[MS-WSSCADM]:
Windows SharePoint Services Content Database Administrative Communications Protocol Specification

---

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability
04/25/2008	0.2	Editorial	Revised and edited the technical content
06/27/2008	1.0	Major	Revised and edited the technical content
12/14/2008	1.01	Editorial	Revised and edited the technical content
07/13/2009	1.02	Major	Changes made for template compliance
08/28/2009	1.03	Editorial	Revised and edited the technical content
11/06/2009	1.04	Editorial	Revised and edited the technical content
02/19/2010	2.0	Editorial	Revised and edited the technical content
03/31/2010	2.01	Editorial	Revised and edited the technical content
04/30/2010	2.02	Editorial	Revised and edited the technical content
06/07/2010	2.03	Editorial	Revised and edited the technical content
06/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.05	Editorial	Changed language and formatting in the technical content.
03/18/2011	2.05	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	2.05	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

The Windows SharePoint Services Content Database Administrative Communications protocol specifies the communication sequences used by front-end Web servers and application servers to perform administrative operations on a back-end database server related to content databases. This includes recycle bin, quota, database integrity and maintenance, and auditing operations.

1.1 Glossary

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

anonymous user
Coordinated Universal Time (UTC)
GUID
language code identifier (LCID)

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

absolute URL
Active Directory account creation mode
All Site Members
attachment
audit entry
audit event
audit log
author
back-end database server
configuration database
content database
content database lock
content type
current user
current version
datetime
delete transaction
delete transaction identifier
deleted
directory name
dirty
display name
document
document identifier
document library
document version
domain account mode
e-mail address
event
external security provider
farm
field
file
first-stage Recycle Bin
folder
forward link
front-end Web server

full URL
group
item
item identifier
leaf name
list
list identifier
list item
list schema
list template
list template identifier
locked
login name
notify count
object model
page
parent list
parent site
permission
permission level
publishing level
quota template identifier
quota warning
quota warning level
Recycle Bin
Recycle Bin item
regional settings
result set
return code
role
root folder
row
scope identifier
second-stage Recycle Bin
securable object
security group
security principal
security role
security scope
server-relative URL
site
site certification
site collection
site collection administrator
site collection identifier
site identifier
site template
stored procedure
store-relative form
subsite
survey list
top-level site
Transact-Structured Query Language (T-SQL)
Uniform Resource Locator (URL)
user identifier

Web Part
Web Part Page
workflow
zero-based index

The following terms are specific to this document:

first unique ancestor: The root securable object of the security scope to which a securable object belongs.

orphaned object: A content database object that lacks a requisite relationship to a corresponding object.

quota template: A group of default quotas that can be applied to a site collection. It is stored in the configuration database.

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

1.2 References

1.2.1 Normative References

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

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

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

[MS-TDS] Microsoft Corporation, "Tabular Data Stream Protocol Specification".

[MS-WSSDLIM] Microsoft Corporation, "Windows SharePoint Services: Content Database Document and List Item Management Communications Protocol Specification"

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

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

1.2.2 Informative References

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

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

1.3 Protocol Overview (Synopsis)

This protocol specifies the communication between the **front-end Web server** and the **back-end database server** used to satisfy requests involving management and administration of **content databases**. This client-to-server protocol uses the Tabular Data Stream Protocol, as described in [MS-TDS], as its transport between the front-end Web server, acting as a client, and the back-end database server, acting as a server.

Content Database Administrative Communications Protocol is composed of five major operation areas as follows.

1.3.1 Auditing Operations

A common business requirement is that a protocol server be able to provide a recorded history of operations performed on various objects stored on the protocol server, such as when and by whom an object was viewed or modified. This recorded history can be used for the purposes of forensic monitoring ("auditing") to verify conformance of the operations performed on the objects to business requirements. This recorded history of **audit entries** is generally persisted during and potentially after the end-of-life of the objects, and as such is generally stored in an **audit log**.

The auditing operations allow protocol clients to determine and configure settings on objects that specify which operations performed on those objects are to be recorded. The protocol further provides methods for protocol clients to record audit entries to an audit log.

1.3.2 Quota Management Operations

Quota management allows the administrators to set a quota for a **site collection**, create **quota templates** for use on many site collections, and get a list of site collections that are near their quota limits. A site collection quota is set to block any updates of existing content or additions of new content or users to a site collection that has reached its quota limit.

1.3.2.1 Query and Update Quota Operations

The back-end database server provides methods for the client to query and update quota information for site collections. Quota information consists of data about disk space allowed in Bytes, and the number of users allowed per site collection. When client requests for quota information are sent to the front-end Web server, the front-end Web server sends a series of **stored procedure** calls to the back-end database server for the requested information. The stored procedures return data which in turn can be used for further calls to other stored procedures. The front-end Web server turns the values in the **return codes** and **result sets** of the stored procedures into objects which contain the data requested by the client and uses the objects according to implementation-specific procedures.

1.3.2.2 Query and Update Usage Operations

The back-end database server provides methods for the client to query and update usage information for a site collection. Usage information consists of data about actual disk space used, in Bytes, by various types of content in a site collection including **documents**, **document libraries**, **lists**, and the **Recycle Bin**. When client requests for usage information are sent to the front-end Web server, the front-end Web server sends a series of stored procedure calls to the back-end database server for the requested information. The stored procedures return data which in turn can be used for further calls to other stored procedures. The front-end Web server turns the values in the return codes and result sets of the stored procedures into objects which contain the data requested by the client and uses the objects according to implementation-specific procedures.

1.3.2.3 Query Warn Operations

The back-end database server provides methods for the client to query site collections whose actual disk space used has crossed the warning limits set in the quota. The front-end Web server sends a series of stored procedure calls to the back-end database server to get a list of site collections which have crossed the warning limits for actual disk space used. The stored procedures return data which in turn can be used for further calls to other stored procedures. The front-end Web server turns the values in the return codes and result sets of the stored procedures into objects which contain the data requested by the client and uses the objects according to implementation-specific procedures to send notifications to the **site collection administrators** of those site collections. Next, the front-end Web server sends a series of stored procedure calls to mark the notified site collections as already been notified.

1.3.3 Recycle Bin Operations

The Recycle Bin allows users and site collection administrators to restore **items** back to their original location. The **first-stage Recycle Bin** allows users to restore their items, and the **second-stage Recycle Bin** allows site collection administrators in case the items can no longer be restored from the first-stage Recycle Bin.

1.3.3.1 Query Operations

The back-end database server provides methods to query for all **Recycle Bin items** in the first-stage Recycle Bin for a specific user in a specific **site** or for all Recycle Bin items in the second-stage Recycle Bin in a site collection when the user is a site collection administrator. The front-end Web server sends a series of stored procedure calls to the back-end database server for the requested information. The stored procedures return data which in turn are used for further calls to other stored procedures. The front-end Web server turns the values in the return codes and result sets of the stored procedures into objects which contain the data requested by the client and uses the objects according to implementation-specific procedures.

1.3.3.2 Administration Operations

There are two sets of administration operations. The first set is used to move items from a site collection through the different Recycle bins. The second set is used to restore items from the Recycle Bins back to their original location in the site collection.

1.3.3.2.1 Delete Operations

The back-end database server provides methods to delete items from a site collection. These operations move the **deleted** items into the first-stage Recycle Bin. The back-end database server also provides methods to delete items from the Recycle Bins. Deleting items in the first-stage Recycle Bin moves the items to the second-stage Recycle Bin. Deleting items in the second-stage Recycle Bin removes them in a way that they no longer exist on disk on the back-end database server.

Each individual **delete transaction** is identified by a **delete transaction identifier**, which is unique in a content database. All items in the same delete transaction are assigned the same delete transaction identifier value.

The front-end Web server sends a series of stored procedure calls to the back-end database server to delete items from a site collection or the Recycle Bins. The stored procedures return data which in turn are used for further calls to other stored procedures. The front-end Web server turns the values in the return codes and result sets of the stored procedures into objects which contain the data requested by the client and uses the objects according to implementation-specific procedures.

1.3.3.2.2 Restore Operations

The back-end database server provides methods to restore all Recycle Bin items from a single delete transaction. Items are restored from the first-stage Recycle Bin or from the **second-stage Recycle Bin** to their original locations in the site collection. Items in the **second-stage Recycle Bin** can only be restored by a site collection administrator.

The front-end Web server sends a series of stored procedure calls to the back-end database server to restore items from the Recycle Bins. The stored procedures return data which in turn are used for further calls to other stored procedures. The front-end Web server turns the values in the return codes and result sets of the stored procedures into objects which contain the data requested by the client and uses the objects according to implementation-specific procedures.

1.3.4 Security Operations

Security operations satisfy requests involving file access and administration of users and **Groups** within the system.

1.3.4.1 Operations related to External Security Provider

This protocol allows **permissions** on a site to be enforced by an **external security provider**. The external security provider can also be removed to allow to the client manage permissions directly.

1.3.4.2 Operations related to ACL

This protocol provides methods for retrieving information about access control lists (ACL) and WSS Rights Masks, as defined in [MS-WSSFO] section 2.2.2.13, for **anonymous users** on **securable objects** such as sites, lists, and **list items**.

1.3.4.3 User and Group Operations

This protocol provides methods for retrieving information about individual users and groups. When the **object model** on the front-end Web server operates on requests to query or update users or groups, the front-end Web server confirms if the data is already populated in the local objects that represent the specific user or group, and if it does not exist, it sends a series of stored procedure calls to the back-end database server for the requested information. The stored procedures return data which in turn are used for further calls to other stored procedures. The front-end Web server turns the values in the return codes and result sets of the stored procedures into objects that contain the data and metadata for the requested users or groups and uses the objects according to implementation specific procedures.

1.3.5 Database Integrity and Maintenance Operations

The database integrity and maintenance operations allow protocol clients to perform maintenance activities on the content database such as detecting and fixing **orphaned objects**.

1.3.5.1 Orphaned Objects Management

The back-end database server provides methods for the client to look for objects in the content database that are orphaned and, optionally, fix them.

1.3.5.2 Dead Web Management

The back-end database server provides methods for the client to enumerate through **top-level sites** and collect information such as the number of days that have elapsed between the last **site**

certification and the current date. This information can be used to determine which site collections have been inactive so that they can be deleted if desired.

1.3.5.3 Maintenance Operations

The back-end database server provides methods for dealing with database integrity issues such as deleting a corrupted list, a list with no **parent site**, or a list with items without a **parent list**.

1.4 Relationship to Other Protocols

The following diagram shows the transport stack that the protocol uses:

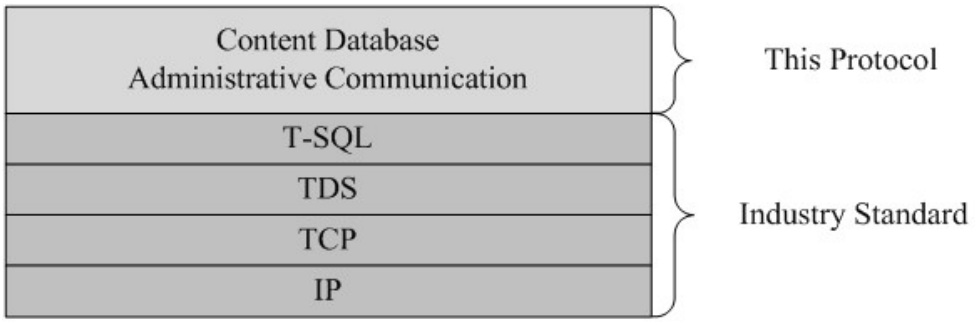

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

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

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

1.6 Applicability Statement

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

1.7 Versioning and Capability Negotiation

- **Security and Authentication Methods:** This protocol supports the SSPI and SQL Authentication with the Protocol Server role 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, return result codes, and return result sets.

2.2 Common Data Types

The following sections define the common data types that are used in this protocol.

2.2.1 Simple Data Types and Enumerations

2.2.2 Simple Data Types

2.2.2.1 Audit Event Source

A 1-byte unsigned integer enumeration specifying the source of the audit entry. This MUST be a value specified as follows:

Value	Meaning
0x00	The audit entry is generated by the server code internally.
0x01	The audit entry is generated by object model code.

2.2.2.2 Audit Event Type

A 4-byte unsigned integer enumeration specifying the type of operation that generated the **audit event**. This MUST be a value specified as follows:

Value	Meaning
0x00000001	The audit event was generated when a document was checked out.
0x00000002	The audit event was generated when a document was checked in.
0x00000003	The audit event was generated when an object was viewed.
0x00000004	The audit event was generated when an object was deleted.
0x00000005	The audit event was generated when an object was updated.
0x00000006	The audit event was generated when a **content type** was updated.
0x00000007	The audit event was generated when a child object was deleted.
0x00000008	The audit event was generated when a **list schema** changed.
0x0000000A	The audit event was generated when an object was undeleted.
0x0000000B	The audit event was generated by **workflow**.
0x0000000C	The audit event was generated when an object was copied.
0x0000000D	The audit event was generated when an object was moved.

Value	Meaning
0x0000000E	The audit event was generated when the audit flags, as described in [MS-WSSFO], of an object were updated.
0x0000000F	The audit event was generated when a search operation was performed.
0x00000010	The audit event was generated when a child object was moved.
0x0000001E	The audit event was generated when a **security group** was created.
0x0000001F	The audit event was generated when a security group was deleted.
0x00000020	The audit event was generated when a **security principal** was added to a security group.
0x00000021	The audit event was generated when a security principal was removed from a security group.
0x00000022	The audit event was generated when a **security role** was created.
0x00000023	The audit event was generated when a security role was deleted.
0x00000024	The audit event was generated when a security role was updated.
0x00000025	The audit event was generated when a security role breaks inheritance.
0x00000026	The audit event was generated when a **security scope** was updated.
0x00000027	The audit event was generated when a security scope restores inheritance.
0x00000028	The audit event was generated when a security scope breaks inheritance.
0x00000032	The audit event was generated when audit events was deleted.
0x00000064	The audit event was generated by a custom operation.

2.2.2.3 Delete Item Type

A 32-byte signed integer value indicating the type of the Recycle Bin item. It MUST be one of the following values:

Value	Description
1	The item is a document.
2	The item is a **document version** of a document.
3	The item is a list item.
4	The item is a list.
5	The item is a **folder**.
6	The item is a folder which contains lists.
7	The item is an **attachment**.
8	The item is a version of a list item.

2.2.2.4 Recycle Bin Stage

A 1-byte signed integer value indicating the stage of the Recycle Bin. It MUST be one of the following values:

Value	Description
1	First-stage Recycle Bin
2	Second-stage Recycle Bin

2.2.3 Bit Fields and Flag Structures

None.

2.2.4 Enumerations

None.

2.2.5 Binary Structures

None.

2.2.6 Common Result Sets

The following common result sets are used by this protocol.

2.2.6.1 Site Collection with no Sites Result Set

The Site Collection with no Sites result set returns site collections without sites. The Site Collection with no Sites result set MUST return a result set having a number of rows equal to the number of site collections without sites. If there are no such site collections, it returns zero rows. The **Transact-Structured Query Language (T-SQL)** syntax for the result set is as follows:

```
Id    uniqueidentifier;
```

Id: The **site collection identifier** of the site collection with no sites. Id MUST NOT be NULL.

2.2.6.2 Sites with no Site Collection Result Set

The Sites with no Site Collection result set returns sites with no site collection and no associated folder. The Sites with no Site Collection result set MUST return a result set having a number of rows equal to the number of sites with no site collection. If there are no such sites, it returns zero rows. The T-SQL syntax] for the result set is as follows:

```
Id          uniqueidentifier,
Title       nvarchar(255),
SiteId      uniqueidentifier,
FullUrl     nvarchar(256);
```

Id: The **site identifier** of the site with no site collection. Id MUST NOT be NULL.

Title: The **display name** of the site.

SiteId: The site collection identifier of the site collection which contains the site. SiteId MUST NOT be NULL.

FullUrl: The **store-relative form** of the site. FullUrl MUST NOT be NULL.

2.2.6.3 Sites with no Parent Site Result Set

The Sites with no Parent Site result set returns those sites that have no parent site. The Sites with no Parent Site result set MUST return a number of rows equal to the number of sites with no parent site. If there are no such sites, it returns zero rows. The T-SQL syntax for the result set is as follows:

```
Id          uniqueidentifier,
Title       nvarchar(255),
SiteId      uniqueidentifier,
FullUrl     nvarchar(256);
```

Id: The site identifier of the site with no parent site. Id MUST NOT be NULL.

Title: The display name of the site.

SiteId: The site collection identifier of the site collection which contains the site. SiteId MUST NOT be NULL.

FullUrl: The store-relative form of the site. FullUrl MUST NOT be NULL.

2.2.6.4 Folders with no Site Result Set

The Folders with no Site result set returns orphaned folders at the root of the site that have no associated site. The Folders with no Site result set MUST return a result set having a number of rows equal to the number of folders with no site. If there are no such folders, it returns zero rows. The T-SQL syntax for the result set is as follows:

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

WebId: The site identifier of the site which contains the orphaned folder. WebId MUST NOT be NULL.

SiteId: The site collection identifier of the site collection which contains the orphaned folder. SiteId MUST NOT be NULL.

2.2.6.5 Orphaned Lists Result Set

The Orphaned Lists result set returns orphaned lists in the following three categories:

- Lists with no parent site
- Lists with documents that have no parent list
- Lists with items that have no parent list

The Orphaned Lists result set MUST return a result set having a number of rows equal to the number of orphaned lists. If there are no such lists, the result set returns zero rows. The T-SQL syntax for the result set is as follows:

```
ListId      uniqueidentifier,
Title       nvarchar(255),
WebId       uniqueidentifier,
SiteId      uniqueidentifier;
```

ListId: The **list identifier** of an orphaned list.

Title: The display name of the list. Title MUST be NULL if the result is in either "Lists with documents with no Parent List" or "Lists with items with no Parent List" categories.

WebId: The site identifier of the site which contains the list. WebID MUST be NULL if the result is in either "Lists with documents with no Parent List" or "Lists with items with no Parent List" categories.

SiteId: The site collection identifier of the site collection that contains the list. SiteId MUST be NULL if the result is in "Lists with no Parent Site" category.

2.2.6.6 User Storage Info Result Set

The User Storage Info result set returns the **user identifier**, **login name**, and the total size, in bytes, of the personalizations and **Web Parts** on a particular **Web Part Page**. There is one row returned for each user that customizes the Web Part page.

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

```
tp_Id          int,
tp_Login       nvarchar(255),
Size           bigint;
```

tp_Id: The user identifier.

Tp_Login: The login name of the user.

Size: The size, in Bytes, of both the personalization and Web Parts on a particular Web Part page.

2.2.7 SQL Structures

None.

2.2.8 Tables and Views

This section describes the tables and views used in this protocol.

2.2.8.1 AuditData

The AuditData table stores audit entries. The table is defined using T-SQL syntax, as follows:

```
TABLE AuditData(
  SiteId            uniqueidentifier NOT NULL,
  ItemId            uniqueidentifier NOT NULL,
  ItemType          smallint NOT NULL,
  UserIdint         NULL,
  MachineName       nvarchar(128) NULL,
  MachineIp         nvarchar(20) NULL,
  DocLocation       nvarchar(260) NULL,
```

```
  LocationType      tinyint NULL,
  Occurred          datetime NOT NULL,
  Event             int NOT NULL,
  EventName         nvarchar(128) NULL,
  EventSource       tinyint NOT NULL,
  SourceName        nvarchar(256) NULL,
  EventData         ntext NULL
);
```

SiteId: The site collection identifier of the **site collection** which contains the object for which the audit entry was added.

ItemId: The identifier of the object.

ItemType: The type of object, which MUST be a value from the [MS-WSSFO] section 2.2.3.2.

UserId: The user identifier of the user who performed the operation that generated the audit entry.

MachineName: Reserved. MUST be NULL.

MachineIp: Reserved. MUST be NULL.

DocLocation: The location of the object, which MUST be a store-relative form or a **leaf name**.<1>

LocationType: Reserved. MUST be 0x00**.**

Occurred: The date and time, in **Coordinated Universal Time (UTC)**, when the operation occurred.

Event: The type of audit event, which MUST be a value from the Audit Event Type enumeration.

EventName: The name of the audit event.

EventSource: The source of the audit event, which MUST be a value from the Audit Event Source enumeration.

SourceName: The display name of the audit event source.

EventData: The **event** data of the audit event.

2.2.8.2 RecycleBin Table

The RecycleBin table stores the descriptions and properties of items in the **Recycle Bin for all site collections** in the current database. The table is defined using T-SQL syntax as follows:

```
TABLE [dbo].[RecycleBin]
(
  SiteId                uniqueidentifier NOT NULL,
  WebId                 uniqueidentifier NOT NULL,
  BinId                 tinyint NOT NULL,
  DeleteUserId          int NOT NULL,
  DeleteTransactionId   varbinary(16) NOT NULL,
  DeleteDate            datetime NOT NULL,
  ItemType              tinyint NOT NULL,
  ListId                uniqueidentifier NULL,
  DocId                 uniqueidentifier NULL,
  DocVersionId          int NULL,
```

```
    ListItemId            int NULL,
    Title                 nvarchar(260) NOT NULL,
    DirName               nvarchar(256) NOT NULL,
    LeafName              nvarchar(128) NOT NULL,
    AuthorId              int NULL,
    Size                  bigint NOT NULL,
    ListDirName           nvarchar(256) NULL,
    ScopeId               uniqueidentifier NULL,
    ProgId                nvarchar(255) NULL
  )
```

SiteId: The site collection identifier of the site collection in which this item belongs.

WebId: The site identifier of the site in which this item belongs.

BinId: The Recycle Bin stage for this item.

DeleteUserId: The user identifier of the user who placed the specified item in the Recycle Bin.

DeleteTransactionId: The delete transaction identifier of the delete transaction to which this item belongs.

DeleteDate: The date on which this item was placed in the Recycle Bin.

ItemType: The type of the Recycle Bin item. ItemType MUST NOT be NULL and MUST be one of the values in Delete Item Type.

ListId: The list identifier of the list to which this item belongs.

DocId: The **document identifier** of the document that corresponds to this item. DocId MUST be NULL if ItemType is either 4 or 8.

DocVersionId: The document version of the document that corresponds to this item. DocVersionId MUST be NULL for the following values of ItemType: 1, 3, 4, 5, 6, or 7.

ListItemId: The identifier of the list item that corresponds to this item. ListItemId MUST be NULL for the following values of ItemType: 2, 4, or 6.

Title: The display name of the item.

DirName: The **directory name** of the document which corresponds to this item.

LeafName: The leaf name of the document which corresponds to this item.

AuthorId: The identifier of the user who originally created the item. If the value of ItemType is 7, AuthorId value MUST be NULL.

Size: The size of the item in Bytes.

ListDirName: The directory name of the list to which this item belongs. If the value of ItemType is 4, ListDirName value MUST be NULL.

ScopeId: The **scope identifier** of the security scope of the document that corresponds to this item.

ProgId: The file type which specifies the preferred application used to open the document that corresponds to this item.

2.2.9 XML Structures

None.

3 Protocol Details

This section provides detailed information about this protocol.

3.1 Back End Database Server Details

This section provides detailed information about the back-end database server.

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.

The back-end database server maintains the following sets of data for this protocol within both a **configuration database** and one or more content databases. Data within the appropriate databases are maintained until updated or removed.

Documents: A set of information about all documents in a content database. Document entries are identified by document identifiers, and are also are represented by store-relative form.

Lists: A set of information about all lists in a content database. List entries are identified by list identifiers, and are also represented by store-relative form.

List Items: A set of information about all list items in a content database. List item entries are identified by list item identifiers.

Quota Templates: Objects that represent the quota limits that can be set on a site collection. Quota templates are stored in the configuration database.

Roles: A set of information about all **roles** in a content database. Role entries are identified by role identifiers.

Site Collections: A set of information about all site collections in a content database. Site collection entries are identified by site collection identifiers and are also represented by either **absolute URLs** or store-relative form.

Permission levels: A set of information about all **permission levels** in a content database. Permission level entries are identified by permission level identifiers.

Sites: A set of information about all sites in a content database. Site entries are identified by site identifiers and are also represented by store-relative form.

Users: A set of information about all users in a content database. User entries are identified by user identifiers.

Versions: A set of information indicating the **current version** information for various components in the **farm**.

3.1.1.1 Audit Operations

The protocol server stores a hierarchy of objects. Operations that can be performed against those objects are divided into categories (for example updates, deletes, copies), and the protocol server maintains three bit masks, as defined in [MS-WSSFO] section 2.2.2.1, that specify which categories

of operations will be recorded ("audited"). An audit entry is recorded for an operation performed against the object if the corresponding category is set in any of the three sets of audit flags as specified in [MS-WSSFO] section 2.2.2.1. The three sets of audit flags are as follows:

1. **Direct Audit Flags:** These audit flags indicate operations that are audited when performed directly on an object.

2. **Inherited Audit Flags:** These audit flags indicate operations that are audited for an object due to the direct audit flags being set on an object contained within the object. In this manner, the inherited audit flags indicate that an audit entry is recorded for operations that indirectly affect an object even if the action is not performed directly on that object. For example, if the direct audit flags on a document in a folder specify that the "Delete" event will be audited on the document, then the inherited audit flags of the folder that contains the document will indicate that the "Delete" event will be audited for the folder.

3. **Global Audit Flags:** These audit flags indicate operations that are audited for all objects.

The protocol server stores the audit entries to an "Audit Log" table.

3.1.1.2 Quota Management Operations

Data is stored in hierarchical objects on the protocol server. A site collection is at the root of the logical partitioning of this hierarchy. Site collections can contain sites, which in turn can contain lists and items. A site collection can be considered as an independent unit and can be managed for security and size limits. Quota management allows for setting the maximum allowed disk space on the protocol server for a site collection and the maximum number of users allowed in a site collection if **Active Directory account creation mode** is used. The actual disk space used refers to the total size of all the content such as documents, lists, and list items in a site collection stored on the protocol server. Quota management also allows for setting warning limits for actual disk space used and the number of users in a site collection. Protocol clients can use the warning limits to warn the site collection administrators that their site crossed the warning limits but is still lower than the maximum allowed limits.

Protocol clients can set the maximum allowed and the warning limits on the actual disk space used and the number of users in a site collection using a call the stored procedure proc_SetSiteQuota. Protocol clients can request the quota information for a site collection using the stored procedure proc_GetSiteQuota.

Protocol clients can get a list of site collections that have crossed the warning limits set in the quota using the stored procedure proc_QMGetDiskWarning and can send out notifications to the site collection administrators of those site collections from the result set of the stored procedure. A call to the stored procedure proc_QMGetDiskWarning MUST be followed by a call to the stored procedure proc_QMMarkDiskWarning, which indicates that the site collections have been warned so that those site collections are not returned in the next call to proc_QMGetDiskWarning if there are no further changes in them.

The actual disk space used by a site collection is updated when stored procedures such as proc_AddDocument and proc_AddListItem, as specified in [MS-WSSFO], are used to add documents or list items to the site collection. The actual disk space used by a site collection on the protocol server can also be updated using the stored procedure proc_CalculateAndUpdateSiteDiskUsed. Stored procedures such as proc_AddDocument and proc_AddListItem also prevent adding of content to the protocol server if the site collection to which data is be added has crossed the allowed disk space limits set by a quota.

The number of users in a site collection is updated when the stored procedures such as proc_SecAddUser is used to add a user to a site collection. proc_SecAddUser also prevents adding

users to a site collection if the number of users exceeds its quota limit set for the maximum number of users allowed.

3.1.1.3 Recycle Bin Operations

3.1.1.3.1 Recycle Bin

The protocol server stores a collection of objects that are intended to be removed from a site collection in a logical container called a Recycle Bin, which is split into two parts. The first-stage Recycle Bin is visible to site collection administrators and Users, and the second-stage Recycle Bin is visible to site collection administrators.

A protocol client can move a Recycle Bin item from a site collection into the first-stage Recycle Bin or from the first-stage Recycle Bin to the second-stage Recycle Bin dependent on permission checks. A protocol client can also move an item from both of the Recycle Bins back to its original location in a site collection dependent on permission checks. A protocol client can also delete a Recycle Bin item from the second-stage Recycle Bin, which permanently deletes the data on the back-end database server physical disk.

The act of moving a Recycle Bin item into a Recycle Bin is called delete, and the inverse action is called restore. These actions are explained based on the diagrams in the following three subsections. In the diagrams, the first-stage Recycle Bin and the second-stage Recycle Bin are abbreviated as FSRB and SSRB, respectively.

3.1.1.3.2 Query Operations

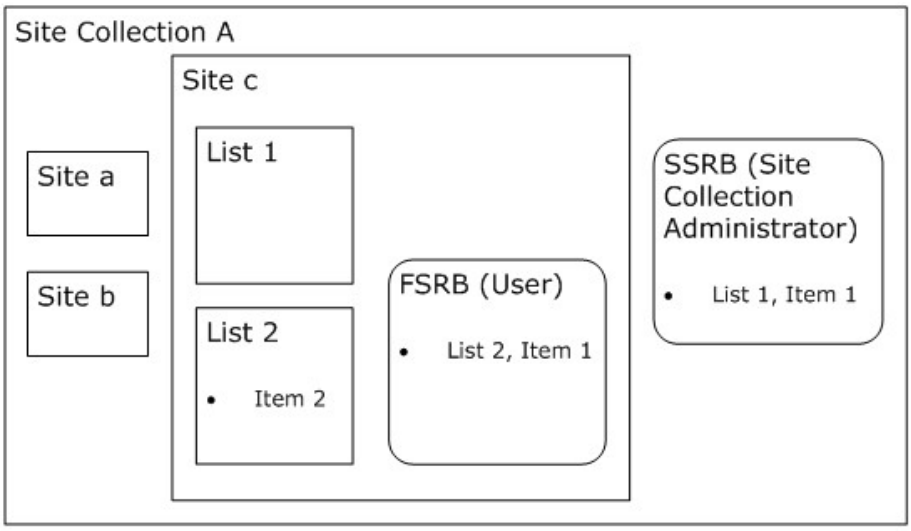

Figure 2: Query Operations on Recycle Bins

The protocol server can query the contents of the first-stage Recycle Bin if it has the appropriate permissions for Site c, which would return List 2, Item 1. The protocol server can query the contents of the second-stage Recycle Bin if it has site collection administrator permission on Site Collection A, which would return List 1, Item 1 and List 2, Item 1.

3.1.1.3.3 Delete Operations

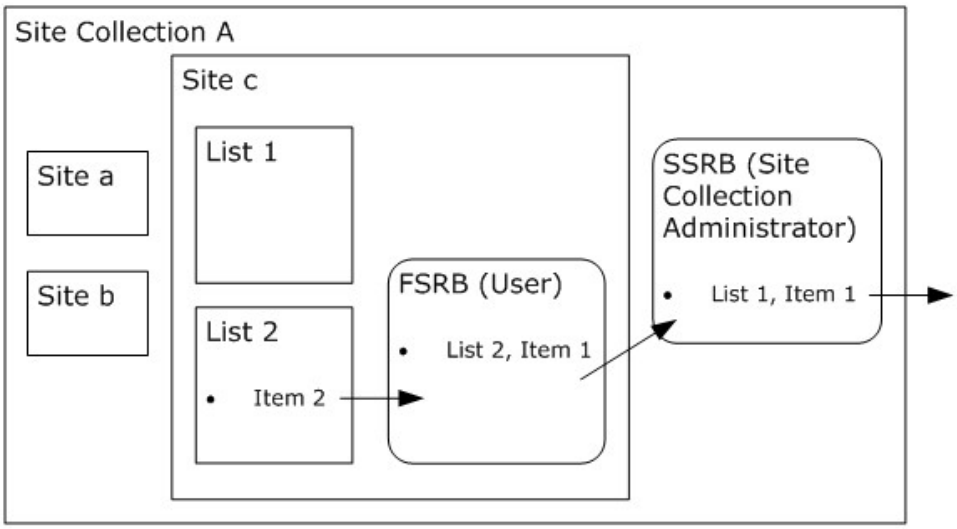

Figure 3: Delete Operations on Recycle Bins

The protocol server can delete List1, Item1 from the second-stage Recycle Bin and remove it from the back-end database server physical disk if it has site collection administrator permissions on Site Collection A. The protocol server can delete List 2, Item 1 from the first-stage Recycle Bin into the second-stage Recycle Bin if it has the appropriate permissions for Site c. The protocol server can delete List 2, Item 2 into the first-stage Recycle Bin if it has appropriate permission for Site c.

3.1.1.3.4 Restore Operations

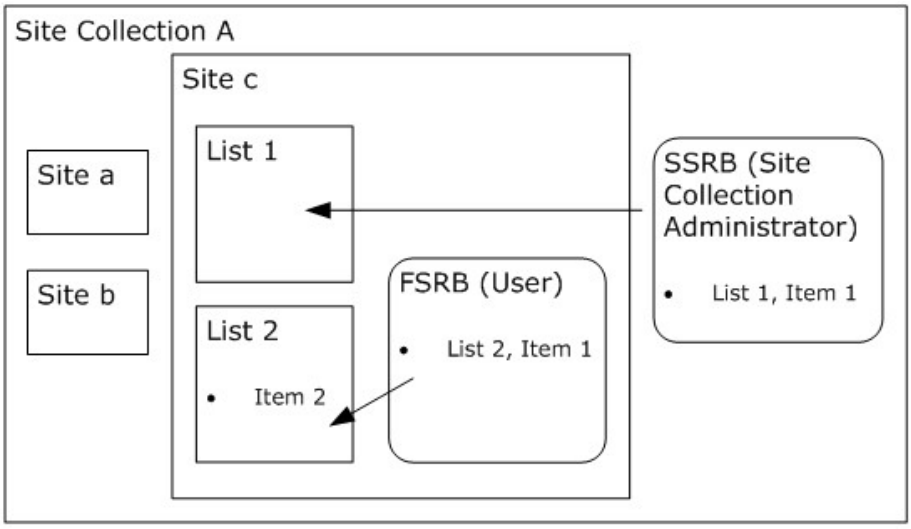

Figure 4: Restore Operations on Recycle Bins

The protocol server can restore List 2, Item1 from the first-stage Recycle Bin to its original location in List 2 if it has the appropriate permissions on Site c. The protocol server can restore List1, Item1

from the second-stage Recycle Bin to its original location in List 1 if it has the appropriate permissions on Site Collection A.

3.1.1.4 Security Operations

The protocol server stores a collection of sites with a **GUID** indicating whether there is an external security provider associated with the site, and the scope identifier associated with the site.

3.1.1.4.1 Operations related to External Security Provider

When the protocol server is called to remove external security provider for a site, the GUID associated with that site is removed.

Figure 5: Removal of External Security Provider

3.1.1.4.2 Operations related to ACL

The protocol server stores a collection of the binary serialization of ACL that is indexed by the scope identifier.

The protocol server stores a collection of list items that are indexed by site identifier, list identifier, and **item identifier**. The scope identifier for every list item is also stored.

When the protocol server is called to retrieve ACLs about a specific list item, a data structure containing the list items is joined with a data structure containing ACLs and returns a result set that contains ACL information for the list items.

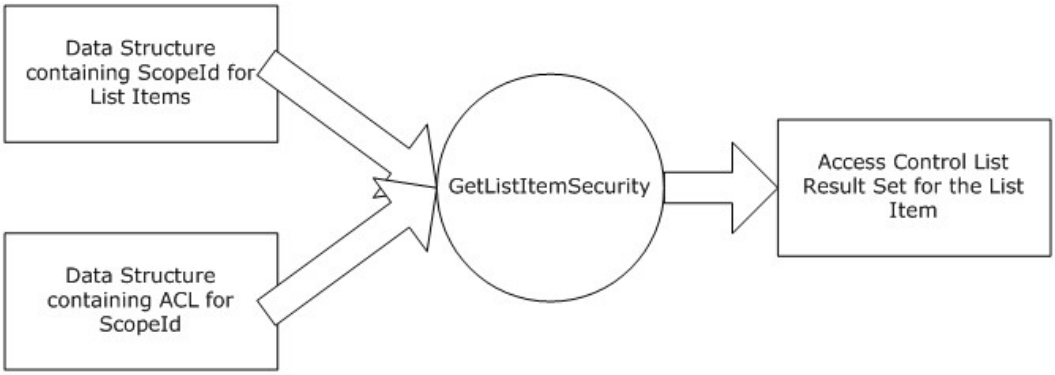

Figure 6: Retrieval of List Item Security

When the protocol server is called to retrieve ACL for all the unique security scopes of subsites contained in a specified site, a data structure that contains scope identifiers for all the subsites is

joined with a data structure that contains ACLs with the scope identifier and returns a result set that contains ACL information for all the unique security scopes of the subsites.

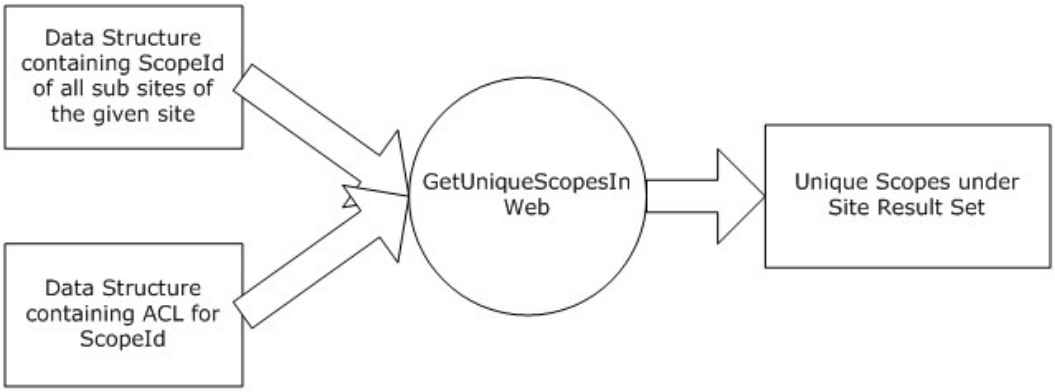

Figure 7: Retrieval of Unique Scopes for Subsites of a Specified Site

3.1.1.4.3 Operations related to User and Group

The protocol server stores a data structure that contains user identifier of **All Site Members**, and a data structure that contains information about all users within the site collection.

When the protocol server is called to return information about All Site Members, the two data structures are joined to return a result set that contains user information for All Site Members.

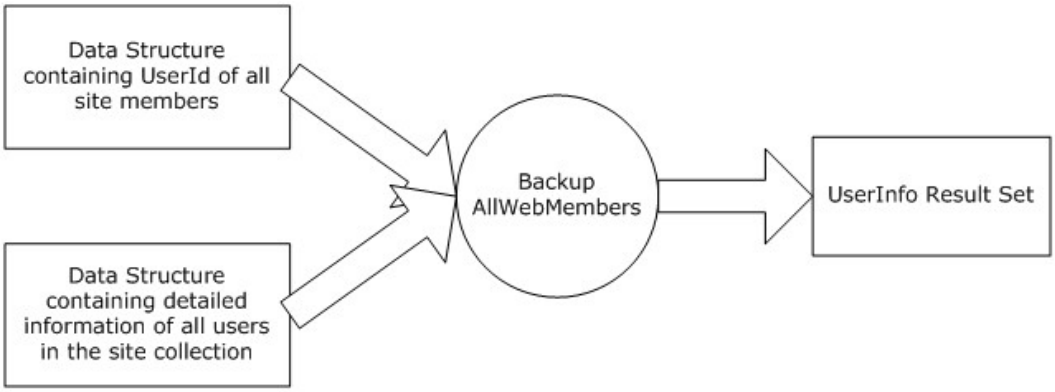

Figure 8: Retrieval of Information for All Site Members

3.1.2 Timers

An execution timeout timer on the protocol server governs the execution time for any requests. 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 Relationship to Other Protocols MUST be established before using this protocol as specified in [MS-TDS].

3.1.4 Message Processing Events and Sequencing Rules

3.1.4.1 fn_CompareTZTransitionDate

The fn_CompareTZTransitionDate function is called to compare a date and time value to a second date and time value described by its month, week, day of the week, and hour. The second date and time value has an implicit year equal to the first date and time value.

The T-SQL syntax for the function is as follows.

```
FUNCTION fn_CompareTZTransitionDate (
  @dtLocal      datetime,
  @_m           int,
  @_nwd         int,
  @_wd          int,
  @_h           int
)
RETURNS         bit;
```

@dtLocal: The date and time value to compare.

@_m: The month of the second date and time value.

@_nwd: The week of the month of the second date and time value.

@_wd: The weekday of the second date and time value.

@_h: The hour of the day of the second date and time value.

Return Value: The function MUST return 1 if @dtLocal is greater than or equal to the second date and time value. Otherwise, it MUST return 0.

3.1.4.2 fn_EscapeForLike

The fn_EscapeForLike function is called to prepare a string for use in a T-SQL LIKE search pattern. The characters '%', '_', and '[' are used as wildcard characters in a LIKE search pattern. To use them as literal characters rather than wildcard characters, each of these wildcard characters MUST be surrounded by square brackets('[' and ']'). This process is known as escaping the string. The T-SQL syntax for the function is as follows.

```
FUNCTION dbo.fn_EscapeForLike(
  @Source              nvarchar(260),
  @AddTerminalWildcard bit = 1
)
RETURNS nvarchar(1024);
```

@Source: The string to be escaped.

@AddTerminalWildcard: A bit flag specifying whether a wildcard search pattern should be added to the escaped @Source string.

Return Value: If @Source is NULL, then fn_EscapeForLike MUST return NULL. Otherwise, fn_EscapeForLike MUST escape the @Source string by replacing wildcard characters according to the following table.

Character in Source	Escaped Character
%	[%]
_	[_]
[	[[]

If @AddTerminalWildcard is set to 1 and @Source is not NULL, the non-escaped pattern '/%' MUST be appended to the escaped @Source string.

3.1.4.3 fn_GetRootFolder

The fn_GetRootFolder function is called to retrieve the **full URL** for the **root folder** of a list.

The T-SQL syntax for the function is as follows.

```
FUNCTION fn_GetRootFolder(
  @tp_RootFolder    uniqueidentifier
)
RETURNS             nvarchar(260);
```

@tp_RootFolder: The identifier of a root folder.

Return Value: If @tp_RootFolder does not reference a valid root folder, then fn_GetRootFolder MUST return NULL. Otherwise, fn_GetRootFolder MUST return the full URL of the specified root folder.

3.1.4.4 fn_HtmlEncode

The fn_HtmlEncode function is called to replace ampersand, less-than, greater-than, single-quote, double-quote, and line-feed characters in input strings with the appropriate entity references and return it. The T-SQL syntax for the function is as follows.

```
FUNCTION fn_HtmlEncode (
  @Value              nvarchar(1023),
  @PreserveNewLine    bit
)
RETURNS               nvarchar(4000);
```

@Value: The string to be encoded.

@PreserveNewLine: A bit value that specifies whether to append <br> after the line-feed character. If the value is 1,
 MUST be appended to every line-feed character.

Return Value: The string that is converted according to the following table.

Character in input string	HTML character-entity in output string
& (ampersand)	&amp;
< (less-than)	&lt;
> (greater-than)	&gt;

Character in input string	HTML character-entity in output string
' (single-quote)	&#39;
" (double-quote)	&quot;
"\n"(ASCII line feed)	"\n" (if @PreserveNewLine is 1)

If the converted string has more than 4000 characters, only the first 4000 characters MUST be returned.

3.1.4.5 fn_LocalDayFromUTCDate

The fn_LocalDayFromUTCDate function is called to calculate the day in local time from the specified Coordinated Universal Time (UTC) date and time value and the specified time zone information. The T-SQL syntax for the function is as follows.

```
FUNCTION fn_LocalDayFromUTCDate (
  @dtUTC            datetime,
  @BiasMinutes      int,
  @Dlt_m            int,
  @Dlt_nwd          int,
  @Dlt_wd           int,
  @Dlt_h            int,
  @Std_m            int,
  @Std_nwd          int,
  @Std_wd           int,
  @Std_h            int
)
RETURNS             datetime;
```

@dtUTC: The date and time value, expressed in UTC, for which the local day is desired.

@BiasMinutes: The difference, in minutes, between UTC and local time for the time zone.

@Dlt_m: The month in which Daylight Saving Time begins for the time zone.

@Dlt_nwd: The week of the month in which Daylight Saving Time begins for the time zone.

@Dlt_wd: The weekday on which Daylight Saving Time begins for the time zone.

@Dlt_h: The hour of the day when Daylight Saving Time begins for the time zone.

@Std_m: The month in which Daylight Saving Time ends for the time zone.

@Std_nwd: The week of the month in which Daylight Saving Time ends for the time zone.

@Std_wd: The weekday on which Daylight Saving Time ends for the time zone.

@Std_h: The hour of the day when Daylight Saving Time ends for the time zone.

Return Value: The function MUST return a date and time value expressed in local time that occurs in the same day as the specified UTC date and time value as expressed in local time.

3.1.4.6 proc_AddAuditEntry

The proc_AddAuditEntry stored procedure is called to add an audit entry about an object identified by its identifier to the audit log.

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

```
PROCEDURE proc_AddAuditEntry(
  @SiteId          uniqueidentifier,
  @ItemId          uniqueidentifier,
  @ItemType        smallint,
  @UserId          int,
  @MachineName     nvarchar(128) = NULL,
  @MachineIp       nvarchar(20) = NULL,
  @Location        nvarchar(260)= NULL,
  @LocationType    tinyint= NULL,
  @Occurred        datetime,
  @Event           int,
  @EventName       nvarchar(128)= NULL,
  @EventSource     tinyint,
  @SourceName      nvarchar(256)= NULL,
  @EventData       ntext= NULL
);
```

@SiteId: The site collection identifier which contains the object for which the audit entry is being added.

@ItemId: The identifier of the object.

@ItemType: The type of object, which MUST be a value from the Audit Item Type enumeration as specified in [MS-WSSFO] section 2.2.3.2.

@UserId: The user identifier of the User who performed the operation that generated the audit entry.

@MachineName: Reserved. MUST be NULL.

@MachineIp: Reserved. MUST be NULL.

@Location: The store-relative form of the object<2>.

@LocationType: Reserved. MUST be 0x00.

@Occurred: The date and time, in UTC, when the operation occurred.

@Event: The type of Audit Event, which MUST be a value from the Audit Event Type enumeration.

@EventName: If **@**Event is 0x00000064 (Custom), then this is the name of the event, which MUST be a non-empty string. Otherwise, this MUST be NULL.

@EventSource: The source of the Audit Event, which MUST be a value from the Audit Event Source enumeration.

@SourceName: The display name of the Audit Event source.

@EventData: The event data of the Audit Event. This MUST be less than or equal to 4000 characters in length unless **@**Event is 0x00000006 (Content Type updated).

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.7 proc_AddAuditEntryUrl

The proc_AddAuditEntryUrl stored procedure is called to add an audit entry about an object identified by its **URL** (as specified by its directory name and leaf name) to the audit log.

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

```
PROCEDURE proc_AddAuditEntryUrl(
  @SiteId         uniqueidentifier,
  @DirName        nvarchar(256),
  @LeafName       nvarchar(128),
  @ItemType       smallint,
  @UserId         int,
  @MachineName    nvarchar(128)= NULL,
  @MachineIp      nvarchar(20)= NULL,
  @Location       nvarchar(260)= NULL,
  @LocationType   tinyint= NULL,
  @Occurred       datetime,
  @Event          int,
  @EventName      nvarchar(128)= NULL,
  @EventSource    tinyint,
  @SourceName     nvarchar(256)= NULL,
  @EventData      ntext= NULL
  @EventFlag      int= 0
);
```

@SiteId: The site collection identifier which contains the object for which the audit entry is being added.

@DirName: The directory name of the object.

@LeafName: The leaf name of the object.

@ItemType: The type of object, which MUST be a value from the Audit Item Type enumeration as specified in [MS-WSSFO] section 2.2.3.2.

@UserId: The user identifier of the User who performed the operation that generated the audit entry.

@MachineName: Reserved. MUST be NULL.

@MachineIp: Reserved. MUST be NULL.

@Location: The URL of the object. SHOULD be in Store-relative Form but MAY be the leaf name of the object<3>.

@LocationType: Reserved. MUST be 0x00.

@Occurred: The date and time, in UTC, when the operation occurred.

@Event: The type of Audit Event, which MUST be a value from the Audit Event Type enumeration.

@EventName: If **@**Event is 0x00000064 (Custom), then this is the name of the event, which MUST be a non-empty string. Otherwise, this MUST be NULL.

@EventSource: The source of the Audit Event, which MUST be a value from the Audit Event Source enumeration.

@SourceName: The display name of the Audit Event source.

@EventData: The event data of the Audit Event. This MUST be less than or equal to 4000 characters in length unless **@**Event is 0x00000006 (Content Type updated).

@EventFlag: The audit flags, as defined in [MS-WSSFO] section 2.2.2.1, for which the **@**Event applies, which MUST be an audit flags bit mask. The protocol server determines the direct, inherited, and global audit flags applicable to the object and MUST add the audit entry only if **@**EventFlag is zero or (**@**EventFlag & (**{**AuditFlags**}** | **{**InheritAuditFlags**}** | **{**GlobalAuditMask**}**)) is nonzero.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.8 proc_CalculateAndUpdateSiteDiskUsed

The proc_CalculateAndUpdateSiteDiskUsed stored procedure is called to calculate and update the size of disk, in Bytes, used by a site collection.

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

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

@SiteId: The site collection identifier of the site collection of which disk size is calculated and updated.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.9 proc_ConfirmSiteUsage

The proc_ConfirmSiteUsage stored procedure is called to confirm that the specified site collection is in use and to specify whether it can be deleted automatically when not in use.

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

```
PROCEDURE proc_ConfirmSiteUsage(
  @SiteId     uniqueidentifier NOT NULL,
  @Disable    bit
);
```

@SiteId: The site collection identifier of the site collection which is being confirmed.

@Disable: A bit value that specifies if the site collection can be deleted automatically when not in use. @Disable MUST be one of the following values:

Value	Description
0	Delete site collection automatically when not in use.
1	Do not delete site collection automatically when not in use.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.10 proc_ConvertStringToDate

The proc_ConvertStringToDate stored procedure is called to transform text stored in a **field** into a **datetime** string. The conversion logic only preserves the date part and sets the time part to 1:00 AM based on the **regional settings** of the site which contains the specified list.

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

```
PROCEDURE proc_ConvertStringToDate (
    @ListId         uniqueidentifier,
    @OldColName     nvarchar(255),
    @OldRowOrdinal  int,
    @NewColName     nvarchar(255),
    @NewRowOrdinal  int,
    @WebTimeBias    int,
    @DateOrder      nvarchar(3),
    @bFromNtext     bit
);
```

@ListId: The list identifier of the list where this data transformation takes place.

@OldColName: The SQL column name in which the text data is stored.

@OldRowOrdinal: A **zero-based index** number that specifies which row the text Field data is stored in the UserData View (as defined in [MS-WSSFO] section 2.2.7.8) if each list item of this list takes up more than one row in the UserData View.

@NewColName: The SQL column name in which the datetime string is going to be stored.

@NewRowOrdinal: A zero-based index number that specifies which row the datetime string field data will be stored in the UserData View (as defined in [MS-WSSFO] section 2.2.7.8) if each list item of this list takes up more than one row in the UserData View.

@WebTimeBias: An integer number that specifies the difference in minutes between the current time zone and UTC. This number MUST take daylight savings time into consideration where applicable. The stored procedure then uses this number to convert the text field into UTC times. It assumes the text string contains a local time.

@DateOrder: A string of three characters. It will be passed to T-SQL statement "SET DATEFORMAT". This setting is used by the back-end database server in the interpretation of character strings as they are converted to date values. It has no effect on the display of date values. @DateOrder MUST be one of the following values.

Value	Description
‘mdy’	The date string will be interpreted as "Month, Day, Year"
‘dmy’	The date string will be interpreted as "Day, Month, Year"
‘ymd’	The date string will be interpreted as "Year, Month, Day"

@bFromNText: A bit that specifies how the stored procedure considered the database column specified by @OldColName. @bFromNText MUST be either 0 or 1. If @bFromNText is 1, the stored procedure will assume the specified database column as an ntext type column. Otherwise, it will assume the database column as a nvarchar(255) column.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.11 proc_DeleteRecycleBinItem

The proc_DeleteRecycleBinItem stored procedure is called to permanently delete a Recycle Bin item.

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

```
PROCEDURE proc_DeleteRecycleBinItem(
  @SiteId                uniqueidentifier,
  @WebId                 uniqueidentifier,
  @UserId                int,
  @DeleteTransactionId   varbinary(16)
);
```

@SiteId: The site collection identifier of the site collection which contains the Recycle Bin item. This parameter MUST NOT be NULL.

@WebId: The site identifier of the site which contains the Recycle Bin item. If the @UserId parameter is nonzero, then this parameter MUST NOT be NULL.

@UserId: A numeric value that MUST either be 0 or the user identifier of the User who deleted the Recycle Bin item. This parameter MUST NOT be NULL.

@DeleteTransactionId: The delete transaction identifier of the delete transaction. DeleteTransactionId is a varbinary(16) type, but it MUST contain a valid GUID and MUST NOT be NULL.

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

Value	Description
0	Successful completion.
1168	Either the Recycle Bin item could not be found or the site which contains the Recycle Bin item could not be found.
1359	An internal error occurred.

Result Sets: MUST NOT return any result sets.

3.1.4.12 proc_DetectOrphans

The proc_DetectOrphans stored procedure is called to detect orphaned objects in the content database for reporting purposes.

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

```
PROCEDURE proc_DetectOrphans();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return five result sets in the following order:

3.1.4.12.1 Site Collection with no Sites Result Set

The Site Collection with no Sites result set returns site collections without sites.

The T-SQL syntax for the result set is defined in the Site Collection with no Sites Result Set.

3.1.4.12.2 Sites with no Site Collection Result Set

The Sites with no Site Collection result set returns sites with no site collection and no associated folders.

The T-SQL syntax for the result set is defined in the Sites with no Site Collection Result Set.

3.1.4.12.3 Sites with no Parent Site Result Set

The Sites with no Parent Site result set returns those sites that have no parent site.

The T-SQL syntax for the result set is defined in the Sites with no Parent Site Result Set.

3.1.4.12.4 Folders with no Site Result Set

The Folders with no Site result set returns folders at the root of the site that have no associated site.

The T-SQL syntax for the result set is defined in the Folders with no Site Result Set.

3.1.4.12.5 Orphaned Lists Result Set

The Orphaned Lists result set returns orphaned Lists in the following three categories:

- Lists with no parent site
- Lists with documents with no parent list
- Lists with items with no parent list

The Orphaned Lists result set MUST return a result set having a number of rows equal to the number of orphaned lists. If there are no such lists, the result set returns zero rows.

The T-SQL syntax for the result set is defined in the Orphaned Lists Result Set.

3.1.4.13 proc_DetectOrphansFix

The proc_DetectOrphansFix stored procedure is called to detect orphaned objects in the content database upon a database repair attempt.

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

```
PROCEDURE proc_DetectOrphansFix();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return five result sets in the following order:

3.1.4.13.1 Site Collection with no Sites Result Set

The Site Collection with no Sites result set returns site collections without sites.

The T-SQL syntax for the result set is defined in the Site Collection with no Sites Result Set.

3.1.4.13.2 Sites with no Site Collection Result Set

The Sites with no Site Collection result set returns sites with no site collection and no associated folder.

The T-SQL syntax for the result set is defined in the Sites with no Site Collection Result Set.

3.1.4.13.3 Sites with No Parent Site Result Set

The Sites with No Parent Site result set returns sites with no parent site.

The T-SQL syntax for the result set is defined in the Sites with no Parent Site Result Set.

3.1.4.13.4 Folders with no Site Result Set

The Folders with no Site result set returns folders at the root of the site that have no associated site.

The T-SQL syntax for the result set is defined in the Folders with no Site Result Set.

3.1.4.13.5 Orphaned Lists Result Set

Orphaned Lists result set returns lists in the following three categories:

- Lists with no parent site
- Lists with documents with no parent list
- Lists with items with no parent list

The Orphaned Lists result set MUST return a result set having a number of rows equal to the number of orphaned lists. If there are no such lists, the result set returns 0 rows.

The T-SQL syntax for the result set is defined in the Orphaned Lists Result Set.

3.1.4.14 proc_DTSetRelationship

The proc_DTSetRelationship stored procedure is called to set the parent/child relationship between two documents. The documents are an original document (the parent) and a document created by transforming the original document (the child).

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

```
PROCEDURE proc_DTSetRelationship (
  @SiteId            uniqueidentifier,
  @DirName           nvarchar(256),
  @ChildDocLevel     tinyint,
  @ChildLeafName     nvarchar(128),
  @ParentDocLevel    tinyint,
  @ParentLeafName    nvarchar(128),
  @TransformerId     uniqueidentifier,
  @ParentVersion     int,
);
```

@SiteId: The site collection identifier of the site collection which contains the parent and child documents.

@DirName: The directory name of the documents.

@ChildDocLevel: The **publishing level** for the child document.

@ChildLeafName: The leaf name for the child document.

@ParentDocLevel: The publishing level for the parent document.

@ParentLeafName: The leaf name for the parent document. This value MUST NOT be NULL.

@TransformerId: The identifier of the agent that performed the transformation. If the document is a transformed version of another document, this MUST be the GUID of the agent that performed the transformation. Otherwise, @TransformerId MUST be NULL.

@ParentVersion: The document version of the parent document.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.15 proc_EnumRecycleBinItemsForCleanup

The proc_EnumRecycleBinItemsForCleanup stored procedure is called to get the Recycle Bin items that have been in the Recycle Bin longer than the specified number of days. Recycle Bin items whose site has been deleted will not be included in the result set.

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

```
PROCEDURE proc_EnumRecycleBinItemsForCleanup (
  @Days      int
);
```

@Days: An integer that specifies the number of days a Recycle Bin item needs to be in the Recycle Bin to be included in the result set. If this value is NULL, an empty result set will be returned. If this value is 0 or a negative number, then all of the Recycle Bin items will be returned.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.15.1 Recycle Bin Items For Cleanup Result Set

The Recycle Bin Items For Cleanup result set returns a list of Recycle Bin items. The proc_EnumRecycleBinItemsForCleanup stored procedure will always return a Recycle Bin Items For Cleanup result set, and there is no limit on how many rows it can contain. The T-SQL syntax for the result set is as follows.

```
SiteId                  uniqueidentifier,
WebId                   uniqueidentifier,
{DeleteTransactionId}   uniqueidentifier;
```

SiteId: The site collection identifier of the site collection which contains the Recycle Bin item. SiteId MUST NOT be NULL.

WebId: The site identifier of the site which contains the Recycle Bin item. WebId MUST NOT be NULL.

{DeleteTransactionId}: The delete transaction identifier of the delete transaction. {DeleteTransactionId} MUST NOT be NULL.

3.1.4.16 proc_EnumRecycleBinToFreeSecondStageQuota

The proc_EnumRecycleBinToFreeSecondStageQuota stored procedure is called to enumerate the size and delete transaction identifier information of each deleted Recycle Bin item in the second-stage Recycle Bin.

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

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

@SiteId: The site collection identifier of the site collection which contains the second-stage Recycle Bin.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.16.1 Item Metadata Result Set

The Item Metadata result set returns metadata information about deleted items in the second-stage Recycle Bin of a site collection. It contains the size, in bytes, and the delete transaction identifier of the deleted Recycle Bin items. The T-SQL syntax for the result set is as follows.

```
{DeleteTransactionId}   uniqueidentifier,
```

```
Size                    bigint;
```

{DeleteTransactionId}: The delete transaction identifier of the delete transaction on this Recycle Bin item.

Size: The size, in bytes, of the Recycle Bin item.

3.1.4.17 proc_EnumSitesForDeadWebCheck

The proc_EnumSitesForDeadWebCheck stored procedure is called to return information about top-level sites such as the number of days that have elapsed between the last site certification and the current date, the **e-mail address** of the site collection administrators, and the **notify count** of the top-level site.

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

```
PROCEDURE proc_EnumSitesForDeadWebCheck
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.17.1 Site Information Result Set

The Site Information result set returns information about top-level sites such as the number of days that have elapsed between the last site certification and the current date, the e-mail address of the site collection administrators, and the notify count of the top-level site. The Site Information result set MUST return one row for every combination of top-level site and site collection administrators for that site or no rows if there are no top-level sites. The T-SQL syntax for the result set is as follows.

```
Id                      uniqueidentifier,
{DiffTodayCertDate}     int,
DeadWebNotifyCount      smallint,
FullUrl                 nvarchar(256),
Title                   nvarchar(255),
Language                int,
tp_E-mail               nvarchar(255);
```

Id: The site identifier of the top-level site.

{DiffTodayCertDate}: The number of days between the certification date of the top-level site to the current date.

DeadWebNotifyCount: The notify count value of the top-level site.

FullUrl: The store-relative form of the top-level site.

Title: The title of the top-level site.

Language: The **language code identifier (LCID)** which indicates the language of a top-level site.

tp_E-mail: The e-mail address, if it exists, for the site collection administrators associated with the top-level site.

3.1.4.18 proc_ForceDeleteList

The proc_ForceDeleteList stored procedure is called to delete a list in a corrupted state. It is intended as a last resort to remove a list. The list will be permanently deleted without being moved to Recycle Bin.

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

```
PROCEDURE proc_ForceDeleteList(
  @SiteId      uniqueidentifier,
  @DirName     nvarchar(256),
  @LeafName    nvarchar(128),
  @UserId      int
);
```

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

@DirName: The store-relative form of the folder which contains the **page** specified by @LeafName.

@LeafName: The file name of the Page from which to delete the list.

@UserId: The user identifier of the **current user**.

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

Value	Description
0	Successful completion.
2	The list does not exist.
87	The specified @SiteId, @DirName and @LeafName do not point to a root folder of a list.
8398	A list marked as undeletable cannot be deleted.

Result Sets: MUST NOT return any result sets.

3.1.4.19 proc_GetAdminRecycleBinInfo

The proc_GetAdminRecycleBinInfo stored procedure is called to get information about the first-stage Recycle Bin, including the number of Recycle Bin items it contains and the overall size of those Recycle Bin items.

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

```
PROCEDURE proc_GetAdminRecycleBinInfo(
  @SiteId       uniqueidentifier,
  @ItemCount    intOUTPUT,
  @Size         bigintOUTPUT
);
```

@SiteId: The site collection identifier of the site collection which contains the Recycle Bin. This parameter MUST NOT be NULL.

@ItemCount: The total number of Recycle Bin items in the first-stage Recycle Bin of the site collection specified by the @SiteId parameter. This parameter MUST NOT be NULL and MUST be a non-negative numeric value.

@Size: The total size, in Bytes, of the Recycle Bin items in the first-stage Recycle Bin of the site collection specified by the @SiteId parameter. This parameter MUST NOT be NULL and MUST be a non-negative numeric value.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.20 proc_GetAdminRecycleBinItems

The proc_GetAdminRecycleBinItems stored procedure is called to get the full list of Recycle Bin items for a particular site collection.

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

```
PROCEDUREproc_GetAdminRecycleBinItems(
  @SiteId    uniqueidentifier
);
```

@SiteId: The site collection identifier of the site collection which contains the Recycle Bin whose Recycle Bin items MUST be returned. This parameter MUST NOT be NULL.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.20.1 Admin Recycle Bin Items Result Set

The Admin Recycle Bin Items result set returns the details of a set of Recycle Bin items. There is no limit on the number of rows that the Admin Recycle Bin Items result set can contain. The T-SQL syntax for the result set is as follows.

```
Title                   nvarchar(260),
DirName                 nvarchar(256),
LeafName                nvarchar(128),
AuthorId                int,
AuthorName              nvarchar(255),
AuthorE-mail            nvarchar(255),
DeleteDate              datetime,
Size                    bigint,
DeleteUserId            int,
DeletedByName           nvarchar(255),
DeletedByE-mail         nvarchar(255),
DeleteTransactionId     varbinary(16),
ItemType                tinyint,
BinId                   tinyint,
{tp_ImageUrl}           nvarchar(255),
ProgId                  nvarchar(255),
WebId                   uniqueidentifier;
```

Title: The display name of the Recycle Bin item. Title MUST NOT be NULL.

DirName: The directory name of the Recycle Bin item. DirName MUST NOT be NULL.

LeafName: The file name of the Recycle Bin item. LeafName MUST NOT be NULL.

AuthorId: The identifier of the User who is the **author** of the Recycle Bin item.

AuthorName: The name of the User who is the author of the Recycle Bin item.

AuthorE-mail: The e-mail address of the User who is the author of the Recycle Bin item.

DeleteDate: The date when the Recycle Bin item was deleted. DeleteDate MUST NOT be NULL.

Size: The size, in Bytes, of the Recycle Bin item. Size MUST NOT be NULL.

DeleteUserId: The identifier of the User who deleted the Recycle Bin item.

DeletedByName: The display name of the User who deleted the Recycle Bin item.

DeletedByE-mail: The e-mail address of the User who deleted the Recycle Bin item.

DeleteTransactionId: The delete transaction identifier of the delete transaction. DeleteTransactionId is a varbinary(16) value, but it MUST contain a valid GUID and MUST NOT be NULL.

ItemType: A numerical value indicating the type of the Recycle Bin item. ItemType MUST NOT be NULL and MUST be one of the values in Delete Item Type.

BinId: A numerical value that indicates which Recycle Bin Stage the Recycle Bin item is in. BinId MUST NOT be NULL and MUST be one of the following values.

Value	Description
1	The Recycle Bin item is in the first-stage Recycle Bin.
2	The Recycle Bin item is in the second-stage Recycle Bin.

{tp_ImageUrl}: The URL for the icon of the list if the Recycle Bin item is a list or list item.

ProgId: The file type if the Recycle Bin item is a **file**.

WebId: The site identifier of the site which contains the Recycle Bin item. WebId MUST NOT be NULL.

3.1.4.21 proc_GetCustomizedDocumentsInWeb

The proc_GetCustomizedDocumentsInWeb stored procedure is called to get customized documents in a site.

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

```
PROCEDURE proc_GetCustomizedDocumentsInWeb (
    @SiteId         uniqueidentifier,
    @WebFullUrl     nvarchar(260),
    @DocIdLast      uniqueidentifier
);
```

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

@WebFullUrl: The URL of the site which contains the requested documents.

@DocIdLast: The lowest document identifier considered for the selection.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.21.1 DocumentID Result Set

The DocumentID result set returns the document identifiers of the customized documents in a site. The DocumentID result set can return fewer rows than the total number of documents found. The T-SQL syntax for the result set is as follows.

```
Id    uniqueidentifier
```

Id: The document identifier of the customized document.

3.1.4.22 proc_GetDeadWebInfo

The proc_GetDeadWebInfo stored procedure is called to get the last certification date and the notify count of the specified site collection.

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

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

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.22.1 Dead Web Information Result Set

The Dead Web Information result set returns information about the specified site collection**.** The Dead Web Information result set MUST return one row for the specified site collection or no rows if there is no such site collection. The T-SQL syntax for the result set is as follows.

```
CertificationDate     datetime
DeadWebNotifyCount    smallint
```

CertificationDate: The last certification date of the site collection.

DeadWebNotifyCount: The value of notify count of the site collection.

3.1.4.23 proc_GetDocLibrarySizes

The proc_GetDocLibrarySizes stored procedure is called to get the size, in bytes, and metadata information of the document libraries of a site collection.

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

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

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.23.1 Document Library Size Result Set

proc_GetDocLibrarySizes returns the size, in Bytes, and metadata information of the document libraries of a site collection. The Document Library Size result set MUST contain no rows if there is no such site collection or there are no document libraries in the site collection. Otherwise, it MUST contain one row for each document library in the site collection. The T-SQL syntax for the result set is as follows.

```
TotalSize            bigint,
tp_ID                uniqueidentifier,
tp_WebID             uniqueidentifier,
tp_Modified          datetime,
tp_ServerTemplate    int,
DirName              nvarchar(256),
LeafName             nvarchar(128),
tp_ImageUrl          nvarchar(255),
tp_Title             nvarchar(255),
tp_ItemCount         int,
WebTemplate          int,
Language             int,
FullUrl              nvarchar(256);
```

TotalSize: The sum of the document sizes, document version sizes, personalization sizes and Web Part sizes in Bytes.

tp_ID: The list identifier of the document library.

tp_WebID: The site identifier of the site to which the document library belongs.

tp_Modified: The date and time the document library was last modified.

tp_ServerTemplate: The **list template** used to create the list.

DirName: The directory name of the document library.

LeafName: The leaf name of the document library.

tp_ImageUrl: The store-relative form of the image associated with the document library.

tp_Title: The name of the document library.

tp_ItemCount: The number of Items in this document library.

WebTemplate: The **site template** of this site.

Language: The LCID that indicates the language of a site.

FullUrl: The store-relative form of the site.

3.1.4.24 proc_GetDocSizeInfo

The proc_GetDocSizeInfo stored procedure is called to get the size, in Bytes and the metadata information (described in detail in the following Document Size result set section) of each document of a site collection.

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

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

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.24.1 Document Size Result Set

proc_GetDocSizeInfo returns the metadata and different kinds of size information for each document. The Document Size result set MUST contain no rows if there is no such site collection or there are no documents in the site collection. Otherwise, it MUST contain one row for each document in the site collection. The T-SQL syntax for the result set is as follows.

```
TotalSize                          bigint,
TimeLastModified                   datetime,
Id                                 uniqueidentifier,
DirName                            nvarchar(256),
LeafName                           nvarchar(128),
Size                               int,
SetupPath                          nvarchar(255),
{PersonalizationandWebPartSize}    bigint,
{DocumentVersionSize}              bigint,
DoclibRowId                        int,
WebId                              uniqueidentifier,
{DocListId}                        uniqueidentifier,
FullUrl                            nvarchar(256),
Language                           int,
{WebPartTypeSize}                  bigint;
```

TotalSize: The total size, in Bytes, of the document, personalization, Web Part and document version of a document.

TimeLastModified: The date and time the document was last modified.

Id: The document identifier of the document.

DirName: The directory name of the list that is not a document library**.**

LeafName: The leaf name of the list that is not document library.

Size: The document size**.**

SetupPath: The path from which the document was originally installed.

{**PersonalizationandWebPartSize**}**:** The total size, in Bytes, of the personalization and Web Parts of a document.

{DocumentVersionSize}: The size, in Bytes, of the document versions of a document

DoclibRowId: The **row** identifier of the document in a list.

WebId: The site identifier of the site.

{DocListId}: The list identifier of the list which contains the document

FullUrl: The store-relative form of the site.

Language: The LCID which indicates the language of a site.

{WebPartTypeSize}: The Web Part type size.

3.1.4.25 proc_GetFirstUniqueAncestorWebUrl

The proc_GetFirstUniqueAncestorWebUrl stored procedure is called to get the URL in store-relative form of the **first unique ancestor** of the specified site.

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

```
PROCEDURE proc_GetFirstUniqueAncestorWebUrl(
  @WebId    uniqueidentifier
);
```

@WebId: The site identifier of the requested site.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.25.1 First Ancestor Site Url Result Set

proc_GetFirstUniqueAncestorWebUrl returns the URL in store-relative form of the first unique ancestor of the specified site. This result set MUST return no rows if the requested site cannot be found. Otherwise, it MUST return one row. The T-SQL syntax for the result set is as follows.

```
FullUrl    nvarchar(256);
```

FullUrl: The store-relative form of the first unique ancestor of the specified site.

3.1.4.26 proc_GetListSizes

The proc_GetListSizes stored procedure is called to get the size, in Bytes, and metadata information of the lists that are not document libraries of a site collection.

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

```
PROCEDURE proc_GetListSizes (
  @SiteId    uniqueidentifier
```

```
);
```

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.26.1 Lists Size Result Set

The Lists Size result set returns the size, in Bytes, and metadata information about the lists that are not document libraries in a site collection. The Lists Size result set MUST contain no rows if there is no such site collection or all lists are document libraries in the site collection. Otherwise, it MUST contain one row for each list that is not a document library of the site collection. The T-SQL syntax for the result set is as follows.

```
TotalSize            bigint,
tp_ID                uniqueidentifier,
tp_WebID             uniqueidentifier,
tp_Modified          datetime,
tp_ServerTemplate    int,
DirName              nvarchar(256),
LeafName             nvarchar(128),
tp_ImageUrl          nvarchar(255),
tp_Title             nvarchar(255),
tp_ItemCount         int,
WebTemplate          int,
Language             int,
FullUrl              nvarchar(256);
```

TotalSize: The total size of the document and User data of a list that is not a document library.

tp_ID: The list identifier of the list that is not a document library.

tp_WebID: The site identifier of the site to which the list belongs.

tp_Modified: The date and time the list was last modified.

tp_ServerTemplate: The **list template identifier** used to create the list.

DirName: The directory name of the list**.**

LeafName: The leaf name of the list.

tp_ImageUrl: The store-relative form of the image associated with the list.

tp_Title: The name of the list.

tp_ItemCount: The number of Items in the list.

WebTemplate: The value that indicates the site Template used for the site.

Language: The LCID which indicates the language of a site.

FullUrl: The store-relative form of the site.

3.1.4.27 proc_GetRecycleBinItemInfo

The proc_GetRecycleBinItemInfo stored procedure is called to get information about a particular Recycle Bin item.

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

```
PROCEDURE proc_GetRecycleBinItemInfo (
  @SiteId                 uniqueidentifier,
  @DeleteTransactionId    varbinary(16),
  @ItemType               int OUTPUT,
  @ContainingListDeleted  bit OUTPUT,
  @ContainingListName     nvarchar(255) OUTPUT,
  @DirName                nvarchar(256) OUTPUT,
  @WebUrl                 nvarchar(256) OUTPUT
);
```

@SiteId: The site collection identifier of the site collection that contains the Recycle Bin item.

@DeleteTransactionId: The delete transaction identifier of the delete transaction. DeleteTransactionId is a varbinary(16) type, but MUST contain a valid GUID.

@ItemType: A numerical value that indicates the type of the Recycle Bin item. ItemType MUST be one of the values in Delete Item Type.

@ContainingListDeleted: A numeric value that indicates whether the list that contains the Recycle Bin item has been deleted. @ContainingListDeleted can be either NULL or one of the following values:

Value	Description
0	The Recycle Bin item is itself a list or the list that contains the Recycle Bin item has not been deleted.
1	The list that contains the Recycle Bin item has been deleted.

@ContainingListName: The name of the list that contains the Recycle Bin item.

@DirName: The directory name, of the Recycle Bin item.

@WebUrl: The store-relative form of the site which contains the Recycle Bin item.

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

Value	Description
0	Successful completion.
1168	The Recycle Bin item specified by the @DeleteTransactionId parameter could not be found in the Recycle Bin of the site collection specified by the @SiteId parameter.

Result Sets: MUST NOT return any result sets.

3.1.4.28 proc_GetRecycleBinItems

The proc_GetRecycleBinItems stored procedure is called to get the full list of Recycle Bin items for a specified site and User.

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

```
PROCEDUREproc_GetRecycleBinItems(
  @SiteId     uniqueidentifier,
  @WebId      uniqueidentifier,
  @UserId     int
);
```

@SiteId: The site collection identifier of the site collection that contains the Recycle Bin items. SiteId value is ignored.

@WebId: The site identifier of the site that contains the Recycle Bin items.

@UserId: The user identifier of the User who deleted the Recycle Bin item.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.28.1 Recycle Bin Items Result Set

The Recycle Bin Items result set returns the details about a set of Recycle Bin items. There is no limit on the number of rows that the Recycle Bin Items result set can contain. The T-SQL syntax for the result set is as follows.

```
Title               nvarchar(260),
DirName             nvarchar(256),
LeafName            nvarchar(128),
AuthorId            int,
AuthorName          nvarchar(255),
AuthorE-mail        nvarchar(255),
DeleteDate          datetime,
Size                bigint,
DeleteUserId        int,
DeletedByName       nvarchar(255),
DeletedByE-mail     nvarchar(255),
DeleteTransactionId varbinary(16),
ItemType            tinyint,
BinId               tinyint,
{tp_ImageUrl}       nvarchar(255),
ProgId              nvarchar(255);
```

Title: The display name of the Recycle Bin item. Title MUST NOT be NULL.

DirName: The directory name of the Recycle Bin item. DirName MUST NOT be NULL.

LeafName: The file name of the Recycle Bin item. LeafName MUST NOT be NULL.

AuthorId: The identifier of the User who is the author of the Recycle Bin item.

AuthorName: The name of the User who is the author of the Recycle Bin item.

AuthorE-mail: The e-mail address of the User who is the author of the Recycle Bin item.

DeleteDate: The date when the Recycle Bin item was deleted. DeleteDate MUST NOT be NULL.

Size: The size, in Bytes, of the Recycle Bin item. Size MUST NOT be NULL.

DeleteUserId: The user identifier of the User who deleted the Recycle Bin item.

DeletedByName: The login name of the User who deleted the Recycle Bin item.

DeletedByE-mail: The e-mail address of the User who deleted the Recycle Bin item.

DeleteTransactionId: The delete transaction identifier of the delete transaction. DeleteTransactionId is a varbinary(16) value, but it MUST contain a valid GUID and MUST NOT be NULL.

ItemType: A numerical value indicating the type of the Recycle Bin item. ItemType MUST NOT be NULL and MUST be one of the values in Delete Item Type.

BinId: A numerical value that indicates which Recycle Bin Stage the Recycle Bin item is in. BinId MUST NOT be NULL and MUST be 1.

{tp_ImageUrl}: The URL for the icon of the list if the Recycle Bin item is a list or list item.

ProgId: The file type if the Recycle Bin item is a file.

3.1.4.29 proc_GetSiteQuota

The proc_GetSiteQuota stored procedure is called to retrieve quota information about a site collection.

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

```
PROCEDUREproc_GetSiteQuota(
  @WebSiteId    uniqueidentifier
);
```

@WebSiteId: The site collection identifier of the site collection.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.29.1 Quota Information Result Set

The Quota Information result set returns information about the quota settings for the specified site collection. If the specified site collection exists, Quota Information result set MUST contain exactly one row referring to the specified site collection. The T-SQL syntax for the result set is as follows.

```
QuotaTemplateID    smallint,
DiskQuota          bigint,
DiskWarning        bigint,
UserQuota          int;
```

QuotaTemplateID: The **quota template identifier** associated with the site collection. QuotaTemplateID MUST be either zero or NULL if the site collection has no quota templates associated to it.

DiskQuota: The disk quota size in Bytes. DiskQuota Must be zero or NULL if the site collection has no disk quota associated to it.

DiskWarning: The **quota warning level** in Bytes. It MUST be a value between zero and the quota size. DiskWarning MUST be zero or NULL if the site collection has no quota warning level associated to it.

UserQuota: The maximum number of Users. UserQuota MUST be zero or NULL if the site collection has no maximum number of users associated to it. UserQuota can only be larger than zero if the server is in Active Directory account creation mode.

3.1.4.30 proc_GetSiteUsage

The proc_GetSiteUsage stored procedure is called to get the size on disk and bandwidth that the specified site collection is utilizing.

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

```
PROCEDURE proc_GetSiteUsage (
  @WebSiteId    uniqueidentifier
);
```

@WebSiteId: A site collection identifier of the site collection that is being queried.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.30.1 Usage Totals Result Set

The Usage Totals result set returns disk and bandwidth usage for the specified site collection. The Usage Totals result set will be returned always and contain one row. The T-SQL syntax for the result set is as follows.

```
DiskUsed    bigint,
BWUsed      bigint;
```

DiskUsed: An integer that specifies the amount of storage, in Bytes, that the content of the site collection is consuming.

BWUsed: An integer that specifies the cumulative bandwidth, in Bytes, that the site collection consumed on the previous day.

3.1.4.31 proc_GetSizeOfWebPartsOnPage

The proc_GetSizeOfWebPartsOnPage stored procedure is called to get the size, in Bytes, of all the Web Parts on a Web Part page.

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

```
PROCEDURE proc_GetSizeOfWebPartsOnPage (
  @SiteId     uniqueidentifier,
  @WebId      uniqueidentifier,
  @DirName    nvarchar(256),
  @LeafName   nvarchar(128),
  @UserID     int
);
```

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

@WebId: The site identifier of the site,

@DirName: The directory name of the requested document.

@LeafName: The leaf name of the requested document.

@UserId: The user identifier of the User.

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

Value	Description
0	Successful completion.
2	The document does not exist.

Result Sets: MUST return zero result sets if the document does not exist or the following result set when the document exists.

3.1.4.31.1 Webparts Size Result Set

The Webparts Size result set returns the size, in Bytes, of all the Web Parts and personalization data associated with the Web Parts on a Web Part page. This result set MUST return one row. The T-SQL syntax for the result set is as follows.

```
{WebPartSize}    bigint;
```

{WebPartSize}: The size, in Bytes, of all the Web Parts and personalization data associated with the Web Parts on a Web Part page.

3.1.4.32 proc_GetTimerLock

The proc_GetTimerLock stored procedure is called to request a **content database lock** for the specified Server.

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

```
PROCEDURE proc_GetTimerLock(
  @WebServer            nvarchar(255),
  @LockTimeout          int,
  @LockStatus           int OUTPUT,
  @OverrideWebServer    nvarchar(255) OUTPUT
);
```

@WebServer: The name of the server requesting the lock.

@LockTimeout: The maximum age, in minutes, of an existing content database lock before it will be considered expired.

@LockStatus: A Lock Status Type which indicates the status of the requested content database lock. The value is an integer and MUST be one of the following values.

Value	Description
1	Another server already holds the content database lock.
2	Content database lock has been acquired by, or refreshed for, the server requesting the lock.
3	Content database lock acquired by the requesting server. Previous lock had expired.
4	Unexpected failure.

@OverrideWebServer: The name of the server that held an expired lock which was overwritten. This output parameter MUST be set if the @LockStatus output parameter is set to "3".

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

Value	Description
0	Successful completion.
167	An error occurred while retrieving the state of the content database lock.

Result Sets: MUST NOT return any result sets.

3.1.4.33 proc_GetTotalDiscussionsSize

The proc_GetTotalDiscussionsSize stored procedure is called to get the total discussions size, in Bytes, for the specified site.

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

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

@SiteId: The site collection identifier for a site collection.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.33.1 Discussions Size Result Set

The Discussions Size result set returns the total discussions size, in Bytes, for the site. The Discussions Size result set MUST contain one row. The T-SQL syntax for the result set is as follows.

```
{DiscussionsSize}    bigint;
```

{DiscussionsSize}: The total discussions size, in Bytes, of the site collection specified by input parameter @SiteId.

3.1.4.34 proc_GetUniqueScopesInWeb

The proc_GetUniqueScopesInWeb stored procedure is called to get the ACL information for all the unique security scopes of the **subsites** or lists contained in a specified site.

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

```
PROCEDURE proc_GetUniqueScopesInWeb(
  @SiteId           uniqueidentifier,
  @WebId            uniqueidentifier,
  @SelectSubWebs    bit,
  @SelectDoclibs    bit
);
```

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

@WebId: The site identifier of the site.

@SelectSubWebs: A bit specifying whether to get the unique security scopes of the subsites or the lists of the specified site. When set to 1, proc_GetUniqueScopesInWeb MUST return the unique security scopes of the subsites, otherwise it MUST return the unique security scopes of the lists.

@SelectDoclibs: A bit specifying whether to get the unique security scopes of the document libraries. When @SelectSubWebs is set to 1, @SelectDocLibs is ignored. When @SelectDocLibs is set to 0, proc_GetUniqueScopesInWeb MUST return the unique security scopes of all the lists including document libraries, otherwise proc_GetUniqueScopesInWeb MUST return the unique security scopes of document libraries.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.34.1 Unique Scopes under Site Result Set

The Unique Scopes under Site result set returns the ACL information for all the unique security scopes of subsites or lists contained in a specified site. This result set MUST return a row for each security scope found. The T-SQL syntax for the result set is as follows.

```
ScopeId      uniqueidentifier,
Acl          image,
AnonymousPermMask    bigint;
```

ScopeId: The scope identifier of the security scope.

Acl: The binary serialization of the ACL of the security scope in the WSS ACL Format as described in [MS-WSSFO] section 2.2.4.6.

AnonymousPermMask: The WSS Rights Mask, as described in [MS-WSSFO] section 2.2.2.13, that indicates the rights granted to an Anonymous User or a User who has no specific rights in this security scope.

3.1.4.35 proc_GetUserStorageInfo

The proc_GetUserStorageInfo stored procedure is called to get the user identifier, login name and the size, in Bytes, of the personalizations on a particular Web Part page.

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

```
PROCEDURE proc_GetUserStorageInfo (
  @SiteId      uniqueidentifier,
  @WebId       uniqueidentifier,
  @DirName     nvarchar(256),
  @LeafName    nvarchar(128)
);
```

@SiteId: The site collection identifier of the site collection that contains the requested document.

@ WebId: The site identifier of the site that contains the requested document.

@DirName: The directory name of the requested document.

@LeafName: The leaf name of the requested document.

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

Value	Description
0	Successful completion.
2	The document does not exist.

Result Sets: MUST return zero result sets if the document does not exist or the following result set when the document exists.

3.1.4.35.1 User Storage Info Result Set

The User Storage Info result set returns the user identifier, login name and the total size, in Bytes, of the personalizations and Web Parts on a particular Web Part page. There is one row returned for each user that customizes the Web Part page.

The T-SQL syntax for the result set is defined in the User Storage Info Result Set.

3.1.4.36 proc_MoveRecycleBinItemToSecondStage

The proc_MoveRecycleBinItemToSecondStage stored procedure is called to move a Recycle Bin item from the first-stage Recycle Bin to the second-stage Recycle Bin.

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

```
PROCEDURE proc_MoveRecycleBinItemToSecondStage(
  @SiteId                       uniqueidentifier,
  @WebId                        uniqueidentifier,
  @UserId                       int,
  @DeleteTransactionId          varbinary(16),
  @SecondStageRecycleBinQuota   int,
  @SpaceRequired                bigint OUTPUT,
  @DiskUsedBefore               bigint= 0 OUTPUT,
```

```
    @DiskUsedAfter                bigint= 0 OUTPUT,
    @MaxSSDiskSpace               bigint= 0 OUTPUT
  );
```

@SiteId: The site collection identifier of the site collection which contains the Recycle Bin item. This parameter MUST NOT be NULL.

@WebId: The site identifier of the site which contains the Recycle Bin item. If the @UserId parameter is NOT 0, then this parameter MUST NOT be NULL.

@UserId: A numeric value that MUST either be 0 or the user identifier of the User who deleted the Recycle Bin item. This parameter MUST NOT be NULL.

@DeleteTransactionId: The delete transaction identifier of the delete transaction. DeleteTransactionId is a varbinary(16) type, but it MUST contain a valid GUID and MUST NOT be NULL.

@SecondStageRecycleBinQuota: The percentage of the site collection quota allocated to the second-stage Recycle Bin. This parameter MUST NOT be NULL and MUST be a positive whole number. If the size of the Recycle Bin item is larger than the size determined by this percentage of the site collection quota, then the Recycle Bin item MUST be permanently deleted. If the size of the Recycle Bin item is less than the size determined by this percentage of the site collection quota, but there is not enough free space within the size determined by this percentage of the site collection quota, then the Recycle Bin item MUST NOT be permanently deleted and MUST NOT be moved to the second-stage Recycle Bin.

@SpaceRequired: A value indicating **t**he amount of additional space, in Bytes, that would be required to store the Recycle Bin item in the second-stage Recycle Bin for the specified @SecondStageRecycleBinQuota parameter and the site collection quota.

@DiskUsedBefore: A value indicating the amount of storage space, in Bytes, used by the second-stage Recycle Bin before moving the Recycle Bin item to the second-stage Recycle Bin. This parameter MUST NOT be NULL.

@DiskUsedAfter: A value indicating the amount of storage space, in Bytes, used by the second-stage Recycle Bin after moving the Recycle Bin item to the second-stage Recycle Bin. This parameter MUST NOT be NULL.

@MaxSSDiskSpace: A value indicating the maximum available storage space, in Bytes, of the second-stage Recycle Bin for the specified @SecondStageRecycleBinQuota parameter and the site collection quota. This parameter MUST NOT be NULL.

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

Value	Description
0	Successful completion.
112	There is not enough space on the disk. This happens if the @SecondStageRecycleBinQuota parameter is 0 or negative, or if there is not enough remaining space in the specified percentage of the site collection quota to contain the Recycle Bin item.
1168	The Recycle Bin item specified by either the @SiteId and @DeleteTransactionId parameters or the @SiteId, @WebId, @UserId, and @DeleteTransactionId parameters could not be found.
1359	An internal error occurred.

Result Sets: MUST NOT return any result sets.

3.1.4.37 proc_QMGetDiskWarning

The proc_QMGetDiskWarning stored procedure is called to get a list of sites whose disk used size, in Bytes, is larger than the quota warning level.

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

```
PROCEDURE proc_QMGetDiskWarning(
  @dtLast     datetime OUTPUT,
  @dtCur      datetime OUTPUT
);
```

@dtLast: An output parameter that contains the date and time when the last **quota warning** was sent.

@dtCur: An output parameter that contains the current date and time.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.37.1 Disk Warning Result Set

The Disk Warning result set returns the information needed for the quota warnings to be sent out. The Disk Warning result set will contain all the site collections whose size is larger than the quota warning level, the e-mail addresses of their site collection administrators, the URLs, and their language code identifiers (LCIDs). The T-SQL syntax for the result set is as follows.

```
FullUrl      nvarchar(256),
Id           uniqueidentifier,
tp_E-mail    nvarchar(255),
Language     int;
```

FullUrl: The URL of the site. This MUST be blank if it is the top-level site in the site collection.

Id: The site identifier of the site collection.

tp_E-mail: The e-mail address of the site collection administrator.

Language: The LCID which indicates the language of the top-level site.

3.1.4.38 proc_QMMarkDiskWarning

The proc_QMMarkDiskWarning stored procedure is called to update the stored date and time of the last quota warning.

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

```
PROCEDURE proc_QMMarkDiskWarning(
  @dtLast     datetime,
  @dtCur      datetime
);
```

@dtLast: The date and time when the last quota warning was sent.

@dtCur: The current date and time.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.39 proc_RenameTimerLock

The proc_RenameTimerLock stored procedure is called to update the name of a specified server in the database tables that store content database locks**.**

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

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

@OldServerName: The old name of the server being renamed.

@NewServerName: The new name of the server being renamed.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.40 proc_RestoreRecycleBinItem

The proc_RestoreRecycleBinItem stored procedure is called to restore a Recycle Bin item.

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

```
PROCEDURE proc_RestoreRecycleBinItem(
  @SiteId               uniqueidentifier,
  @WebId                uniqueidentifier,
  @UserId               int,
  @DeleteTransactionId  varbinary(16)
);
```

@SiteId: The site collection identifier of the site collection that contains the Recycle Bin item. This parameter MUST NOT be NULL.

@WebId: The site identifier of the site that contains the Recycle Bin item. If the @UserId parameter is NOT 0, then this parameter MUST NOT be NULL.

@UserId: A numeric value that MUST either be 0 or the user identifier of the User who deleted the Recycle Bin item. This parameter MUST NOT be NULL.

@DeleteTransactionId: The delete transaction identifier of the delete transaction. DeleteTransactionId is a varbinary(16) type, but it MUST contain a valid GUID and MUST NOT be NULL.

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

Value	Description
0	Successful completion.
3	The Recycle Bin item could not be restored because the parent folder of the Recycle Bin item, as indicated by the item's DirName, could not be found.
80	The Recycle Bin item could not be restored for one of the following reasons: ▪ The Recycle Bin item is a list item in a **survey list** that doesn't allow multiple votes and another vote is already present for the User. ▪ The Recycle Bin item is a file, list, folder, or attachment and another document with the same path exists. ▪ The Recycle Bin item is a list and another list with the same display name exists. ▪ The Recycle Bin item is a folder which contains lists and one or more lists exist with the same display name as one of the lists in that folder.
87	The Recycle Bin item is a file or an attachment and could not be restored because an error occurred recreating the image or thumbnail for the Recycle Bin item.
212	The site collection is **locked** and the Recycle Bin item cannot be restored.
1168	Either the Recycle Bin item could not be found or the site that contains the Recycle Bin item could not be found.
1359	An internal error occurred.
1816	The site collection does not have enough remaining quota available to restore the Recycle Bin item.
1979	The Recycle Bin item could not be restored for one of the following reasons: ▪ The list that contains the Recycle Bin item could not be found. ▪ The Recycle Bin item is a file version, but the corresponding file could not be found. ▪ The Recycle Bin item is an attachment or list item version, but the corresponding list item could not be found.

Result Sets: MUST NOT return any result sets.

3.1.4.41 proc_RevertDocContentStreams

The proc_RevertDocContentStreams stored procedure is called to revert a customized document to a previous uncustomized state.

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

```
PROCEDURE proc_RevertDocContentStreams (
  @SiteId        uniqueidentifier,
  @WebId         uniqueidentifier,
  @WebFullUrl    nvarchar(260),
  @DocDirName    nvarchar(256),
  @DocLeafName   nvarchar(128),
  @UserId        int
```

```
);
```

@SiteId: The site collection identifier of the site collection that contains the requested document.

@WebId: The site identifier of the site that contains the requested document.

@WebFullUrl: This parameter MUST be ignored.

@ DocDirName: The directory name of the requested document.

@ DocLeafName: The leaf name of the requested document.

@UserId: The user identifier of the User calling the stored procedure.

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

Value	Description
0	Successful completion.
2	The specified document was NOT found.

Result Sets: MUST NOT return any result sets.

3.1.4.42 proc_ScorchList

The proc_ScorchList stored procedure is called to delete a list and all its contents if the list belongs to one of the following three categories:

- Lists with no parent site
- Lists with documents that have no parent list
- Lists with items that have no parent list.

It is not used on any other type of list. To delete a list that does not match one of the three criteria earlier in this section, use proc_DeleteUrl instead.

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

```
PROCEDURE proc_ScorchList(
  @ListId    uniqueidentifier
);
```

@ListId: The list identifier of the list to be deleted.

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

Value	Description
0	Successful completion.
3	The list does not exist.
8398	A list marked as undeletable cannot be deleted.

Result Sets: MUST NOT return any result sets.

3.1.4.43 proc_ScorchWeb

The proc_ScorchWeb stored procedure is called to delete an orphaned folder with no site associated. To delete a site that does not meet this criterion, use proc_DeleteWeb, as defined in [MS-WSSDLIM], instead.

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

```
PROCEDURE proc_ScorchWeb (
  @WebId    uniqueidentifier
);
```

@WebId: The site identifier of the site to be deleted. @WebId contains the orphaned folder.

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

Value	Description
0	Successful completion.
3	The site does not exist.

Result Sets: MUST NOT return any result sets if the site does not exist. Otherwise, it MUST return the following result set.

3.1.4.43.1 Audit Mask Result Set

The Audit Mask result set returns audit configuration information. The Audit Mask result set MUST be returned with one row of audit configuration information upon successful completion.

The Audit Mask result set is defined in [MS-WSSFO].

3.1.4.44 proc_SecBackupAllWebMembers

The proc_SecBackupAllWebMembers stored procedure is called to return information about All Site Members.

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

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

@SiteID: The site collection identifier that contains the site.

@WebID: The site identifier whose All Site Members are to be returned.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.44.1 UserInfo Result Set

The UserInfo result set returns information about All Site Members. The result set contains one row per non-deleted User. The T-SQL syntax for the result set is as follows.

```
tp_ID      int,
tp_Title    nvarchar(255),
tp_Login    nvarchar(255),
tp_E-mail    nvarchar(255),
tp_Notes    nvarchar(1023),
tp_Deleted    int;
```

tp_ID: The user identifier of the user.

tp_Title: The display name of the user.

tp_Login: The login name of the user.

tp_E-mail: The e-mail address of the user.

tp_Notes: A string that contains information about the user.

tp_Deleted: An integer value indicating whether the user has been deleted from the site collection. This value MUST be 0.

3.1.4.45 proc_SecGetListItemSecurity

The proc_SecGetListItemSecurity stored procedure is called to get the security information of a list item.

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

```
PROCEDURE proc_SecGetListItemSecurity(
  @SiteId              uniqueidentifier,
  @ListId              uniqueidentifier,
  @ListRootFolderUrl   nvarchar(260),
  @ItemId              int
);
```

@SiteId: The site collection identifier of the site collection that contains the requested list item.

@ListId: The list identifier of the list that contains the requested list item.

@ListRootFolderUrl: The store-relative form of the root folder of the list. It MUST match the list specified by the @ListId.

@ItemId: The item identifier of the requested list item.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set.

3.1.4.45.1 Access Control List Result Set

The Access Control List result set returns ACL information of the list item specified by the input parameters. The Access Control List result set MUST contain one row if the list item exists.

Otherwise, zero rows if the list item does not exist or if the @ListRootFolderUrl and @ListId do not point to the same list. The T-SQL syntax for the result set is as follows.

```
Acl                 image,
AnonymousPermMask   bigint;
```

Acl: The binary serialization of the ACL in effect for the list item, in the format described in [MS-WSSFO] section 2.2.4.6.

AnonymousPermMask: The WSS Rights Mask, as described in [MS-WSSFO] section 2.2.2.13, that indicates the rights granted to any Anonymous User for this list item.

3.1.4.46 proc_SecRemoveExternalSecurityProvider

The proc_SecRemoveExternalSecurityProvider stored procedure is called to remove the External Security Provider that enforces User authorization on the site.

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

```
PROCEDURE dbo.proc_SecRemoveExternalSecurityProvider(
    @SiteId uniqueidentifier,
    @WebId uniqueidentifier
);
```

@SiteId: The site collection identifier of the site collection that contains the specified site, @SiteId MUST NOT be NULL.

@WebId: The site identifier of the site whose External Security Provider will be removed. @WebId MUST NOT be NULL.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.47 proc_SetAuditMask

The proc_SetAuditMask stored procedure is called to set the audit flags, as described in [MS-WSSFO], directly applicable to an object.

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

```
PROCEDURE proc_SetAuditMask(
  @ItemType      tinyint,
  @SiteId        uniqueidentifier,
  @DirName       nvarchar(256),
  @LeafName      nvarchar(128),
  @AuditFlags    int,
);
```

@ItemType: The type of object which MUST be a value from the Audit Item Type enumeration described in [MS-WSSFO].

@SiteId: The site collection identifier of the site collection which contains the object for which the audit flags are being requested.

@DirName: The directory name of the object.

@LeafName: The leaf name of the object.

@AuditFlags: The audit flags directly applicable to the object, which MUST be an audit flags bit mask as described in [MS-WSSFO].

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

Value	Description
0	Successful completion.
2	If the object identified by the specified @ItemType, @SiteId, @DirName and @LeafName parameters does not exist , the protocol server SHOULD return 2<4>.

Result Sets: MUST NOT return any result sets.

3.1.4.48 proc_SetDeadWebNotificationCount

The proc_SetDeadWebNotificationCount stored procedure is called to set a value of notify count for the specified site collection and to log the update event.

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

```
PROCEDURE proc_SetDeadWebNotificationCount (
  @SiteId         uniqueidentifier,
  @NotifyCount    smallint
);
```

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

@ NotifyCount: The value of notify count. This value MUST NOT be NULL.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.49 proc_SetListRequestAccess

The proc_SetListRequestAccess stored procedure is called to update the request access setting of the list specified by the @ListID parameter. When the parameter @RequestAccess is set to 1, access requests are enabled and a User that gets access denied can then submit a request to access the list. When the parameter @RequestAccess is set to 0, access requests are disabled and a User that gets access denied will not be able to request access to the list.

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

```
PROCEDURE proc_SetListRequestAccess(
    @SiteID         uniqueidentifier,
    @WebID          uniqueidentifier,
    @ListID         uniqueidentifier,
    @RequestAccess  bit
);
```

@SiteID: The site collection identifier of the site collection that contains the specified list. @SiteID MUST be a GUID.

@WebID: The site identifier of the site that contains the specified list. @WebID MUST be a GUID.

@ListID: The list identifier of the list. @ListID MUST be a GUID.

@RequestAccess: A bit flag specifying whether the request access setting of the list is enabled or disabled. If the flag is set to 0, the list does not allow access requests. If the flag is set to 1 or NULL, the list allows access requests.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.50 proc_SetSiteQuota

The proc_SetSiteQuota stored procedure is called to set quota settings on a specified site collection.

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

```
PROCEDURE proc_SetSiteQuota(
  @WebSiteId    uniqueidentifier,
  @quotaId      smallint,
  @diskQuota    bigint,
  @diskWarning  bigint,
  @userQuota    int
);
```

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

@quotaId: The value that specifies the quota template associated with the site collection. This can be NULL if the site collection is not to be associated to any quota template. When this is not NULL, the @diskQuota, @diskWarning and @userQuotaparameters MUST be set to the corresponding values in the quota template.

@diskQuota: The quota size in Bytes. This can be NULL if no quota is wanted for the site collection.

@diskWarning: The disk space, in Bytes, that will trigger a warning message. This MUST be a value between zero and @diskQuota. This can be NULL if no warning message is wanted for the site collection.

@userQuota: The maximum number of Users allowed for the site collection. This MUST be NULL in **domain account mode**. In Active Directory account creation mode, @userQuota can be either a nonzero or NULL if no maximum number is defined.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.51 proc_SizeOfPersonalizationsPerUser

The proc_SizeOfPersonalizationsPerUser is called to get the user identifier, login name, and the total size, in Bytes, of the personalizations and Web Parts on a particular Web Part page.

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

```
PROCEDURE proc_SizeOfPersonalizationsPerUser (
  @SiteId    uniqueidentifier,
  @DocId     uniqueidentifier
);
```

@SiteId: The site collection identifier of the site collection that contains the requested document.

@DocId: The document identifier of the document.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return a following result set:

3.1.4.51.1 User Storage Info Result Set

The User Storage Info result set returns the user identifier, login name and the total size, in Bytes, of the personalizations and Web Parts on a particular Web Part page. There is one row returned for each user that customizes the Web Part page.

The T-SQL syntax for the result set is defined in the User Storage Info Result Set.

3.1.4.52 proc_UpdateStatistics

The proc_UpdateStatistics stored procedure is called to update statistics that are used internally by the back-end database server on user tables in the content database. Statistics are used by the database server to optimize query performance.

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

```
PROCEDURE proc_UpdateStatistics();
```

proc_UpdateStatistics MUST NOT take any parameters.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.53 proc_GetDatabaseInformation

The proc_GetDatabaseInformation<5> stored procedure is called to retrieve a specific property which is stored in the database, and referenced by a name.

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

```
PROCEDURE proc_ GetDatabaseInformation (
  @Name    nvarchar(128)
);
```

@Name: The name of the property to be retrieved.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.53.1 Database Information Result Set

If the back-end server finds a property whose name matches @Name, the Database Information result set returns the value of that property. If no property stored on the server has the same name as @Name, the server will return for any property whose name starts with @Name. The T-SQL syntax for the result set is as follows.

```
Name      nvarchar(128)
Value     nvarchar(1023)
```

Name: Name of the property

Value: Value of the property

The result set is sorted in the order of the Name column.

3.1.4.54 proc_SetDatabaseInformation

The proc_SetDatabaseInformation<6> stored procedure is called to update a specific property which is stored in the database and referenced by a name. If the property does not exist, then the proc_SetDatabaseInformation stored procedure creates the property and sets its value.

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

```
PROCEDURE proc_ SetDatabaseInformation (
  @Name       nvarchar(128),
  @Value      nvarchar(1023)
);
```

@Name: The name of the property to be saved.

@Value: The value of the property to be saved.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.55 proc_DirtyDocWithForwardLinks

The proc_DirtyDocWithForwardLinks Stored Procedure is called to mark all documents which contain **forward link**(s) in the content database to be **dirty**.

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

```
PROCEDURE proc_DirtyDocWithForwardLinks();
```

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.56 proc_DirtyDocWithForwardLinksInSite

The proc_DirtyDocWithForwardLinksInSite Stored Procedure is called to mark all documents which contain forward link(s) in a site collection to be dirty.

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

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

@SiteId: Identifier of the site collection

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.5 Timer Events

If the execution timeout event is triggered, the execution of the stored procedure is terminated and the call fails

3.1.6 Other Local Events

None.

3.2 Front-end Web Server Client Details

The front-end Web server acts as a client when it calls the back-end database server to request execution of stored procedures.

3.2.1 Abstract Data Model

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

The front-end Web server can maintain the following sets of data for this protocol within object structures. There is no requirement for the data within these structures to be a complete representation of all data maintained on the back-end database server. They can be populated as various requests to the back-end database server are fulfilled. Data maintained on the front-end Web server can be discarded after individual sequences of requests have finished as part of a response for a higher level event.

- Configuration
- Site Collections
- Sites
- Lists
- List Items
- Documents
- Users
- Groups

3.2.2 Timers

A connection timeout timer is set up on the front-end Web server to govern the total connection time for any requests to the back-end database server. The amount of time is governed by a timeout value configured on the front-end Web server for all back-end database server connections.

3.2.3 Initialization

The front-end Web server MUST validate the user making the request before calling the stored procedures. The site collection identifier and the user identifier for the user making the request are looked up by the front-end Web server before calling additional stored procedures.

3.2.4 Message Processing Events and Sequencing Rules

The front-end Web server handles each stored procedure with the same processing method of calling the stored procedure and waiting for the return code and any result sets that will be returned.

The front-end Web server can execute dynamically generated SQL queries against the stored procedures or the tables and views used within the database. Unless otherwise specified, any data addition, removal, or modification MUST occur only by calling the listed stored procedures. SQL queries MUST NOT attempt to add, remove, or update data in any table or view in the content database or configuration database, unless explicitly described in this section.

3.2.5 Timer Events

If the connection timeout event is triggered, the connection and the stored procedure call fails.

3.2.6 Other Local Events

None

4 Protocol Examples

This section provides specific example scenarios for end-to-end data query and update comments as part of file, user, and group administration operations. These examples describe in detail the process of communication between the various server components.

4.1 Auditing Operations

This example describes the requests a protocol client makes to change the audit mask of a document.

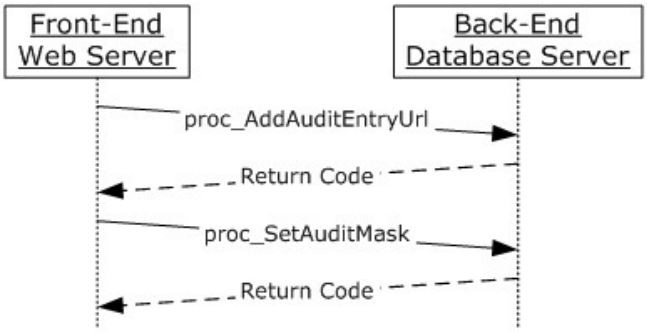

Figure 9: Changing an Audit Mask for a Document

First, the protocol client sends the proc_AddAuditEntryUrl T-SQL message to the protocol server to audit the fact that it is about to set the audit mask.

The protocol server adds the audit entry and returns 0. Next, the protocol client sends the proc_SetAuditMask T-SQL message to the protocol server to set the audit flags as specified in [MS-WSSFO] section 2.2.2.1 to audit view operations on the document 'Documents/test.doc':

The protocol server sets the audit mask for the document and returns 0.

4.2 Quota Management Operations

This section provides examples of quote management operations.

4.2.1 Querying Quota

This example describes the requests made and responses returned when a User retrieves the quota information for a site collection.

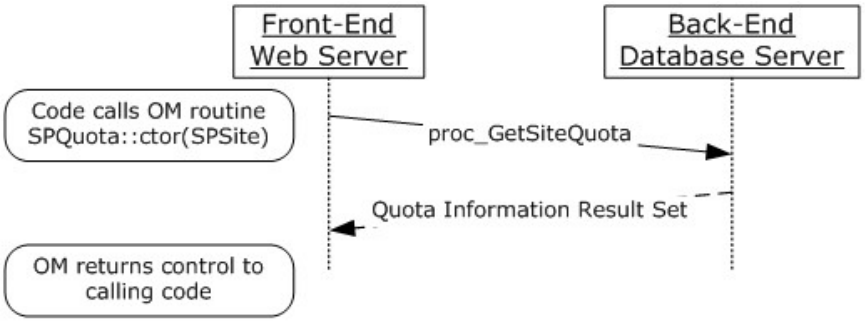

Figure 10: Retrieving Quota Information for a Site Collection

This scenario is initiated when a new SPQuota object is created using SPQuota(SPSite) constructor in the object model.

For simplicity's sake the example assumes that:

1. The code has already instantiated the required site collection object
2. The code has already instantiated the quota template object if the site collection uses a quota template to set its quota.

The following actions happen:

1. The front-end Web server requests for the quota information of a site by calling the stored procedure proc_GetSiteQuota.
2. The stored procedure returns the Quota Information Result Set.
3. The front-end Web server constructs the SPQuota object using the values from the Quota Information Result Set.

4.2.2 Updating Quota

This example describes the requests made and responses returned when a User requests to update the quota information for a site collection. The quota for a site collection can be set either from a quota template or the values can be set directly.

4.2.2.1 Setting quota from a Quota Template

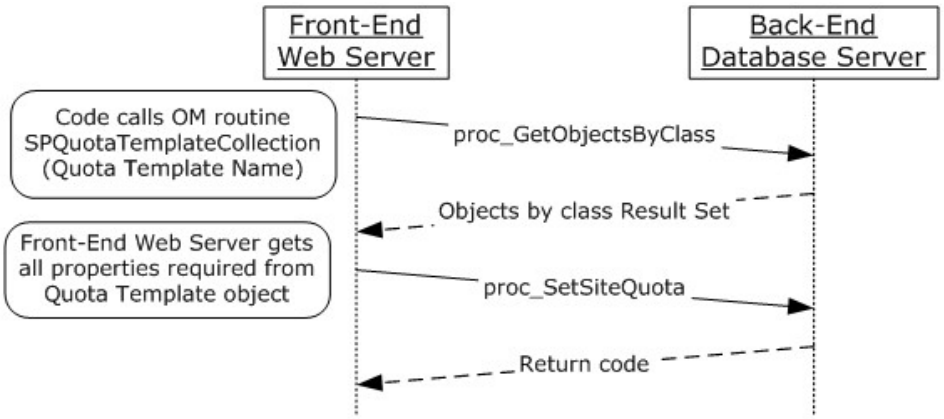

Figure 11: Setting a Quota Using a Quota Template

The following actions happen:

1. The front-end Web server requests for all the quota templates available by calling the stored procedure proc_GetObjectsByClass.
2. When a User selects a particular quota template to be applied, the front-end Web server constructs the quota template object from the serialized data.

3. The front-end Web server gets the information about the allowed disk space, disk space warning limit, and the maximum number of users allowed if in Active directory account creation mode from the quota template object.

4. The front-end Web server updates the quota for the site collection using a call to the proc_SetSiteQuota stored procedure specifying the quota template identifier, disk space, disk space warning limit, and maximum number of users allowed as parameters.

4.2.3 Get Usage Information for a Site Collection

The front-end Web server calls the proc_GetSiteUsage stored procedure to get the usage information for a site collection.

Figure 12: Retrieving Usage Information for a Site Collection

4.2.4 Warning Site Collections which are near the allowed disk space

This example describes the actions that the front-end Web server performs during a timer job to get information about site collections that are near the limits set by a quota and the owners that need to be warned.

The following diagram shows the sequence of calls the front-end Web server performs for quota warning operations.

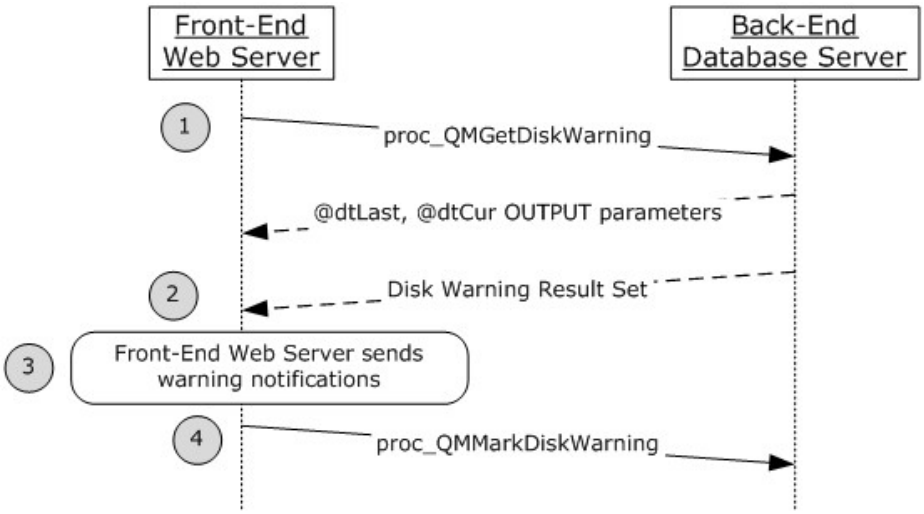

Figure 13: Warning Site Collections near a Quota Limit

The actions that happen are:

1. The front-end Web server calls the proc_QMGetDiskWarning stored procedure on a content database.

2. The back-end database server returns the last time when a quota warning was sent, the current time, and the list of all site collections along with the e-mail addresses of their site collection administrators who have crossed the quota warning limits and need to be warned.

3. The front-end Web server sends the notifications to the site collection administrators of the site collections.

4. The front-end Web server calls the procQMMarkDiskWarning indicating to the back-end database server that the notifications have been sent for the site collections.

4.3 Recycle Bin Operations

This section provides examples of recycle bin operations.

4.3.1 Query items in First-Stage Recycle Bin

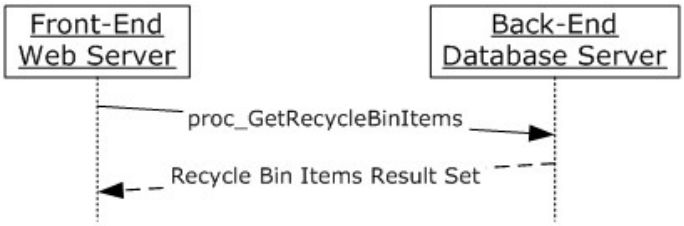

Figure 14: Retrieve Items in the First-Stage Recycle Bin

This example describes the actions the front-end Web server takes when querying items in the first-stage Recycle Bin. The actions that happen are:

1. The front-end Web server calls the proc_GetRecycleBinItems stored procedure on a content database.

2. The back-end database server returns a Recycle Bin items result set, as defined in Recycle Bin Items Result Set.

4.3.2 Delete a first-stage Recycle Bin item to second-stage Recycle Bin

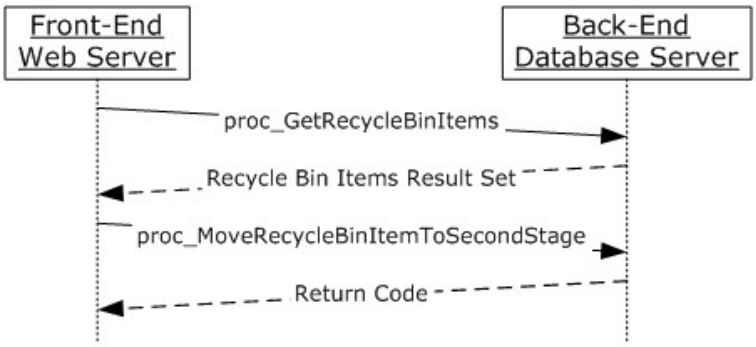

Figure 15: Move an Item from a First-stage Recycle Bin to a Second-stage Recycle Bin

This example describes the actions a front-end Web server takes when querying and deleting an item in the first-stage Recycle Bin. The actions that happen are:

1. The front-end Web server calls the proc_GetRecycleBinItems stored procedure on a content database.
2. The back-end database server returns a Recycle Bin Items result set as defined in Recycle Bin Items Result Set.
3. The front-end Web server chooses an item from the result set to delete and calls proc_MoveRecycleBinItemToSecondStage with appropriate parameters.
4. The back-end database server returns an error code indicating if the operation succeeded or not.

4.3.3 Restore a first-stage Recycle Bin item

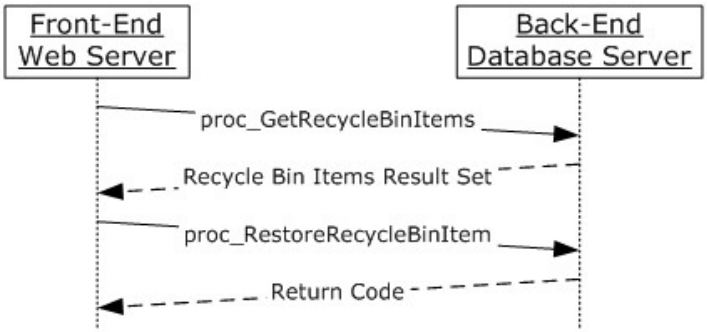

Figure 16: Restore an item from the First-stage Recycle Bin

This example describes the actions the front-end Web server takes when querying and restoring an item in the first-stage Recycle Bin to a list. The actions that happen are:

1. The front-end Web server calls the proc_GetRecycleBinItems stored procedure on a content database to enumerate items in the first-stage Recycle Bin.
2. The back-end database server returns a Recycle Bin Items result set as defined in Recycle Bin Items Result Set.
3. The front-end Web server chooses an item from the result set to restore and calls proc_RestoreRecycleBinItem with appropriate parameters.
4. The back-end database server returns an error code indicating if the operation succeeded or not.

4.3.4 Delete a second-stage Recycle Bin Item

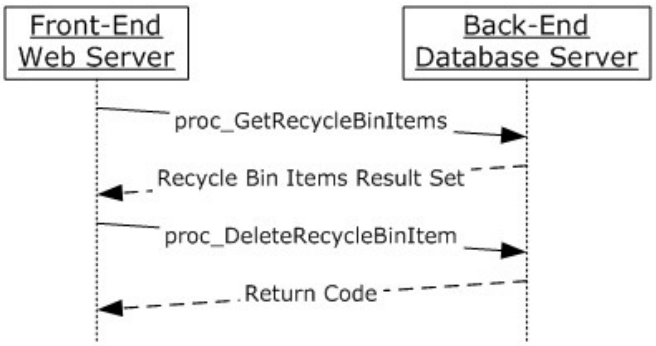

Figure 17: Delete a Second-stage Recycle Bin Item

This example describes the actions a front-end Web server takes when querying an item in the second-stage Recycle Bin and deleting it permanently from the physical disk of the back-end database server. The actions that happen are:

1. The front-end Web server calls the proc_GetRecycleBinItems stored procedure on a content database to enumerate items in the second-stage Recycle Bin.
2. The back-end database server returns a Recycle Bin Items result set as defined in Recycle Bin Items Result Set.
3. The front-end Web server chooses an item from the result set to delete and calls proc_DeleteRecycleBinItem.
4. The back-end database server returns an error code indicating if the operation succeeded or not.

4.4 Security Operations

This section provides examples of security operations.

4.4.1 Remove External Security Provider

This example describes the interactions made when a user removes External Security Provider from a site.

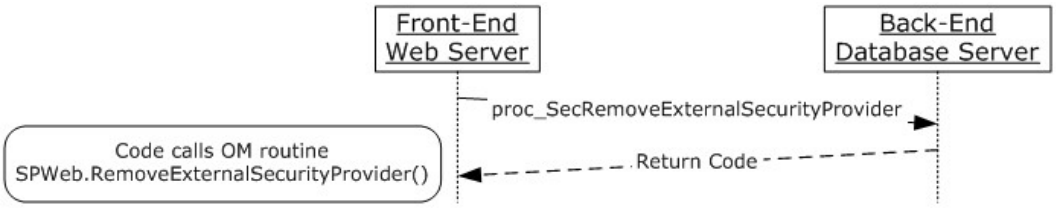

Figure 18: Removing an External Security Provider

This scenario is initiated by a call to the object model command SPWeb.RemoveExternalSecurityProvider(). For simplicity's sake, this example assumes that the code has already instantiated the site collection (SPSite) and site (SPWeb) object.

The following actions happen:

1. The front-end Web server calls the stored procedure proc_SecRemoveExternalSecurityProvider using the site collection identifier and site identifier that was initialized before.

2. The back-end database server returns return code 0.

4.4.2 Get ACL of a specific SPListItem

This example describes the interactions made when the user wants to get the ACL about a specific list item.

Figure 19: Retrieving an ACL for a List Item

This scenario is initiated when the user wants to update a list item with unique permission. The front-end Web server checks if the user has enough permission before updating the item. For simplicity's sake, this example assumes that the code has already instantiated the site collection (SPSite), site (SPWeb), and list (SPList) objects.

The following actions happen:

1. The front-end Web server calls the stored procedure proc_SecGetListItemSecurity using the site collection identifier, list identifier, and list root folder that was initialized before.

2. The back-end database server returns the ACL and WSS Rights Masks for anonymous users as defined in [MS-WSSFO] section 2.2.2.13 of the list item.

3. The front-end Web server uses the returned ACL and WSS Rights Masks for anonymous users as defined in [MS-WSSFO] section 2.2.2.13 to verify that the current user has enough permission to update the list item.

4.4.3 Retrieve All Site Members

This example describes the interactions made when the user wants to get information about All Site Members such as for backup.

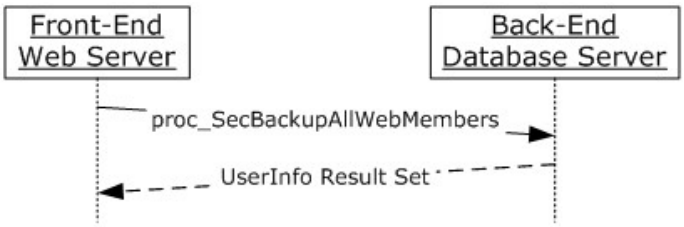

Figure 20: Retrieve All Site Members

The front-end Web server gets All Site Members from the back-end database server and returns to the user. For simplicity's sake, this example assumes that the code has already instantiated the site collection (SPSite) and site (SPWeb) objects.

The following actions happen:

1. The front-end Web server calls the stored procedure proc_SecBackupAllWebMembers using the site collection identifier and site identifier that were initialized before.
2. The back-end database server returns the UserInfo Result Set for the site.
3. The front-end Web server puts the returned UserInfo Result Set into an XML file to return to the caller.

4.5 Database Integrity and Maintenance Operations

This section provides examples of database integrity and maintenance operations.

4.5.1 Find Orphaned Objects for Repair

This example describes the interaction between the front-end Web server and the back-end database server when searching for orphaned objects for repair purposes.

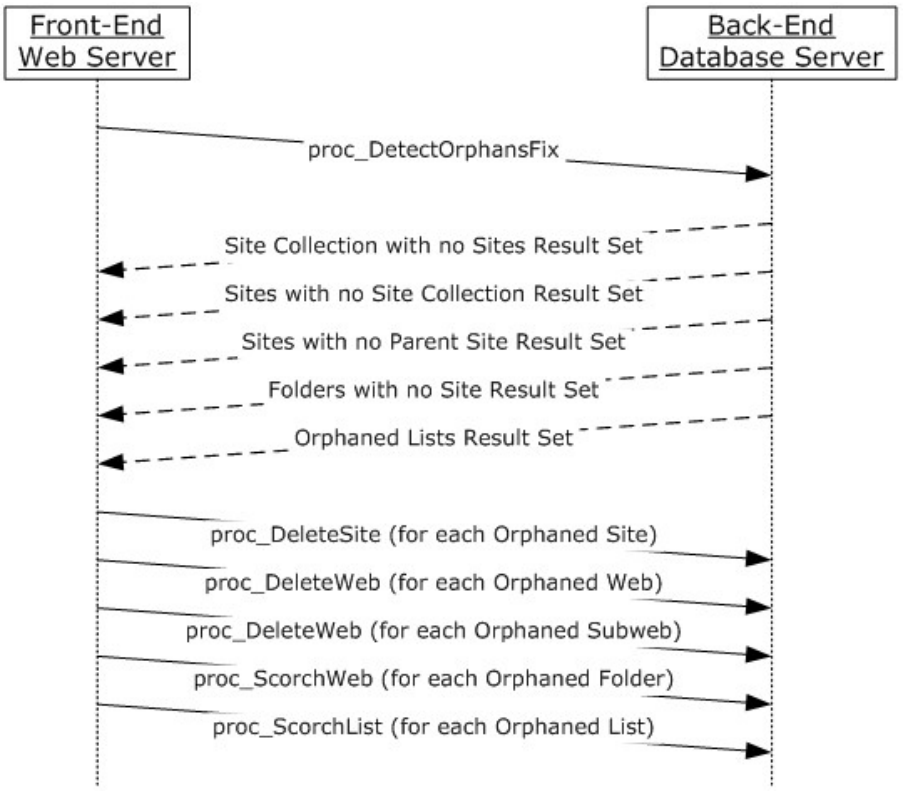

Figure 21: Find Orphaned Objects to Repair

The following actions occur:

1. The front-end Web server calls the stored procedure proc_DetectOrphansFix.
2. The back-end database server returns five result sets as follows:

- Site Collection with no Sites result set (orphaned site collections) as defined in Site Collection with no Sites Result Set.
- Sites with no Site Collection result set (orphaned sites) as defined in Sites with no Site Collection Result Set.
- Sites with no Parent Site result set (orphaned subsites) as defined in Sites with no Parent Site Result Set.
- Folders with no Site result set (orphaned folders) as defined in Folders with no Site Result Set.
- Orphaned Lists result set as defined in Orphaned Lists Result Set.

3. The front-end Web server deletes orphaned site collections by calling the stored procedure proc_DeleteSite, as defined in [MS-WSSDLIM], for each site collection.
4. The front-end Web server deletes orphaned sites by calling stored procedure proc_DeleteWeb, as defined in [MS-WSSDLIM], for each site.
5. The front-end Web server deletes orphaned subsites by calling stored procedure proc_DeleteWeb, as defined in [MS-WSSDLIM], for each subsite.
6. The front-end Web server deletes orphaned folders by calling the stored procedure proc_ScorchWeb for each folder.
7. The front-end Web server deletes orphaned lists by calling the stored procedure proc_ScorchList for each list.

5 Security

5.1 Security Considerations for Implementers

Security for this protocol is controlled by the permissions to the databases on the back-end database server, which is negotiated as part of the Tabular Data Stream [MS-TDS] protocol.

The database access account used by the front-end Web server must have access to the appropriate content database on the back-end database server. If the account does not have the correct permissions, access will be denied when attempting to set up the [MS-TDS] connection to the content database or when calling the stored procedures.

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

5.2 Index of Security Parameters

Security Parameter	Section	
proc_SecBackupAllWebMembers	3.1.4.39	
proc_SecGetListItemSecurity	3.1.4.40	
proc_SecRemoveExternalSecurityProvider	3.1.4.41	

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® Office SharePoint® Server 2007
- Microsoft® SQL Server® 2005
- Microsoft® SQL Server® 2008
- Microsoft® SQL Server® 2008 R2
- Windows® SharePoint® Services 3.0

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

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

<1> Section 2.2.8.1: In Windows SharePoint Services 3.0, **DocLocation** is a leaf name if **EventSource** is 0x01 (Object Model). Otherwise, the url is either the store-relative form or a **server-relative URL**.

<2> Section 3.1.4.6: In Windows SharePoint Services 3.0, the **@Location** is a leaf name if **@EventSource** is 0x01 (Object Model). Otherwise, the url is either the store-relative form or a server-relative URL.

<3> Section 3.1.4.7: In Windows SharePoint Services 3.0, the **@Location** is a leaf name if **@EventSource** is 0x01 (Object Model). Otherwise, the url is either the store-relative form or a server-relative URL.

<4> Section 3.1.4.47: In Windows SharePoint Services 3.0, if **@ItemType** is 0x7 (Site Collection), the Return Value is always ERROR_FILE_NOT_FOUND. Otherwise, the Return Value is always ERROR_SUCCESS.

<5> Section 3.1.4.53: Prior to the Infrastructure Update for Windows SharePoint Services 3.0 (KB951695), the stored procedure proc_GetDatabaseInformation MUST NOT be used.

<6> Section 3.1.4.54: Prior to the Infrastructure Update for Windows SharePoint Services 3.0 (KB951695), the stored procedure proc_SetDatabaseInformation MUST NOT be used.

7 Change Tracking

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

8 Index

Q

R

S

T

U

V

X