lists as deleted example: 76
- Message processing: server: 17
- Messages:
  - binary structures: 12
  - common data types: 11
  - enumerations: 11
  - job lock type bit field: 11
  - job status type bit field: 11
  - lock status type bit field: 12
  - result sets: 12
  - simple data types: 11
  - table structures: 14
  - transport: 11
  - view structures: 14
  - XML structures: 14
- Methods:
  - proc_AddTimerJobHistory: 17
  - proc_AddTimerLockForJob: 18
  - proc_AddTimerTargetInstance: 19
  - proc_ClearProductVersions: 20
  - proc_CompleteTimerRunningJob: 20
  - proc_DeleteAllMarkedTimerLocks: 21
  - proc_DeleteAllTimerLocksAndRunningJobs: 21
  - proc_DeleteOldTimerJobHistory: 22
  - proc_DeleteScheduledJob: 22
  - proc_DeleteTimerLockForJob: 23
  - proc_DeleteTimerRunningJobs: 24
  - proc_DeleteTimerTargetInstance: 24
  - proc_DeleteTimerTargetInstances: 25
  - proc_dropClass: 26
  - proc_dropDeletedEmailAliases: 29
  - proc_dropDeletedSite: 73
  - proc_dropDeletedSiteByQuery: 73
  - proc_dropDeletedSitesBySiteId: 73
  - proc_dropDeletedSitesByUrl: 73
  - proc_GetDependentObjectsByBaseClass: 30
  - proc_getEmailEnabledListByAlias: 31
  - proc_getFile: 31
  - proc_getNewObjects: 32
  - proc_getPendingDistributionListByAlias: 31
  - proc_getPendingDistributionListsSinceVersion: 34
  - proc_getProductVersions: 35
  - proc_getSiteBestMatch: 36
  - proc_getSiteCount: 38
  - proc_getSiteIdOfHostHeaderSite: 38
  - proc_getSiteIds: 39
  - proc_getSiteNames: 39
  - proc_getSiteSubscriptionsInContentDB: 41
  - proc_getSiteSubscriptionsInWebApplication: 41
  - proc_getSiteSubscriptionSiteIdsInContentDatabase: 42
  - proc_getSiteSubscriptionSiteIdsInWebApplication: 43
  - proc_getSiteSubset: 44
  - proc_getSubscriptionIds: 44
  - proc_getTemplate: 45
  - proc_GetTimerJobHistory: 46
  - proc_GetTimerJobLastRunTime: 48
  - proc_GetTimerRunningJobs: 48
  - proc_GetTimerScheduledJobs: 50
  - proc_GetTimerTargetInstance: 51
  - proc_GetTimerTargetInstanceState: 51
  - proc_hideSiteMap: 67
  - proc_markForDeletionEmailEnabledList: 52
  - proc_markForDeletionEmailEnabledListsBySite: 52
  - proc_markForDeletionEmailEnabledListsByWeb: 52
  - proc_markForDeletionEmailEnabledListsBySite: 52
  - proc_putClass: 53
  - proc_putDeletedSiteMap: 74
  - proc_putDependency: 54
  - proc_putDistributionListToDelete: 54
  - proc_putEmailEnabledList: 55
  - proc_putFileSegment: 55
  - proc_putObject: 56
  - proc_putPendingDistributionList: 58
  - proc_putSiteMap: 58
  - proc_refreshAllTimerLocks: 60
  - proc_RegisterProductVersion: 60
  - proc_renameSiteMap: 61
  - proc_ScheduleTimerJob: 62
  - proc_setSubscription: 63
  - proc_siteExists: 63
  - proc_startTimerRunningJob: 63
  - proc_unhideSiteMap: 67
  - proc_UpdateTimerRunningJobProgress: 64
  - proc_UpdateTimerRunningJobStatus: 65
  - proc_UpdateTimerRunningJobTarget: 66
  - proc_UpdateTimerTargetInstanceState: 67

N

- Normative references: 8
O
Overview (synopsis) 9

P
Parameters - security index 85
Pending distribution list 13
Pending distribution list example 85
  add a pending distribution list 77
  overview 77
  remove a pending distribution list 78
  retrieve pending distribution lists 77
Pending distribution lists 9
server 17
Preconditions 10
Prerequisites 10
proc_AddTimerJobHistory method 17
proc_AddTimerLockForJob method 18
proc_AddTimerTargetInstance method 19
proc_ClearProductVersions method 20
proc_CompleteTimerRunningJob method 20
proc_DeleteAllMarkedTimerLocks method 21
proc_DeleteAllTimerLocksAndRunningJobs method 21
proc_DeleteOldTimerJobHistory method 22
proc_DeleteScheduledJob method 23
proc_DeleteTimerLockForJob method 23
proc_DeleteTimerRunningJobs method 24
proc_DeleteTimerTargetInstance method 25
proc_DeleteTimerTargetInstances method 25
proc_dropClass method 26
proc_dropEmailEnabledList method 27
proc_dropEmailEnabledListByAlias method 27
proc_DropObject method 28
proc_dropPendingDistributionList method 28
proc_DropSiteMap method 29
proc_dropSiteMap2 method 68
proc_getDeletedEmailAliases method 29
proc_getDeletedSite method 73
proc_getDeletedSitesByQuery method 70
proc_getDeletedSitesBySiteId method 71
PROC_getDeletedSitesByUrl method 68
proc_GetDependentObjectsByBaseClass method 30
proc_getEmailEnabledListByAlias method 31
proc_getFile method 31
proc_getNewObjects method 32
proc_getPendingDistributionListsSinceVersion method 34
proc_GetProductVersions method 35
proc_getSiteBestMatch method 36
proc_getSiteCount method 38
proc_getSiteIdOfHostHeaderSite method 38
proc_getSiteIds method 39
proc_getSiteNames method 39
proc_getSiteSubscriptionsInContentDB method 41
proc_getSiteSubscriptionsInWebApplication method 41
PROC_getSiteSubscriptionSiteIdsInWebApplication method 43
PROC_getSiteSubscriptionSiteIdsInWebApplication method 44
PROC_getSubscriptionIds method 44
PROC_getTemplate method 45
PROC_GetTimerJobHistory method 46
PROC_GetTimerJobLastRunTime method 48
PROC_GetTimerRunningJobs method 48
PROC_GetTimerScheduledJobs method 50
PROC_GetTimerTargetInstance method 51
PROC_GetTimerTargetInstanceState method 51
PROC_hideSiteMap method 67
PROC_markForDeletionEmailEnabledList method 52
PROC_markForDeletionEmailEnabledListBySite method 52
PROC_markForDeletionEmailEnabledListByWeb method 53
PROC_putClass method 53
PROC_putDeletedSiteMap method 74
PROC_putDependency method 54
PROC_putDistributionListToDelete method 54
PROC_putEmailEnabledList method 55
PROC.PutFileSegment method 55
PROC.PutObject method 56
PROC_putPendingDistributionList method 58
PROC_putSiteMap method 58
PROC_RefreshAllTimerLocks method 60
PROC_RegisterProductVersion method 60
PROC_renameSiteMap method 61
PROC_ScheduleTimerJob method 62
PROC_setSubscription method 63
PROC_SiteExists method 63
PROC_StartTimerRunningJob method 63
PROC_unhideSiteMap method 67
PROC_UpdateTimerRunningJobProgress method 64
PROC_UpdateTimerRunningJobStatus method 65
PROC_UpdateTimerTargetInstance method 66
PROC_UpdateTimerTargetInstanceState method 67
Process additional target instances example 80
Product behavior 86
Properties
  configuration object 12

R
References 8
  informative 8
  normative 8
Relationship to other protocols 10
Remove a pending distribution list example 78
Remove e-mail-enabled lists example 76
Result sets - overview 12
Retrieve a file example 81
Retrieve e-mail aliases marked as deleted example 76
Retrieve pending distribution lists example 77
Retrieve site collections example 83
Retrieve site subscriptions example 82

S
Schedule a job instance example 79
Security

implementer considerations 85
parameter index 85

Sequencing rules

Server

abstract data model 15
configuration object management 15
e-mail-enabled lists 17
higher-layer triggered events 17
initialization 17
local events 75
message processing 17
overview 15
pending distribution lists 17
proc_AddTimerJobHistory method 17
proc_AddTimerLockForJob method 18
proc_AddTimerTargetInstance method 19
proc_ClearProductVersions method 20
proc_CompleteTimerRunningJob method 20
proc_DeleteAllMarkedTimerLocks method 21
proc_DeleteAllTimerLocksAndRunningJobs method 21
proc_DeleteOldTimerJobHistory method 22
proc_DeleteScheduledJob method 23
proc_DeleteTimerLockForJob method 23
proc_DeleteTimerRunningJobs method 24
proc_DeleteTimerTargetInstance method 25
proc_DeleteTimerTargetInstances method 25
proc_dropClass method 26
proc_dropEmailEnabledList method 27
proc_dropEmailEnabledListByAlias method 27
proc_dropEmailEnabledListsByWeb method 27
proc_DropObject method 28
proc_DropPendingDistributionList method 28
proc_DropSiteMap method 29
proc_DropSiteMap2 method 68
proc_getDeletedEmailAlias method 29
proc_getDeletedSite method 73
proc_getDeletedSitesByQuery method 70
proc_getDeletedSitesBySiteId method 71
proc_getDeletedSitesByUrl method 68
proc_GetDependentObjectsByBaseClass method 30
proc_getEmailEnabledListByAlias method 31
proc_getEmailEnabledList method 31
proc_getFile method 32
proc_getPendingDistributionLists method 34
proc_GetProductVersions method 35
proc_getSiteBestMatch method 36
proc_getSiteCount method 38
proc_getSiteIdOfHostHeaderSite method 38
proc_getSiteIds method 39
proc_getSiteNames method 39
proc_getSiteSubscriptionsInContentDB method 41
proc_getSiteSubscriptionsInWebApplication method 41
proc_getSiteSubscriptionSiteIdsInWebApplication method 43
proc_getSiteSubscription method 44
proc_getSubscriptionIds method 44
proc_getTemplate method 45
proc_GetTimerJobHistory method 46
proc_GetSubscriptionLastRunTime method 48
proc_GetTimerRunningJobs method 48
proc_GetTimerScheduledJobs method 50
proc_GetTimerTargetInstance method 51
proc_GetTimerTargetInstances method 51
proc_hideSiteMap method 67
proc markForDeletionEmailEnabledList method 52
proc_markForDeletionEmailEnabledListsBySite method 52
proc_markForDeletionEmailEnabledListsByWeb method 53
proc_putClass method 53
proc_putDeletedSiteMap method 74
proc_putDependency method 54
proc_putDistributionListToDelete method 54
proc_putEmailEnabledList method 55
proc_putFileSegment method 55
proc_putObject method 56
proc_putPendingDistributionList method 58
proc_putSiteMap method 58
proc_RefreshAllTimerLocks method 60
proc_RegisterProductVersion method 60
proc_renameSiteMap method 61
proc_ScheduleTimerJob method 62
proc_setSubscription method 63
proc_SiteExists method 63
proc_StartTimerRunningJob method 63
proc_unregisterSite method 63
proc_unregisterTimerRunningJob method 64
proc_unregisterTimerRunningJobTarget method 66
proc_unregisterTimerTargetInstance method 67
proc_unregisterTimerTargetInstanceState method 67
proc_unregisterTimerTargetInstanceState method 67
sequencing rules 17
timer events 75
timer job management 16
	timers 17
Simple data types
overview 11
Site subscription retrieval example
overview 82
retrieve site collections 83
retrieve site subscriptions 82
Standards assignments 10
Start a job instance example 79
Store a file example 81
Structures
binary 12
Table and view 14
XML 14
T

Table structures – overview 14
Timer events