[MS-WSSCCSP]: Windows SharePoint Services: Content Database Core List Schema and Site Provisioning Communications Protocol Specification

Revision Summary

Date	Revision History	Revision Class	Comments
04/04/2008	0.1		Initial Availability
06/27/2008	1.0	Major	Revised and edited the technical content
12/12/2008	1.01	Editorial	Revised and edited the technical content
03/18/2009	1.02	Editorial	Revised and edited the technical content
07/13/2009	1.03	Major	Changes made for template compliance
08/28/2009	1.04	Editorial	Revised and edited the technical content
11/06/2009	1.05	Editorial	Revised and edited the technical content
02/19/2010	2.0	Minor	Updated the technical content
03/31/2010	2.01	Editorial	Revised and edited the technical content
04/30/2010	2.02	Editorial	Revised and edited the technical content
06/07/2010	2.03	Editorial	Revised and edited the technical content
06/29/2010	2.04	Editorial	Changed language and formatting in the technical content.
07/23/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
09/27/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
11/15/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
12/17/2010	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
03/18/2011	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
06/10/2011	2.04	No change	No changes to the meaning, language, or formatting of the technical content.
01/20/2012	2.5	Minor	Clarified the meaning of the technical content.
04/11/2012	2.5	No change	No changes to the meaning, language, or formatting of the technical content.
07/16/2012	2.5	No change	No changes to the meaning, language, or formatting of the technical content.

Table of Contents

1 Introduction

This document specifies the Windows® SharePoint® Services: Content Database Core List Schema and Site Provisioning Communications protocol that allows Web and application servers to perform data query and update commands on database servers.

Sections 1.8, 2, and 3 of this specification are normative and can contain the terms MAY, SHOULD, MUST, MUST NOT, and SHOULD NOT as defined in RFC 2119. Sections 1.5 and 1.9 are also normative but cannot contain those terms. All other sections and examples in this specification are informative.

1.1 Glossary

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

access control list (ACL)
anonymous user
Coordinated Universal Time (UTC)
GUID
language code identifier (LCID)
XML

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

activation
audit flag
back-end database server
Central Administration site
collation
collation order
column
content database
content type
content type identifier
content type order
content type resource folder
customized
default form
default list view
default mobile list view
directory name
display form
display name
document
document library
domain group
edit form
event host
event receiver
external security provider
farm
feature
feature identifier
feature scope
field

field identifier
file
folder
form
form digest validation
front-end Web server
gallery
group
HTTP referer
Hypertext Markup Language (HTML)
leaf name
list
list column
list identifier
list item
list schema
list server template
list template
List View Web Part
login name
master page
Meeting Workspace site
metadict
mobile device
new form
page
parent site
permission
permission level
portal site
provision
provisioned
query
resource folder
result set
return code
root folder
row
security scope
server-relative URL
site
site collection
site collection administrator
site collection flag
site collection identifier
site collection quota
site column
site content type
site definition
site definition configuration
site description
site identifier
site template
site title
site-relative URL

stored procedure
store-relative form
store-relative URL
Structured Query Language (SQL)
subsite
SystemID
theme
top-level site
transaction
Transact-Structured Query Language (T-SQL)
uncustomized
Uniform Resource Locator (URL)
usage data
user activity status
user identifier
user-agent string
version
view
view form
view identifier
Web Part
Web Part Page
Web Part zone
Windows collation name
workflow
workflow association
XML document
XML fragment
XML schema
XML schema definition (XSD)
zero-based index

The following terms are specific to this document:

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

1.2 References

References to Microsoft Open Specifications documentation do not include a publishing year because links are to the latest version of the technical documents, which are updated frequently. References to other documents include a publishing year when one is available.

1.2.1 Normative References

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

[Iseminger] Microsoft Corporation, "SQL Server 2000 Architecture and XML/Internet Support", Volume 1 of Microsoft SQL Server 2000 Reference Library, Microsoft Press, 2001, ISBN 0-7356-1280-3, http://www.microsoft.com/mspress/books/5001.aspx

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

[MS-FPSE] Microsoft Corporation, "FrontPage Server Extensions Remote Protocol Specification".

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

[MS-WSSCAML] Microsoft Corporation, "Collaborative Application Markup Language (CAML) Structure Specification".

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

[MS-WSSFO2] Microsoft Corporation, "Windows SharePoint Services (WSS): File Operations Database Communications Version 2 Protocol Specification".

[MS-WSSPROG] Microsoft Corporation, "Windows SharePoint Services: Content Database Programmability Extensions Communications Protocol Specification".

[MS-WSSTS] Microsoft Corporation, "Windows SharePoint Services Technical 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 list schema management and site provisioning within Windows® SharePoint® Services. This client-to-server protocol uses the Tabular Data Stream Protocol defined in [MS-TDS] as its transport between the front-end Web server, acting as a protocol client, and the back-end database server, acting as a protocol server.

1.3.1 Content Types

1.3.1.1 List Content Type Overview

List (1) content types are objects defined at list (1) level that specify **list item** behaviors in the list (1). A list (1) can contain multiple list (1) content types, which allows the list (1) to contain list items with different behaviors. Every list item on a list (1) is assigned a content type. The content type specifies which **list columns** are applicable to the list item. The content type can also specify the appearance of the **view forms**, **edit forms**, and **new forms** for the list item. The content type can also specify list item **event receivers** associated on the list item. The content type can also specify **workflows (2)** associated on the list item. For lists that are **document libraries**, the content type can also define the **document** template to use when creating a new document of the specified content type. Finally, the list content type contains a generic **XML document** collection and **resource folder** through which vendor extensions can be added to enable scenarios related to list items of the content type. A list owner can update/delete list content types on the list. A list owner can also add a list content type to a list by applying a **site content type** to the list. List content types are destroyed when the list on which they are defined is deleted.

1.3.1.2 Site Content Type Overview

Site content types are objects defined at **site (2)** level that can be used to share common list (1) content type definitions across lists and sites. Site content types can be **provisioned** on a site through **feature activation**. A designer can create a new site content type by deriving a child site content type from an existing site content type. The derived child site content type will inherit all the settings of the parent site content type. When applying a site content type to a list, a list content type is derived from the site content type and added to the list. The derivations of site content types define an ancestral relationship of all the site content types. A site designer can update and delete an existing site content type. Updates made to a site content type are propagated to all the derived site content types and list content types. The **back-end database server** stores how site content types are being used by lists in the site hierarchy and can block deletion of site content types when there are still derived list content types. Site content types are destroyed when the site on which they are defined is deleted.

1.3.2 Features

Features provides the ability to include and remove pieces of dynamic functionality in **site collections** and sites (2). Features change the runtime behavior of their underlying **feature scope** in an application defined manner. Features have a **GUID** designating its **feature identifier** that is unique in the **farm**.

The lifetime of a feature is as follows:

1. The feature gets installed.

2. The feature gets marked as active at one or more feature scopes.

3. The feature gets marked as inactive from all feature scopes.

4. The feature gets uninstalled.

The first two stages are required in order for the functionality of the feature to take effect. The last two stages are only required to remove the feature from the farm.

1.3.3 Views

This protocol also specifies communication between the **front-end Web server** and the back-end database server to configure default **views** and default **forms (2)** for lists (1).

1.3.4 List Schema

A list (1) owner can create, update and delete **list columns**. The list owner can add a list column based on a **site column**. The site owner can update and delete list content types. The list owner can apply a site content type to the list.

1.3.4.1 List Column Overview

List columns are objects defined at list (1) level that can be used to store data about list items. List columns that are defined in a **list template** are provisioned on the list when the list is created. After the list is created, the list owner can add new list columns, and update or delete existing list columns. A list owner can define views to select which list columns are visible and how list items are organized (sorting order, filtering, grouping etc) based on their values set on list columns.

1.3.4.2 Site Column Overview

Site columns are objects defined at site (2) level that can be used to share common list column definitions across lists (1). Site columns can be provisioned through feature activation. A site designer can create, delete and update a site column. A list owner can create new list columns based on site columns defined on the containing site or its ancestor sites. A site designer can update an existing site column and optionally push the updated site column definition to all the list columns, including those residing in a **subsite**, that are created based on the site column. The back-end database server keeps track of how site columns are being used by lists in the site (2) hierarchy and can block deletion of a site column when there are still list columns referencing it. Site columns are destroyed when the site (2) on which they are defined is deleted.

1.3.5 List/Web Metainfo

The list (1) metadata specifies how a list will appear and behave. A list owner can change the appearance and behaviors of the list by setting different metadata values for the list.

The site (2) metadata specifies how a site will appear and behave. A **site collection administrator** can change the appearance and behaviors of the site by setting different metadata values for the site.

1.3.6 File Handling

The protocol client can fetch the **Uniform Resource Locator (URL)** of the **parent site** of a given site (2) by calling proc_GetParentWebUrl. This can be used to check whether a given site is the **top-level site** of a site collection.

The protocol client can enumerate all the files in a site including those in the descendent sites by calling proc_ListAllFileUrls.

The protocol client can enumerate all the sites in a site collection and fetch the sites name, URL, **site identifier**, and parent site identifier (if the site is not the top-level site of the site collection) and language by calling proc_ListAllWebsOfSite.

The protocol client can enumerate all direct child sites of a parent site in a site collection and fetch the child sites name, URL, site identifier and language by calling proc_ListChildWebs.

The protocol client can enumerate all direct child sites of a parent site in a site collection that is of a specific **site definition**, or with a specific **site definition configuration** by calling proc_ListChildWebsFiltered.

1.3.7 Provisioning

A list (1) can be provisioned from a list template hosted either on a front-end Web server or saved in the list template **gallery (1)** of the site collection. List columns and list content types on the list that are based on site columns and site content types are synchronized to match the site columns and site content types.

A site can be provisioned from a site definition hosted either on a front-end Web server or saved in the **site template** gallery (1) of the parent site. Site (2) provisioning calls list provisioning to create pre-defined lists for the site (2).

A site collection can be provisioned from a site collection template hosted on a front-end Web server only. Site collection provisioning calls site provisioning to create the top-level site of the site collection.

1.4 Relationship to Other Protocols

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

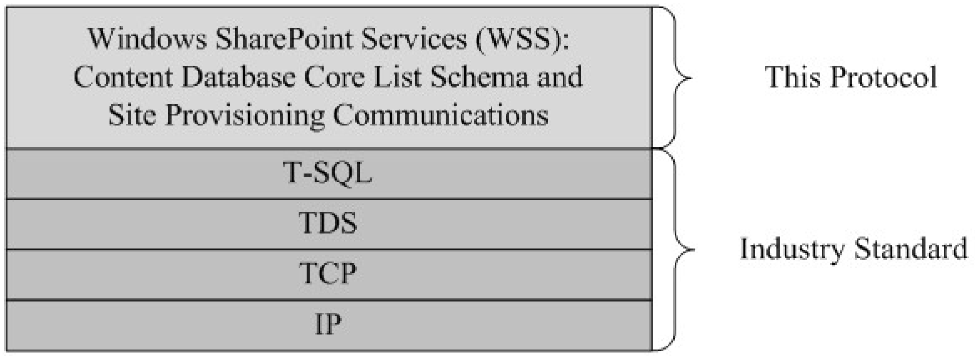
Windows SharePoint Services (WSS): Content Database Core List Schema and Site Provisioning Communications
T-SQL
TDS
TCP
IP
This Protocol
Industry Standard

Figure 1: This protocol in relation to other protocols

1.5 Prerequisites/Preconditions

The operations described by the protocol operate between a client and a back-end database server on which the databases are stored. 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** tables, return result codes, and return **result sets**.

2.2 Common Data Types

This section contains common definitions used by this protocol.

2.2.1 Simple Data Types and Enumerations

2.2.2 Simple Data Types

None.

2.2.3 Bit Fields and Flag Structures

None.

2.2.4 Enumerations

None.

2.2.5 Binary Structures

None.

2.2.5.1 tContentTypeId

tContentTypeId is used to uniquely identify a content type and is designed to be recursive. tContentTypeId encapsulates the lineage of the content type, or the line of parent content types from which the content type inherits. Each tContentTypeId contains the identifier of the parent content type, which in turn contains the identifier of the parent of that content type, and so on, ultimately back to and including the System content type identifier (0x) (For information about content types see [MS-WSSTS] section 2.1.2.8 Content Type.

A tContentTypeId is a numeric string value of arbitrary but limited length, which uniquely identifies a content type, stored on the back-end database server as a varbinary(512).

tContentTypeId MUST follow one of the 2 following valid conventions:

1. Parent tContentType + two hexadecimal values (the two hexadecimal values MUST NOT be "00")

2. Parent tContentType + "00" + hexadecimal GUID

Example 1: Using method 1.

0x01

Example 2: Using method 2 to create a content type whose parent is the content type from Example 1.

0x010077745d60-fb5d-4415-b722-f63181fb6e9d

2.2.5.2 List Identifier Packed Array

A structure that contains the sequential arranged binary representation of one or more **list identifiers**.

0	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	16	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1	32 – n
List Identifier 1																List Identifier 2																

List Identifier 1: The first list identifier in the packed array.

List Identifier 2: The second list identifier in the packed array.

The following example **Transact-Structured Query Language (T-SQL)** code illustrates how to create a List Identifier Packed Array:

```
DECLARE @ GUID1 uniqueidentifier
DECLARE @ GUID2 uniqueidentifier
DECLARE @bin1 binary(16)
DECLARE @bin2 binary(16)
DECLARE @packedarray binary(32)

SET @ GUID1 = '01234567-1234-5678-9012-345678901234'
SET @bin1 = CAST(@GUID1 as binary)
SET @ GUID2 = 'aabbccdd-1234-5678-aaaa-01234bbcdef0'
SET @bin2 = CAST(@GUID2 as binary)

SET @packedarray = @bin1 + @bin2
```

2.2.5.3 List Base Type Pattern

A bit pattern used to indicate which base types an operation will operate on. (For more information about base types, see List Base Type Pattern.)

The pattern is in the following format.

Value	Description
Bit 0	Generic list
Bit 1	Document Library
Bit 2	Unused
Bit 3	Discussion Board
Bit 4	Survey
Bit 5	Issue
Bit 6 – 31	Ignored

2.2.5.4 Usage Data Binary Field Structure

A structure that contains usage data for a site (2). The structure starts with a header that describes the data contained by the field, followed by 5 types of usage data blocks, as shown by the following table:

Usage Data Header (100 bytes)
Page Data (Variable)
User Data (Variable)
Operating System Data (Variable)
Browser Data (Variable)
Referrer Data (Variable)
Reserved (290 bytes)

Usage Data Header (100 bytes): Defined in section 2.2.5.4.1.

Page Data (Variable): A series of Usage Records that specify the pages that have been requested from a site (2). Each Usage Record contains the **site-relative URL** of the **page** that was requested followed by the number of times that it has been requested in each of the last 31 days (for daily **usage data**), or 31 months (for monthly usage data). There MUST NOT be any Usage Records for pages that have not been requested.

User Data (Variable): A series of Usage Records that specify the users that have requested content from a site. Each Usage Records contains the **login name** of a user that requested content followed by the number of requests for each of the last 31 days (for daily usage data), or 31 months (for monthly usage data). There MUST NOT be any Usage Records for users that have not requested content.

Operating System Data (Variable): A series of Usage Records that specify the operating systems that have requested content from a site, as provided in the **user-agent string**. Each Usage Record contains the name of the operating system followed by the number of requests for each of the last 31 days (for daily usage data), or 31 months (for monthly usage data). There MUST NOT be any Usage Records for operating systems that have not requested content.

Browser Data (Variable): A series of Usage Records that specify the browsers that have requested content from a site, as provided in the user-agent string. Each Usage Record contains the name of the browser followed by the number of requests for each of the last 31 days (for daily usage data), or 31 months (for monthly usage data). There MUST NOT be any Usage Records for browsers that have not requested content.

Referrer Data (Variable): A series of Usage Records that specify the **HTTP referer** in requests to content from a site. Each Usage Record contains the address of the HTTP referer**,** followed by the number of times that the address has been present in requests for each of the last 31 days (for daily usage data), or 31 months (for monthly usage data). There MUST NOT be any Usage Records for referrers that did not link to content in the site (2).

Reserved (190 bytes): Reserved. MUST be ignored by reader.

2.2.5.4.1 Usage Data Header Structure

The Usage Data Header describes the information contained by the Usage Data Binary Field structure.

byte1	byte2	byte3	byte4
Size			
Update Counter			
Page Data Offset			
User Data Offset			
Operating System Data Offset			
Browser Data Offset			
Referrer Data Offset			
Reserved 1			
...			
Page Data Count			
User Data Count			
Operating System Data Count			
Browser Data Count			
Referrer Data Count			
Reserved 2			
...			
Last Accessed Day		*Rollover Day*	*Reserved 3*
...			
...			
...			

Size (4 bytes): An unsigned integer that specifies the number of bytes contained by the structure.

Update Counter (4 bytes): An unsigned integer describing the number of times that the structure has been stored.

Page Data Offset (4 bytes): An unsigned integer that counts the number of bytes from the start of the structure to the start of the Page Data.

User Data Offset (4 bytes): An unsigned integer that counts the number of bytes from the start of the structure to the start of the User Data.

Operating System Data Offset (4 bytes): An unsigned integer that counts the number of bytes from the start of the structure to the start of the Operating System Data.

Browser Data Offset (4 bytes): An unsigned integer that counts the number of bytes from the start of the structure to the start of the Browser Data.

Referrer Data Offset (4 bytes): An unsigned integer that counts the number of bytes from the start of the structure to the start of the Browser Data.

Reserved 1 (8 bytes): MUST be ignored by reader.

Page Data Count (4 bytes): An unsigned integer that counts the number of entries of Page Data.

User Data Count (4 bytes): An unsigned integer that counts the number of entries of User Data.

Operating System Data Count (4 bytes): An unsigned integer that counts the number of entries of Operating System Data.

Browser Data Count (4 bytes): An unsigned integer that counts the number of entries of Browser Data.

Referrer Data Count (4 bytes): An unsigned integer that counts the number of entries of Referrer Data.

Reserved 2 (8 bytes): MUST be ignored by reader.

Last Accessed Day (2 bytes): An unsigned integer that contains the number of days since 1/1/1899 to the day that the structure was last stored.

Rollover Day (1 byte): An unsigned integer that specifies the rollover day of usage data from daily data into monthly data. The value MUST be between 1 and 27 (inclusive).

Reserved 3 (33 bytes): MUST be ignored by reader.

2.2.5.4.2 Usage Record Structure

Each of the usage data blocks consists of a series of Usage Records. The first Usage Record in each usage data block contains summary information for the usage data block. Individual usage entries then follow, each in its own Usage Record.

The Usage Record Structure consists of a Description field and a Data field.

Record Description (variable)	***Record Data (variable)***

Record Description (variable): A NULL terminated UTF8 encoded string. It MUST be NULL if this is the first Usage Record. For any other records, it contains the string representation of the usage data being recorded.

Record Data (variable): The usage data for the record is organized as shown in the following table:

byte1	**byte2**								**byte3**	**byte4**
Bytes	*R*	*R*	*R*	*R*	*R*	*R*	*B*	*A*	*Last Accessed*	
Hit Vector										
Total										
Hit Values (variable)										

Bytes (1 byte): An unsigned integer specifying the number of bytes contained by the Record Data structure.

A (1 bit): MUST be set to 1 if the size of values in the Hit Values field is 2 bytes per value. MUST be set to 0 for all other cases.

B (1 bit): MUST be set to 1 if the size of values in the Hit Values field is 4 bytes per value. MUST be set to 0 for all other cases.

R (1 bit): MUST be set to 0.

Last Accessed (2 bytes): An unsigned integer that contains the number of days since 1/1/1899 to the day that the record was last stored.

Hit Vector (4 bytes): A 32-bit value that specifies the number of values in the Hit Values field. The high bit corresponds to the value for the Last Accessed field. The following bits to the previous 31 days (for daily usage data) or previous 31 months (for monthly usage data).

Total (4 bytes): An unsigned integer that contains the sum of the values in the Hit Values field.

Hit Values (variable): A series of 32 unsigned integers that specify a count per day (for daily usage data) or per month (for monthly usage data) for the usage data being described by this record. The size of each value MUST be 1 byte if the A and B flags are set to 0. The size MUST be 2 bytes if the A flag is set to 1 and the B flag is set to 0. The size MUST be 4 bytes if the A flag is set to 0 and the B flag is set to 1. The number of values in this field MUST be equal to the number of bits set to 1 in the Hit Vector field.

2.2.6 Common Result Sets

None.

2.2.7 SQL Structures

None.

2.2.8 Tables and Views

None.

2.2.9 XML Structures

2.2.9.1 Namespaces

None.

2.2.9.2 Simple Types

None.

2.2.9.3 Complex Types

2.2.9.3.1 Feature Property Definitions

The following **XML schema definition (XSD)** defines the Feature Property Definitions:

```
<xs:element name="Properties" type="FeaturePropertyDefinitions" minOccurs="0" maxOccurs="1"
/>
<xs:complexType name="FeaturePropertyDefinitions">
    <xs:sequence>
      <xs:element name="Property" type="FeaturePropertyDefinition" minOccurs="0"
maxOccurs="unbounded" />
    </xs:sequence>
</xs:complexType>
<xs:complexType name="FeaturePropertyDefinition">
    <xs:attribute name="Key" type="xs:string" />
    <xs:attribute name="Value" type="xs:string" />
</xs:complexType>
```

Example:

```
<Properties>
    <Property Name="Color" Value="Red" />
    <Property Name="HatSize" Value="13" />
</Properties>
```

2.2.9.4 Elements

None.

2.2.9.5 Attributes

None.

2.2.9.6 Groups

None.

2.2.9.7 Attribute Groups

None.

3 Protocol Details

3.1 Back-End Database Server Details

3.1.1 Abstract Data Model

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

3.1.1.1 Content Types

3.1.1.1.1 List Content Type Data Model

To apply a site content type to a list (1):

1. Check if the site content type has already been applied to the list by calling proc_GetListContentTypes, if so terminate with error.

2. Derive (see Site Content type data model) a child content type of the site content type. The derived content type will be added to the list's content type collection.

3. For each site column referenced by the site content type, check if the corresponding list column exists on the list by calling **proc_GetListFields** (defined in [MS-WSSFO2]), if not, add the corresponding list column to the list by calling proc_UpdateListFields and create usage tracking records from the site columns to the content type be calling proc_UnmapFieldsFromContentType and proc_MapFieldToContentType.

4. Copy the resource folder and its content of the site content type to the resource folder of the derived list content type in the list by calling **proc_CopyUrl** (defined in [MS-WSSDLIM] section 3.1.4.13).

5. Set up list item **workflow associations** specified by the site content type on the list by calling **proc_AddWorkflowAssociation** (defined in [MS-WSSPROG] section 3.1.4.5) or **proc_UpdateWorkflowAssociation** (defined in [MS-WSSPROG] section 3.1.4.69).

6. Add the derived content type to the list's content type collection by calling proc_UpdateListContentTypes.

7. Record a usage tracking entry for the site content type to the list by calling proc_MapContentTypeToList.

8. Set up list item event receivers specified by the site content type on the list by calling **proc_InsertEventReceiver** (defined in [MS-WSSPROG] section 3.1.4.51).

To update a list content type:

1. Update workflow associations specified by the list content type by calling **proc_AddWorkflowAssociation** (defined in [MS-WSSPROG] section 3.1.4.5), **proc_DropWorkflowAssociation** (defined in [MS-WSSPROG] section 3.1.4.29) or **proc_UpdateWorkflowAssociation** (defined in [MS-WSSPROG] section 3.1.4.69).

2. For each site column referenced by the updated list content type, check if the corresponding list column exists on the list by calling **proc_GetListFields** (defined in [MS-WSSFO2]), if not, add the corresponding list column to the list by calling proc_UpdateListFields and record a usage tracking entry for the site column to the list by calling proc_MapContentTypeToList.

3. Update the list content type schema according to the change by calling proc_UpdateListContentTypes.

4. Update list item event receivers specified by the list content type by calling **proc_DeleteEventReceiversBySourceId** (defined in [MS-WSSPROG] section 3.1.4.20) and **proc_InsertEventReceiver** (defined in [MS-WSSPROG] section 3.1.4.51).

5. If the **display name** of the content type has changed, synchronize the list item's content type **column** value by calling proc_RenameListItemContentType.

To delete a list content type:

1. Check if this is the last list content type on the list by calling proc_GetListContentTypes, if so terminate with error.

2. Check if there are list items whose content type value is set to the list content type by calling proc_CountContentTypeInUseInList. If so, terminate with error.

3. Update **list schema** to remove the list content type from the list's content type collection by calling proc_UpdateListContentTypes and proc_UpdateListFields.

4. Remove the usage tracking entry for the site content type to the list by calling proc_UnmapContentTypeFromList.

5. Remove list item event receivers specified by the list content type by calling **proc_DeleteEventReceiversBySourceId** (defined in [MS-WSSPROG] section 3.1.4.20).

3.1.1.1.2 Site Content Type Data Model

To derive a child site content type from an existing site content type:

1. Fetch the parent content type by calling proc_ListContentTypesInScope.

2. Construct a new **content type identifier** based on the parent site content type's identifier and its next child byte value.

3. Create a blank content type object with the new child content type identifier.

4. Copy the schema of the parent content type to the child content type.

5. Override the child content type's display name and identifier to the values specified by the request.

6. Create the resource folder for the child content type using the child content type's display name by calling **proc_CreateDir** (defined in [MS-WSSFO2] section 3.1.5.8). Copy the content of the parent content type's resource folder to the child content type's resource folder by calling **proc_CopyUrl** (defined in [MS-WSSDLIM] section 3.1.4.13).

7. Create usage tracking records from the site columns to the content type by calling proc_UnmapFieldsFromContentType and proc_MapFieldToContentType.

8. Add the child content type to the content type collection of the site (2) by calling proc_AddContentTypeToScope.

To update a site content type:

1. Update the site content type schema by calling proc_UpdateContentTypeInScope.
2. Remove all usage tracking entries from site columns to the site content type by calling proc_UnmapFieldsFromContentType.
3. For each site column referenced by the updated site content type, add back a usage tracking entry from the site column to the site content type by calling proc_MapFieldToContentType.
4. Find all derived site content types by calling proc_ListDerivedContentTypes.
5. For each site content type returned from step 4, repeat step 1 to 3.
6. For each list (1) content type returned from step 4, update it using list content type data model (see previous procedure).

To delete a site content type, call proc_DeleteContentTypeInScope. This will:

1. Check if the site content type has derived site content types or derived list content types. If so, terminate with error.
2. Check if the content type is provisioned as part of a feature. If so terminate with error.
3. Remove the usage tracking entries recorded from site columns to the site content type.
4. Delete the resource folder of the site content type.
5. Remove the site content type from the site's content type collection.

3.1.1.2 Features

Feature state is changed by the front-end Web server using the following five stored procedures:

1. proc_ActivateFeature
2. proc_DeactivateFeature
3. proc_GetFeatureProperties
4. proc_UpdateFeatureProperties
5. proc_GetWebFeatureList

3.1.1.3 Views

The back-end database server maintains the following sets of data for this protocol within a **content database**.

- **Field (2)**: A data type definition.
- List item: A data unit that stores information for a custom set of fields (2).
- List (1): A collection of list items with associated views.
- **Web Part Page**: A type of Web page that displays **Web Parts** inside **Web Part zones**.
- Web Part zone: A container for Web Parts.

- Web Part: A programmable control that displays information in a Web page.
- Form **control**: A programmable control that creates, updates, or displays items and the fields (2) that they contain.
- **Form page**: A Web Part Page that displays a form control.
- **Default form**: A setting that determines to which URL clients are redirected based on whether they are creating, updating, or displaying list items.
- **List View Web Part**: A type of Web Part that displays formatted list items from a list.
- **View**: A type of Web Part Page that contains a List View Web Part.
- **Default list view**: A setting that determines which view to automatically present to clients.
- **Default mobile list view**: A setting that determines which view to automatically present to mobile clients.

3.1.1.4 List Schema

3.1.1.4.1 List Column Data Model

To add a new list column to a list (1), call proc_UpdateListFields.

To update an existing list column on a list:

1. If the list column data type will be changed as the result of change, convert the list column data to the new data type for all list items in the list.
2. Call proc_UpdateListFields to change the list column definition.

To delete an existing list column on a list:

1. For each view that references the list column , call **proc_UpdateView** (defined in [MS-WSSDLIM] section 3.1.4.65) to remove the reference from the view.
2. Remove the list column from the list by calling proc_DropListField. This will also set the list column value to empty for list items in the list.
3. Remove the usage tracking record from the site column to the list by calling proc_UnmapFieldFromList.
4. Delete the list column from the list definition by calling proc_DropListField.

3.1.1.4.2 Site Column Data Model

To add a site column:

1. Check the identifier and internal name of the new column (1) for duplicate entries already on the site (2) by calling proc_ListContentTypesInScope. If there are duplicates, terminate with error.
2. Add the new site column to the site's column collection by calling proc_AddContentTypeToScope.

To update a site column:

1. Update the column schema of the site column by calling proc_UpdateContentTypeInScope.

2. Find all usage tracking entries from lists (1) to the site column by calling proc_ListsUsingFieldTemplate. For each list that has the site column in use, update the corresponding list column to match the new site column definition by calling proc_UpdateListFields.

To delete a site column:

1. Check if there is a usage tracking entry from a site content type to the site column by calling proc_ListContentTypesInScope. If so, fail with error.
2. Delete the site column from the site's column collection by calling proc_DeleteFieldTemplateInScope.

3.1.1.5 Provisioning

To **provision** a list (1) on a site (2):

1. Create a new list by calling **proc_CreateList** (defined in [MS-WSSDLIM] section 3.1.4.14).
2. Create a usage tracking record from site content type to list for each site content type referenced in the list by calling proc_MapContentTypeToList.
3. Copy the resource folders of each site content type referenced by the list to the corresponding list content type' resource folder by calling proc_CopyResourceDir.
4. Create a usage tracking record from site column to list for each site column referenced by the list columns.
5. Create list views defined in the list template by calling **proc_CreateView** (defined in [MS-WSSDLIM] section 3.1.4.16)..
6. Provision list view pages and form pages by calling **proc_AddGhostDocument** (defined in [MS-WSSDLIM] section 3.1.4.1).
7. Check if the site columns referenced by the list have been modified by calling proc_ListUnghostedFieldTemplatesInList and proc_GetUnghostedBaseFieldTempalteInSite. If so, call proc_UpdateListFields to update the list columns to match the site column definition.
8. Check if the site content types referenced in the list have been modified by calling proc_IsContentTypeGhosted. If so, update the corresponding list content type to match the site content type.
9. For all features activated on the site (including those activated at levels above the site), check if there is an event receiver feature element defined for the base type of the list. (For more information about base types, see List Base Type Pattern.) If so, call proc_ InsertEventReceiver to set up the specified event receiver on the list.

For each list content type defined on the list, check if there are event receivers associated with the content type. If so, call **proc_InsertEventReceiver** (defined in [MS-WSSPROG] section 3.1.4.51) to set up the associated event receiver on the list.

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

The T-SQL syntax for each stored procedure and result set, and the variables they are composed of, is defined in the [MSDN-TSQL-Ref] protocol. In the T-SQL syntax, the variable name is followed by the type of the variable which can optionally have a length value in brackets and can optionally have a default value indicated by an equals sign followed by the default value. Unless otherwise specified, all stored procedures defined in this section are located in the content database.

For definitional clarity, a name has been assigned to any columns in the result sets that do not have a defined name in their current implementation. This does not affect the operation of the result set, as the ordinal position of any column with no defined name is expected by the front-end Web server. Such names are designated in the text using curly braces in the form {*name*}.

3.1.4.1 proc_ActivateFeature

The **proc_ActivateFeature** stored procedure is called to mark a feature active in a site (2) or site collection. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ActivateFeature(
  @SiteId         uniqueidentifier,
  @WebId          uniqueidentifier,
  @FeatureId      uniqueidentifier,
  @Properties     ntext = NULL
);
```

@SiteId: The **site collection identifier** of the site collection in which the feature will be marked active.

@WebId: MUST be a site identifier containing all zeros if the feature is site collection scoped. Otherwise, this parameter is the site identifier of the site in which the feature will be marked active.

@FeatureId: The feature identifier of the feature to be marked active. This parameter MUST NOT be NULL.

@Properties: An **XML fragment**, that MUST conform to the XSD defined in Feature Property Definitions, for the feature. If the *@Properties* parameter is NULL, the Feature Property Definitions MUST be empty.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
3	The site collection or site does not exist.
80	The feature is already marked active in the site or site collection.
1168	Failed to mark feature as active due to an internal error.

Result Sets: MUST NOT return any result sets.

3.1.4.2 proc_AddContentTypeToScope

The **proc_AddContentTypeToScope** stored procedure is called to add a site content type or site column to a given site (2). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_AddContentTypeToScope(
  @SiteId                uniqueidentifier,
  @Class                 tinyint,
  @ContentTypeId         varbinary(512),
  @Scope                 nvarchar(256),
  @Definition            ntext,
  @ParentContentTypeId   varbinary(512) = NULL,
  @ParentScopeIn         nvarchar(256) = NULL,
  @ResourceDir           nvarchar(128) = NULL
);
```

@SiteId: The site collection identifier of the site collection that contains the requested site (2).

@Class: The type of record that should be created. The parameter MUST be in the following table:

Value	Description
0	Site column
1	Site content type

@ContentTypeId: The content type identifier of the site content type or site column to be added. This MUST be of type tContentTypeId and MUST NOT be NULL.

@Scope: The **store-relative form** URL of the site to which the site content type or site is added.

@Definition: The XML fragment that defines the site content type or site column. The **XML schemas** for these structures are defined in the TPContentTypeReferences and TPFields sections of [MS-WSSCAML] respectively.

@ParentContentTypeId: If *@Class* is equal to zero, this MUST be NULL. If *@Class* is equal to "1", this MUST be the identifier of the ancestor site content type or NULL to imply that there is no parent site content type. This MUST be of type tContentTypeId.

@ParentScopeIn: If *@ParentContentTypeId* is NULL, then this MUST be NULL. Otherwise, this MUST be the store-relative form URL to which the parent site content type is registered.

@ResourceDir: The **leaf name** of the site content type's resource folder.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
80	The site content type or site column was not created.
85	The site content type or site column is already registered to the site designated by *@SiteId* and *@Scope* or an ancestor of that site.
144	*@Scope* refers to a site that is not within the site collection designated by the *@SiteId* parameter.

Value	Description
212	The site collection is locked.
1816	The **site collection quota** for the site collection has been exceeded.

Result Sets: MUST NOT return any result sets.

3.1.4.3 proc_CopyResourceDir

The **proc_CopyResourceDir** stored procedure is called to copy a **content type resource folder** and all **folders** and documents subsumed by it. All folders down to the *@TargetDir* will be created if they do not already exist. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_CopyResourceDir(
  @SiteId             uniqueidentifier,
  @WebId              uniqueidentifier,
  @ContentTypeId      varbinary(512),
  @Scope              nvarchar(256),
  @TargetDir          nvarchar(256)
);
```

@SiteId: The site collection identifier of the site collection that the content type resource folder to be copied resides in.

@WebId: The site identifier of the site (2) that the content type resource folder to be copied resides in.

@ContentTypeId: The content type identifier of the content type whose content type resource folder is to be copied. This MUST be of type tContentTypeId.

@Scope: The store-relative form URL of the site that contains the content type whose content type resource folder is to be copied.

@TargetDir: The store-relative form URL of the folder where the specified content type resource folder is to be copied. This folder is created as part of the copy operation. This MUST NOT be NULL.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.4 proc_CountContentTypeInUseInList

The **proc_CountContentTypeInUseInList** stored procedure is called to get the count of list items with the specified list (1) content type in the specified list. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_CountContentTypeInUseInList (
  @SiteId            uniqueidentifier,
  @ListId            uniqueidentifier,
  @ContentTypeId     varbinary(512)
);
```

@SiteId: The site collection identifier for the site collection that contains the list (1) specified by *@ListId*.

@ListId: The list identifier for the list in which to count list items with the list content type specified in *@ContentTypeId*.

@ContentTypeId: The content type identifier of the list content type of list items to be counted. This MUST be of type tContentTypeId.

Return Code Values: The stored procedure MUST return an integer which is the count of list items with the specified list content type in the specified list. This count MUST NOT include list items that have been deleted and MUST NOT include list items versions that are not the current version. If *@SiteId* is NULL or *@ListId* is NULL or *@ContentTypeId* is NULL, then the stored procedure MUST return 0.

Result Sets: MUST NOT return any result sets.

3.1.4.5 proc_CountFieldTemplateInstancesInContentTypeTemplate

The **proc_CountFieldTemplateInstancesInContentTypeTemplate** stored procedure is called to get the count of content types in the specified site collection that use the specified site column. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_CountFieldTemplateInstancesInContentTypeTemplate (
  @SiteId     uniqueidentifier,
  @FieldId    uniqueidentifier
);
```

@SiteId: The site collection identifier for the site collection that contains the content types to be counted.

@FieldId: The GUID for the site column used by the content types to be counted.

Return Code Values: The stored procedure MUST return an integer which is the count of content types in the specified site collection that use the specified site column. If *@SiteId* is NULL or *@FieldId* is NULL, the stored procedure MUST return 0.

Result Sets: MUST NOT return any result sets.

3.1.4.6 proc_DeactivateContentTypeInScope

The **proc_DeactivateContentTypeInScope** stored procedure is called to deactivate a site content type or site column in a specific site (2). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_DeactivateContentTypeInScope(
  @SiteId                  uniqueidentifier,
  @WebId                   uniqueidentifier,
  @UserId                  int,
  @Class                   tinyint,
  @Scope                   nvarchar(256),
  @ContentTypeId           varbinary(512),
  @IsDeactivatingFeature   tinyint = 0
);
```

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

@WebId: The site identifier of the site to which the site content type or site column is registered. If this parameter is NULL, then the content type resource folder for the requested site content type or site column SHOULD not be deleted.

@UserId: The **user identifier** of the user who is initiating this procedure.

@Class: The type of record to be deactivated. The parameter MUST be in the following table:

Value	Description
0	Site columns.
1	Site content types.

@Scope: The store-relative form URL of the site to deactivate the site content type or site column from.

@ContentTypeId: Contains the content type identifier of the site content type or site column being requested**.** This MUST be of type tContentTypeId.

@IsDeactivatingFeature: The parameter MUST be in the following table:

Value	Description
0	Do not deactivate specified site content type or site column if it is in use or is part of a feature.
1	Do not deactivate specified site content type or site column if it is in use.
2	Force deactivation even if the site content type or site column is in use or is read-only.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
2	The site content type or site column is not activated on this site.
4307	The site content type or site column is in use.
6009	The site content type or site column is part of a feature and, therefore, is read-only.

Result Sets: MUST return 0, 1 or 2 of the following result sets:

3.1.4.6.1 Is From Feature Result Set

Is From Feature contains a bit which defines if the site content type or site column is part of a feature. The Is From Feature result set MUST only be returned if *@IsDeactivatingFeature* is 0. If the requested site content type or site column is from a feature, then the Is From Feature result set MUST return 1 **row (1)** corresponding to the site content type or site column designated by the *@SiteId*, *@Class*, *@Scope*, and *@ContentTypeId* parameters. If the requested site content type or site column is not from a feature, then the Is From Feature result set MUST return an empty set. The Is From Feature result set is defined using T-SQL syntax, as follows:

```
IsFromFeature    bit;
```

IsFromFeature: Contains a bit which indicates whether the requested site content type or site column is from a feature. The value of this column MUST be 1 in all rows returned in this result set.

3.1.4.6.2 Deleted URLs Result Set

Deleted URLs contains a list of URLs deleted as a result of the action taken by the **proc_DeactivateContentTypeInScope** stored procedure. The Deleted URLs result set MUST only be returned if *@WebId* is not NULL and the site content type or site column designated by *@SiteId*, *@Class*, *@Scope*, and *@ContentTypeId* exists. The Deleted URLs result set is defined using T-SQL syntax, as follows:

```
{Url}          nvarchar(385),
Type           tinyint;
```

{Url}: Contains the **directory name** of the site content type resource folder concatenated with a forward slash ("/") concatenated with the leaf name of the content type resource folder of the site content type that was requested.

Type: The type of **file** located at {Url}.

3.1.4.7 proc_DeactivateFeature

The **proc_DeactivateFeature** stored procedure is called to mark a feature inactive in a site (2) or site collection. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_DeactivateFeature (
  @SiteId         uniqueidentifier,
  @WebId          uniqueidentifier,
  @FeatureId      uniqueidentifier
);
```

@SiteId: The site collection identifier of the site collection in which the feature will be marked inactive.

@WebId: MUST be a site identifier containing all zeros if the feature is site collection scoped. Otherwise, this parameter is the site identifier of the site in which the feature will be marked inactive.

@FeatureId: The feature identifier of the feature to be marked inactive.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
3	The site collection or site does not exist, or the feature is not currently marked active in the site collection or site.

Result Sets: MUST NOT return any result sets.

3.1.4.8 proc_DeleteContentTypeInScope

The **proc_DeleteContentTypeInScope** stored procedure is called to delete a site content type or site column from a specific site (2). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_DeleteContentTypeInScope(
  @SiteId                    uniqueidentifier,
  @Class                     tinyint,
  @Scope                     nvarchar(256),
  @ContentTypeId             varbinary(512),
  @IsDeactivatingFeature     tinyint = 0
);
```

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

@Class: The type of record that should be deactivated. The parameter MUST be in the following table:

Value	Description
0	Site columns.
1	Site content types.

@Scope: The store-relative form URL of the site to deactivate the site content type or site column in.

@ContentTypeId: The identifier of the specific site content type being requested. This MUST be of type tContentTypeId.

@IsDeactivatingFeature: The parameter MUST be in the following table:

Value	Description
0	Do not deactivate specified site content type or site column if it is in use, or is part of a feature.
1	Do not deactivate specified site content type or site column if it is in use.
2	Force deactivation even if the site content type or site column is in use or is read-only.

The only function of this stored procedure is to call another stored procedure, proc_DeactivateContentTypeInScope with the parameters listed in the following table:

Parameter	Passed in Value
@SiteId	*@SiteId*
@WebId	NULL
@UserId	0
@Class	*@Class*
@Scope	*@Scope*
@ContentTypeId	*@ContentTypeId*

Parameter	Passed in Value
@IsDeactivatingFeature	*@IsDeactivatingFeature*

Return Code Values: It MUST return the same **return code** returned by proc_DeactivateContentTypeInScope

Result Sets: It MUST return the same result sets returned by proc_DeactivateContentTypeInScope

3.1.4.9 proc_DeleteFieldTemplateInScope

The **proc_DeleteFieldTemplateInScope** stored procedure is called to delete a site column. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_DeleteFieldTemplateInScope(
  @SiteId        uniqueidentifier,
  @Scope         nvarchar(256),
  @FieldId       uniqueidentifier,
  @BaseTypes     int
);
```

@SiteId: The site collection identifier of the site collection that has the site column to be deleted.

@Scope: The store-relative form URL of the site from which to delete the site column.

@FieldId: The GUID of the site column to be deleted.

@BaseTypes: A bit pattern indicating which base types use the site column being deleted. This MUST include all the base types that use the site column. The bit pattern is described in the List Base Type Pattern section. (For more information about base types and list base type patterns, see List Base Type Pattern.)

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
2	The site collection, scope or field (2) specified by the parameters *@SiteId*, *@Scope*, *@FieldId* respectively does not exist.
4307	The site column is being used in a content type or if there exists any list (1) of type specified by *@BaseTypes* and hence cannot be deleted.

Result Sets: MUST NOT return any result sets.

3.1.4.10 proc_DropListField

The **proc_DropListField** stored procedure is called to delete a field (2) from a list (1). It does not update the views. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_DropListField(
  @SiteId          uniqueidentifier,
  @WebId           uniqueidentifier,
  @ListId          uniqueidentifier,
  @FieldId          uniqueidentifier,
  @ColName          nvarchar(64),
  @RowOrdinal       int,
  @ColName2         nvarchar(64),
  @RowOrdinal2      int,
  @Fields           ntext,
  @ContentTypes     ntext,
  @Version          int
);
```

@SiteId: The site collection identifier in which the list (1) exists.

@WebId: The site identifier in which the list exists.

@ListId: The list identifier.

@FieldId: The **field identifier** to be deleted.

@ColName: The name of the column in the [MS-WSSFO2] section 2.2.5.3, that contains the data for the field (2) being deleted. This value MUST be a valid column name in the [MS-WSSFO2] section 2.2.5.3, or MUST be NULL.

@RowOrdinal: The 0-based ordinal of the row among the set of rows representing a list item of this list, which contains the column representing the field (2) to be deleted.

@ColName2: The column name of the additional column in the [MS-WSSFO2] section 2.2.5.3, that contains data for this field (2) if the field (2) requires two columns to store data. This MUST be a valid column name in the [MS-WSSFO2] section 2.2.5.3 - AllUserData Table, or MUST be NULL.

@RowOrdinal2: The 0-based ordinal of the row among a set of rows representing a list item for this list, which contains the column representing the field (2) to be deleted.

@Fields: The implementation-specific **version** number followed by an XML fragment of the field (2) definitions with the definition of the field (2) being deleted removed. The XML schema for this structure is defined in [MS-WSSCAML] section TPFields.

@ContentTypes: An XML fragment specifying the updated content types registered for this list with the field (2) being deleted removed. The XML schema for this structure is defined in [MS-WSSCAML] section TPContentTypeReferences.

@Version: The version of the list's metadata.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
3	*@SiteId*, *@WebId* or *@ListId* do not specify a valid site collection identifier, site identifier or list identifier.
1150	Version conflict, the version passed in as *@Version* does not match the list's metadata version

Result Sets: MUST NOT return any result sets.

3.1.4.11 proc_EnumListsWithMetadata

The **proc_EnumListsWithMetadata** stored procedure **proc_EnumListsWithMetadata** is called to return the metadata for a set of lists (1). The set of lists is determined based on the input parameters. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_EnumListsWithMetadata(
  @SiteId                uniqueidentifier,
  @WebId                 uniqueidentifier,
  @WebsFilter            int,
  @ListsFilter           int,
  @BaseType              int,
  @ServerTemplate        int,
  @IncludeHidden         bit,
  @FieldId               uniqueidentifier,
  @FieldValue            nvarchar(255),
  @FieldType             int,
  @PrefetchListScopes    bit,
  @MaxLists              int,
  @NumLists              int,
  @ListIds               image
);
```

@SiteId: A site collection identifier.

@WebId: The site identifier of a subsite.

@WebsFilter: Contains a value that specifies the **Web filter** type. It MUST have one of the values listed in the following table:

Value	Description
"0"	The metadata is returned only for lists in the subsite specified by *@WebId*.
"1"	The metadata is returned for lists that belong to the subsite specified by *@WebId* and all of its child subsites.
""2	The metadata is returned for all lists that belong to the site collection given by *@SiteId*.

@ListsFilter: Indicates which lists will have metadata returned. It MUST have one of the values listed in the following table:

Value	Description
"0"	Metadata is only returned for lists that match the specified *@BaseType* and *@ServerTemplate*.
"1"	Metadata is only returned for lists with a match when searching in the field (2) identified by *@FieldId* with the value specified by *@FieldValue*, or in all fields (2) if *@FieldId* is NULL. In addition, the filtering based on *@BaseType* and *@ServerTemplate* is applied.
"2"	Metadata is only returned for the lists (1) that are specified by their identifying list identifier in *@ListIds*.

@BaseType: If *@ListsFilter* is zero ("0") or "1" it MUST be one of the values listed in the following table:

Value	Description
"-1"	Ignored
Other value	The list base type, as specified in [MS-WSSFO2] section 2.2.2.9, for the list (1).

@ServerTemplate: If *@ListsFilter* is zero ("0") or "1", it MUST be one of the values in the following table:

Value	Description
"-1"	Ignored
Other value	The list server template, as specified in [MS-WSSFO2] section 2.2.2.10, used to create the list (1).

@IncludeHidden: If "1", metadata MUST be returned even in the case where the list (1) is hidden. For all other values, metadata MUST NOT be returned for hidden lists.

@FieldId: If *@ListsFilter* is "1", *@FieldId* identifies a field (2) that will be used to search the list (1) based on the value specified in *@FieldValue*. If *@FieldId* is NULL, the search is done based on the *@FieldValue* in all fields (2).

@FieldValue: If *@ListsFilter* is "1", it contains a search value. If *@FieldValue* is NULL, any value is considered a match.

@FieldType: If *@ListsFilter* is "1", it MUST be a value listed in the following table:

Value	Description
"1"	The field (2) used in the search has the type **TEXT**. The search is done based on the subsite **collation order**.
"5"	The field (2) used in the search has the type **CHOICE**. The search is done based on the subsite collation order.
Other value	The search is done ignoring the subsite collation order.

@PrefetchListScopes: If "1", the **List Permissions** result set MUST be returned by this stored procedure. Otherwise the **List Permissions** result set MUST NOT be returned by this stored procedure.

@MaxLists: If it is zero or less, it is ignored. If it is greater than zero, it specifies the maximum number of lists (1) for which metadata should be retrieved. If this stored procedure finds more lists (1) than specified in *@MaxLists,* it returns an error.

@NumLists: Specifies the number of entries in *@ListIds*. Used only when *@ListsFilter* equals "2".

@ListIds: A List Identifier Packed Array of list identifiers. Used only when *@ListsFilter* equals "2"

Return Codes: The stored procedure returns an **integer** return code that MUST be one of the values listed in the following table:

Value	Description
"0"	Successful completion.
"68"	The number of lists (1) that matched the criteria given by the input parameters exceeded the specified *@MaxLists* parameter.

Result Sets: The stored procedure returns 1, 2, 3 or 4 result sets depending on the input parameters and the data.

3.1.4.11.1 List Count Result Set

The **List Count** result set contains only one row with one unnamed column of type **int**. If *@MaxLists* was specified and exceeded, it contains "-1". Otherwise, it contains the number of lists (1) that matched the selection criteria given by the input parameters.

3.1.4.11.2 List Metadata Result Set

The List Metadata result set contains metadata information about each list (1) that matched the selection criteria given by the input parameters.

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

See [MS-WSSFO2] section 2.2.5.12 – List Metadata Result Set, for details.

Note: If the List Metadata result set has no rows then this stored procedure MUST NOT return either of the List Event Receivers result set and List Permissions result set. For the result set of this procedure , the ‘AnonymousePermMask and ‘Acl’ columns in List Metadata result set will not have column names.

3.1.4.11.3 List Event Receivers Result Set

The **List Event Receivers** result set contains information about the event receivers registered for the lists (1) that were returned in the **List Metadata** result set.

There MUST be 1 row in this result set for each event receiver. The result set is ordered by **SiteId**, **WebId**, **HostId**, **Type**, **HostType**, **SequenceNumber**, **Assembly**.

This result set MUST NOT be returned if the **List Metadata** result set was empty.

See [MS-WSSFO2] section 3.1.4.4.4, for details.

3.1.4.11.4 List Permissions Result Set

The **List Permissions** result set identifies **permissions** associated with the lists (1) returned in the **List Metadata** result set.

This result set can contain more than 1 row per list (1). There MUST be as many rows per list (1) as permissions that are associated with that list (1).

This result set MUST NOT be returned if the **List Metadata** result set had no rows or if the input parameter *PrefetchListScopes* was not set to "1".

The **List Permissions** result set is defined using T-SQL syntax, as follows:

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

ListId: The list identifier that identifies the list (1).

ScopeID: The GUID for the **security scope** for this permission.

- If the list (1) inherits permissions from the subsite and has no specific permissions, a row is produced per list (1) and its **ScopeID** MUST be "0x00".
- If the list has its own permission settings, a row is produced per permission and its **ScopeID** indicates the specific **access control list (ACL)** to use for calculating the settings on this list (1).
- If there are list items that have their own permissions, for each list items permission there MUST be 1 row in the result set. In this case, the **ScopeID** indicates the specific **ACL** to use.

Acl: If the list (1) inherits permissions from the subsite, a row is produced per list (1) and **Acl** MUST be NULL.

- If the list has its own permission, a row is produced per permission and **Acl** is the ACL for this list.
- If there are list items that have their own permission, for each permission there is 1 row in the result set. **Acl** is the ACL for the list item.

AnonymousPermMask:

- If the list (1) inherits permission from the Web and has no specific permission, a row is produced per list (1) and **AnonymousPermMask** MUST be "0x00".
- If the list has its own permissions, a row is produced per permission and **AnonymousPermMask** is a **permission level** that indicates the permissions granted to a user that is anonymous, or has no specific rights, on this list.
- If there are list items that have their own permissions, for each list item permission there will be 1 row in the result set. **AnonymousPermMask** is a permission level that indicates the permissions granted to a user that is anonymous, or has no specific rights, on that list item.

3.1.4.12 proc_EnumWebAndSubwebsDTM

The **proc_EnumWebAndSubwebsDTM** stored procedure is called to enumerate date and time properties of the given site and all its subsites. The stored procedure is defined using T-SQL syntax, as follows:

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

@WebID: The site identifier of the site.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion. Result set MUST have 1 or more records.
3	No rows match the given WebID. The result set MUST NOT have any records and MUST be empty.

Result Sets: MUST return the following result set:

3.1.4.12.1 WebsAndSubwebsDTM Result Set

The **WebAndSubwebsDTM** result set returns date and time properties of the site specified in the *@WebID* parameter, and of all its immediate subsites, or child sites.

The **WebAndSubwebsDTM** result set MUST contain 1 row if the site exists, and MUST contain no rows if there is no such Web. The result set MUST contain more than 1 row if there are subsites for the specified *@WebID* parameter.

The **WebAndSubwebsDTM** result set is defined using T-SQL syntax, as follows:

```
Id                          uniqueidentifier,
FullUrl                     nvarchar(257),
LastMetadataChange          datetime,
ListsMaxModified            datetime,
ListsMaxLastSecurityChange  datetime;
```

Id: The site identifier of the site. This is either the **WebID** that was passed in as the parameter *@WebID*, OR the **WebID** of all the subsites, or immediate child sites, of the **WebID** that was passed in as the parameter *@WebID*. The value MUST NOT be NULL.

FullUrl: The store-relative form URL of the site. Each of these URLs MUST have a leading "/". The datatype is a **string** with a maximum length of 257. The value MUST NOT be NULL.

LastMetadataChange: The timestamp in **UTC** format describing the last time this site’s metadata properties were modified by any user. The value MUST NOT be NULL.

ListsMaxModified: The timestamp in UTC format describing the last time when any of the lists (1) contained within the site were modified by any user. The value MUST NOT be NULL.

ListsMaxLastSecurityChange: The timestamp in UTC format describing the last time when any of the security permissions on the lists contained within the site were modified by any user. The value MUST NOT be NULL.

3.1.4.13 proc_EstimateDocsSize

The **proc_EstimateDocsSize** stored procedure is called to provide a rough estimate of the total size of a list (1) or site. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_EstimateDocsSize (
  @ListId             uniqueidentifier,
  @WebId              uniqueidentifier,
  @SiteId             uniqueidentifier,
  @MaxSize            bigint,
  @IncludeListDocs    bit = 1
);
```

@ListId: The list identifier for the list. This parameter MUST be specified and it MUST either identify a list or be NULL.

@WebId: The site identifier for the site. If the *@ListId* parameter is specified, this MUST be the site identifier for the site containing the specified list otherwise, this MUST identify a site in the specified site collection and MUST NOT be NULL.

@SiteId: The site collection identifier for the site collection containing the specified site. This MUST NOT be NULL.

@MaxSize: Specifies a threshold, in bytes, for calculating the estimated size. This protocol client sets this to a size threshold such that, as long as the size is less than *@MaxSize*, the calling protocol client does not care how accurate the estimate is.

@IncludeListDocs: Specifies whether to include documents in the size estimate. This parameter is optional. If specified, the value MUST be one of those listed in the following table:

Value	Description
0	Do not include documents from the specified list, or any list in the site if *@ListId* is NULL, in the size estimate.
1	Include documents from the specified list, or site if *@ListId* is NULL, in the size estimate.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.13.1 EstimatedSize Result Set

EstimatedSize returns a rough estimate of the total size for a list (1) or site. The EstimatedSize result set MUST be returned and MUST contain 1 row. The EstimatedSize result set is defined using T-SQL syntax, as follows:

```
  {Size}    bigint;
```

{Size}: Contains a value representing the estimated total size, in bytes, of the specified list or site. The value MUST be based on the parameters *@ListID* and *@IncludeListDocs* as described in the following table:

@ListID	@IncludeListDocs	{Size}
Defines a list	0	a rough estimate or overestimate of the size of the list excluding documents
Defines a list	1	a rough estimate or overestimate of the size of the list including documents
NULL	0	a rough estimate or overestimate of the size of the site excluding documents
NULL	1	a rough estimate or overestimate of the size of the site including documents that are not part of any list

3.1.4.14 proc_FetchContentTypeInScope

The **proc_FetchContentTypeInScope** stored procedure is called to retrieve information about a specific site content type or site column registered to a specific site. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_FetchContentTypeInScope(
  @SiteId           uniqueidentifier,
  @Class            tinyint,
  @Scope            nvarchar(256),
  @ContentTypeId    varbinary(512)
);
```

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

@Class: The type of record to be retrieved. The parameter MUST be in the following table:

Value	Description
0	Site columns.
1	Site content types.

@Scope: The store-relative form URL of the site to retrieve site content types or site columns from.

@ContentTypeId: The identifier of the specific site content type being requested. This MUST be of type tContentTypeId.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
2	The requested site content type or site column was not found.

Result Sets: MUST return the following result set:

3.1.4.14.1 Content Type Result Set

Returns the definition and version of each site content type or site column that meets the criteria defined in the input parameters. This result set MUST be returned and MUST contain either one row that corresponds to the site content type or site column that meets the criteria defined in the input parameters if such site content type or site column exists, or zero rows if no such site content type or site column exists. The result set is defined using T-SQL syntax, as follows:

```
Definition      ntext,
Version         int;
```

Definition: Contains the XML fragment of the site content type or site column or NULL if the site content type or site column has no XML fragment. The XML schemas for these structures are defined in [MS-WSSCAML] section TPContentTypeReferences, and [MS-WSSCAML] section TPFields, respectively.

Version: The version of the site content type or site column.

3.1.4.15 proc_FixV2ContentTypeField

The **proc_FixV2ContentTypeField** stored procedure is called to set the content type "display name" field (2) for the list items, documents or folders in a particular list (1) in the back-end database server. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_FixV2ContentTypeField(
  @SiteId            uniqueidentifier,
  @ListId            uniqueidentifier,
  @ContentTypeId     varbinary(512),
  @ContentType       nvarchar(255)
);
```

@SiteId: The site collection identifier of the site collection that contains the list where the content type is to be changed.

@ListId: The list identifier of the list that uses the content type whose display name is to be changed.

@ContentTypeId: The content type identifier of the content type whose display name is to be changed. This MUST be of type tContentTypeId.

@ContentType: The new name of the content type whose display name is to be changed. This MUST NOT be NULL.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.16 proc_GetContentTypeIdFromUrl

The **proc_GetContentTypeIdFromUrl** stored procedure is called to retrieve the content type identifier of a document, list item or folder from the back-end database server. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_GetContentTypeIdFromUrl(
  @DocSiteID        uniqueidentifier,
  @DocDirName       nvarchar(256),
  @DocLeafName      nvarchar(128)
);
```

@DocSiteID: The site collection identifier for the site collection that the document**,** list item or folder whose content type identifier is sought, resides in.

@DocDirName: The directory name of the document, list item or folder whose content type identifier is sought.

@DocLeafName: The leaf name of the document**,** list item or folder whose content type identifier is sought.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.16.1 Object Content Type Id Result Set

Returns the content type identifier for the specified document, list item, or folder. This result set MUST contain either:

1. 1 row with 1 column.
2. 0 rows with 1 column

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

```
{ContentTypeId}     varbinary(512)
```

{ContentTypeId}:

Unnamed Column Value	Description
Content Type Id	If the site collection identifier, directory name and leaf name, that are passed in, correspond to a folder, document or list item in the system, then the content type identifier of that folder, document or list item is returned. This MUST be of type tContentTypeId.
0x012001	This is returned if the folder, list item or document specified by the site collection identifier, directory name and leaf name, that are passed in does not have an explicitly assigned content type identifier. One example of this is the **root folder** of a list (1).

If the directory name and leaf name do not correspond to a document, folder or list item, then a result set with an unnamed column and no rows is returned.

3.1.4.17 proc_GetFeatureProperties

The **proc_GetFeatureProperties** stored procedure is called to retrieve the Feature Property Definitions for the feature marked active at a site collection or site. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_GetFeatureProperties(
  @SiteId          uniqueidentifier,
  @WebId           uniqueidentifier,
  @FeatureId       uniqueidentifier
);
```

@SiteId: The site collection identifier of the site collection in which the feature is marked active.

@WebId: MUST be a site identifier containing all zeros if the feature is site collection scoped. Otherwise, this parameter is the site identifier of the site in which the feature is marked active.

@FeatureId: The feature identifier of the feature marked active.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.

Value	Description
3	The target site collection or site does not exist, or the feature is not marked active in the site collection or site.

Result Sets: The stored procedure MUST return one Feature Property Definitions result set when the return code is 0. Otherwise, it MUST NOT return any result sets.

3.1.4.17.1 Feature Properties Result Set

Returns the property set for the feature. This result set, when returned, MUST contain 1 row. The result set is defined using T-SQL syntax, as follows:

```
TimeActivated            datetime,
Flags                    int,
Properties               ntext,
PropertiesModified       datetime;
```

TimeActivated: MUST contain the UTC date and time when the feature was marked activate.

Flags: This MUST be zero.

Properties: MUST be NULL in the case the feature has no properties or MUST contain the XML fragment for the feature properties. The schema of this fragment is defined by Feature Property Definitions.

PropertiesModified: MUST contain the UTC date and time when the Feature Property Definitions for the feature were last modified.

3.1.4.18 proc_GetFolderContentTypeOrder

The **proc_GetFolderContentTypeOrder** stored procedure is called to get the **content type order** of the specified folder. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_GetFolderContentTypeOrder(
  @SiteId              uniqueidentifier,
  @CurrentFolderUrl    nvarchar(260)
);
```

@SiteId: The site collection identifier for the site collection that contains the folder specified by *@CurrentFolderUrl*.

@CurrentFolderUrl: The **server-relative URL** of the folder to get the content type order for.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
87	Invalid Parameters: *@SiteId* is NULL or the site collection specified by *@SiteId* does not exist or *@CurrentFolderUrl* is NULL or the folder specified by *@CurrentFolderUrl* does not exist.

Result Sets: The stored procedure MUST return exactly 1 of 3 possible result sets as follows:

3.1.4.18.1 Invalid Parameters Result Set

This result set is a placeholder that signifies invalid input parameters to the stored procedure. This result set MUST be returned if *@SiteId* is NULL or if the site collection specified by *@SiteId* does not exist or if *@CurrentFolderUrl* is NULL or if the folder specified by *@CurrentFolderUrl* does not exist. This result set MUST contain 1 row. The result set is defined using T-SQL syntax, as follows:

```
{Empty}       nvarchar(260),
{NULL}        varbinary(max);
```

{Empty}: Contains an empty string.

{NULL}: Contains NULL.

3.1.4.18.2 Undefined Content Type Order Result Set

This result set signifies that the specified folder does not have a content type order. This result set MUST be returned only if the folder specified by *@CurrentFolderUrl* does not have a content type order and MUST contain 1 row. The result set is defined using T-SQL syntax, as follows:

```
{CurrentFolderUrl}    nvarchar(260),
{NULL}                varbinary(max);
```

{CurrentFolderUrl}: The Server-relative URL of the folder specified by *@CurrentFolderUrl*.

{NULL}: Contains NULL.

3.1.4.18.3 Defined Content Type Order Result Set

Returns the content type order of the specified folder. This result set MUST be returned only if the folder specified by *@CurrentFolderUrl* has a defined content type order and MUST contain 1 row. The result set is defined using T-SQL syntax, as follows:

```
{CurrentFolderUrl}    nvarchar(260),
MetaInfo              varbinary(max);
```

{CurrentFolderUrl}: The Server-relative URL of the folder specified by *@CurrentFolderUrl*.

MetaInfo: Contains the metadata representation of the document of the folder on which the content type order of the folder specified by *@CurrentFolderUrl* is defined. This document metainfo MUST contain the content type order. (For more information about metainfo, see **metadict**.)

3.1.4.19 proc_GetListContentTypes

The **proc_GetListContentTypes** stored procedure is called to get the list of all content types in the specified list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_GetListContentTypes (
  @WebId       uniqueidentifier,
  @ListId      uniqueidentifier
);
```

@WebId: The site identifier for the site that contains the list specified by *@ListId*.

@ListId: The list identifier for the list in which to get the list of all content types.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.19.1 Content Types Result Set

Returns a list of all content types in the specified list (1). If *@WebId* is not NULL, *@ListId* is not NULL, and the list specified by *@ListId* exists and is not deleted, then this result set MUST contain 1 row. Otherwise, this result set MUST contain 0 rows. The result set is defined using T-SQL syntax, as follows:

```
tp_ContentTypes    nvarchar(max);
```

tp_ContentTypes: Contains an XML fragment representing the content types in the list specified by *@ListId*. The XML schema for this structure is defined in [MS-WSSCAML] section TPContentTypeReferences.

3.1.4.20 proc_GetListIdsToSync

The **proc_GetListIdsToSync** stored procedure is called to enumerate all lists (1) in a specified site that use a **customized (2)** list column or a customized (2) content type. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_GetListIdsToSync (
  @SiteId    uniqueidentifier,
  @WebId     uniqueidentifier,
  @Scope     nvarchar(256)
);
```

@SiteId: The site collection identifier for the site collection that contains the site specified by *@WebId*.

@WebId: The site identifier for the site in which to enumerate lists (1).

@Scope: The store-relative form URL of the site specified by *@WebId*.

Return Code Values: An integer that MUST be zero.

Result Sets: MUST return the **Lists** result set.

3.1.4.20.1 Lists Result Set

The **Lists** result set returns all lists (1) in the specified site that use a customized (2) field (2) or a customized (2) list column. This result set MUST contain 1 row for each distinct list in the site specified by *@WebId* that uses a customized (2) list column or a customized (2) content type. If *@SiteId* is NULL or *@WebId* is NULL, this result set MUST return zero rows. The result set is defined using T-SQL syntax, as follows:

```
ListId        uniqueidentifier;
```

ListId: Contains the list identifier for this list that uses a customized (2) list column or a customized (2) content type.

3.1.4.21 proc_GetParentWebUrl

The **proc_GetParentWebUrl** stored procedure is called to return the store-relative form URL of the parent site of a specified site. The stored procedure is defined using T-SQL syntax, as follows:

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

@WebId: The site identifier for a site.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.21.1 Parent Site URL Result Set

The Parent Site URL result set MUST be returned and MUST contain 1 row defined using T-SQL syntax as follows:

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

FullUrl: The store-relative form URL of the parent site that contains the site whose site identifier is *@WebId*.

3.1.4.22 proc_GetSiteProps

The **proc_GetSiteProps** stored procedure is called to request the metadata information of a site collection. The stored procedure is defined using T-SQL syntax, as follows:

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

@WebSiteId: The site collection identifier for the site collection that contains the site whose metadata is requested.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.22.1 Site Props Result Set

The Site Props result set returns a set of properties of a site collection whose site collection identifier is specified in the *@WebSiteId* parameter.

The Site Props result set MUST contain 1 row if the site collection exists, and MUST contain no rows if there is no such site collection.

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

```
  OwnerID                int,
  SecondaryContactID     int,
  PortalURL              nvarchar(260),
  PortalName             nvarchar(255),
  LastContentChange      datetime,
  LastSecurityChange     datetime;
```

OwnerID: An identifier of the user or **group (2)** that owns the site collection. The value MUST NOT be NULL.

SecondaryContactID: An identifier of the user or group that is the secondary contact of the site collection. This value MUST be NULL in the case where the secondary contact of the site collection is not set.

PortalURL: The complete URL of a portal server connected to the site collection.

PortalName: The name of a portal server used to connect to the site collection**.**

LastContentChange: The timestamp in UTC format that specifies the last time the content of the site collection was changed. The value MUST NOT be NULL.

LastSecurityChange: The timestamp in UTC format that specifies the last time the security settings of the site collection were changed. The value MUST NOT be NULL.

3.1.4.23 proc_GetTpWebMetaData

The **proc_GetTpWebMetaData** stored procedure is called to request the metadata information for a particular site.

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

```
  PROCEDURE proc_GetTpWebMetaData(
    @WebSiteId          uniqueidentifier,
    @WebDirName         nvarchar(256),
    @WebLeafName        nvarchar(128),
    @DGCacheVersion     bigint,
    @SystemId           SystemId = NULL
  );
```

@WebSiteId: The site collection identifier for the site collection that contains the site whose metadata is requested.

@WebDirName: The directory name of the site whose metadata is requested.

@WebLeafName: The leaf name of the site whose metadata is requested.

@DGCacheVersion: The version of the **domain group** cache as seen by the front-end Web server. It is used to compare with the store's domain group cache version to determine if an update is needed. A special value of -2 (Skip) is specified to indicate that information about the domain group cache versions is not requested. In this case, **proc_GetTpWebMetaData** MUST return the value "-2" in all columns of the domain group cache versions result set, and it MUST NOT return either the Domain Group Cache Back-End Database Server Update result set or the Domain Group Cache Front-End Web Server Update result set.

@SystemId: The **SystemID** of the user requesting the site metadata information

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion. Result set MUST have one or more records.
3	No site matched the given information. MUST NOT return a result set.
1271	Access to this site is blocked.

Result Sets: proc_GetTpWebMetaData MUST return up to 4 result sets. Some of the result sets are returned conditionally, and all result sets returned will be sent in the order specified in the following sections.

3.1.4.23.1 Domain Group Cache Versions Result Set

See the [MS-WSSFO2] section 2.2.5.4 - Domain Group Cache Versions Result Set, for details.

3.1.4.23.2 Domain Group Cache Back-End Database Server Update Result Set

See the [MS-WSSFO2] section 2.2.5.3 - Domain Group Cache Back-End Database Server Update Result Set, for details.

3.1.4.23.3 Domain Group Cache Front-End Web Server Update Result Set

See the [MS-WSSFO2] section 2.2.5.5 - Domain Group Cache Front-End Web Server Update Result Set, for details.

3.1.4.23.4 Site MetaData Result Set

See the [MS-WSSFO2] section 2.2.5.23 - Site Metadata Result Set, for details.

3.1.4.23.5 Site Event Receivers Result Set

See the [MS-WSSFO2] section 2.2.5.9 - Event Receivers Result Set, for details.

3.1.4.24 proc_GetUnghostedBaseFieldTempalteInSite

The **proc_GetUnghostedBaseFieldTempalteInSite** stored procedure is called to get the customized (2) field (2) definition of a field (2) in a scope under the specified site collection. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_GetUnghostedBaseFieldTempalteInSite(
  @SiteId       uniqueidentifier,
  @Scope        nvarchar(256),
  @FieldId      uniqueidentifier
);
```

@SiteId: The site collection identifier.

@Scope: The scope of the site in the store-relative form.

@FieldId: The field identifier for which to return the field (2) definition.

Return Code Values: An **integer** that MUST be zero.

Result Sets: MUST return the **Field Definition** result set.

3.1.4.24.1 Field Definition Result Set

The **Field Definition** result set returns the field identifier of the field (2) and the XML fragment that defines the customized (2) field (2) definition. The XML schema for this structure is defined in [MS-WSSCAML] section TPFields.

If the field (2) exists in the specified site collection and in the specified scope, the **Field Definition** result set MUST contain 1 row. If the field (2) does not exist in the specified scope or if the *@SiteId* parameter is invalid, the field definition result set MUST contain zero rows.

The field definition result set is defined using T-SQL syntax, as follows:

```
{FieldId}        uniqueidentifier,
Definition       ntext;
```

{FieldId}: Contains the field identifier of the field (2) for which the definition is being requested. This MUST be the same as the input *@FieldId* parameter.

Definition: Contains the customized (2) XML definition of the field (2). This MUST be NULL if the field (2) is **uncustomized**. The XML schema for this structure is defined in [MS-WSSCAML] section TPFields.

3.1.4.25 proc_GetUniqueListMetaData

The **proc_GetUniqueListMetaData** stored procedure is called to return metadata information and event receivers for a specified list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_GetUniqueListMetaData(
  @SiteId              uniqueidentifier,
  @WebId               uniqueidentifier,
  @ServerTemplate      int,
  @PrefetchListScope   bit
);
```

@SiteId: The site collection identifier for the site collection containing the list whose metadata is being requested.

@WebId: The site identifier for the site containing the list whose metadata is being requested.

@ServerTemplate: The identifier for the **list server template** that defines the base structure of this list.

@PrefetchListScope: A bit flag specifying whether the permissions result set is returned. If this bit is 1, **proc_GetUniqueListMetaData** MUST return the permissions result set.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion. At least 2 and at most 3 result sets MUST be returned.

Value	Description
13	No list matched the parameters supplied. MUST NOT return any result set.

Result Sets: MUST return at least 2 and at most 3 result sets in case of Successful completion depending upon input parameters. Result sets that are returned will be sent in the order defined in the following sections.

3.1.4.25.1 List Metadata Result Set

The List Metadata result set contains the metadata associated with the list (1). This result set MUST return one row for each valid list identifier.

The List Metadata result set is defined in [MS-WSSFO2] section 3.1.5.28.1 - List Metadata Result Set.

3.1.4.25.2 Unique Permissions Result Set

The Unique Permissions result set contains the permissions ACL associated with the list (1). This result set will be returned if *@PrefetchListScope* parameter is set to 1 and the permissions exist.

The Unique Permissions result set is defined in [MS-WSSFO2] section 3.1.5.37.10 – Unique Permissions Result Set.

3.1.4.25.3 NULL Unique Permissions Result Set

The NULL Unique Permissions result set contains an unlabeled result set with 1 row. This result set will be returned if *@PrefetchListScope* parameter is set to 1 and the permissions do NOT exist.

The NULL List Permissions result set is defined in [MS-WSSFO2] section 3.1.5.37.11 – NULL Unique Permissions Result Set.

3.1.4.25.4 Event Receivers Result Set

The Event Receivers result set contains information about the event receivers defined for this list (1). There MUST be 1 row in this result set per event receiver that is registered for this list or 0 rows if no event receivers exist.

The Event Receivers result set is defined in [MS-WSSFO2] section 3.1.5.6.4 – Event Receivers Result Set.

3.1.4.26 proc_GetWebExtendedMetaData

The **proc_GetWebExtendedMetaData** stored procedure is called to return the Web site metadata for creation date and most recent modification date for a given Web site. The stored procedure is defined using T-SQL syntax, as follows:

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

@WebId: The site identifier for a site. It MUST NOT be NULL

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.26.1 Creation and Modification Result Set

The result set returns the Creation and most recent Modification date and time of a site whose site identifier is specified in the *@WebId* parameter. The format of the date and time is: yyyy-mm-dd hh:mm:ss:mmm(24h)

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

```
Webs.TimeCreated    datetime NOT NULL,
{Modified}          datetime;
```

Webs.TimeCreated: Contains the site creation date and time.

{Modified}: Contains the most recent site modification date and time.

3.1.4.27 proc_GetWebFeatureList

The **proc_GetWebFeatureList** stored procedure is called to retrieve the set of Features that are marked active in a site collection or site. The stored procedure is defined using T-SQL syntax, as follows:

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

@SiteId: The site collection identifier of the site collection in which the feature is marked active.

@WebId: MUST be a site identifier containing all zeros if the feature is site collection scoped. Otherwise, this parameter is the site identifier of the target site in which the feature is marked active.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.27.1 Feature Set Result Set

Feature Set returns the list of feature identifiers for those features that are marked activate in the site collection or site. The Feature Set result set MUST contain 0 or more rows. The Feature Set result set is defined using T-SQL syntax, as follows:

```
FeatureId      uniqueidentifier;
```

FeatureId: MUST contain the feature identifier of the feature marked as active.

3.1.4.28 proc_GetWebIdOfListId

The **proc_GetWebIdOfListId** stored procedure is called to return the site identifier of the site containing a given list (1). The T-SQL syntax for the stored procedure is as follows:

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

@ListId: The list identifier for a list. This MUST NOT be NULL.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.28.1 WebId Result Set

Returns the site identifier of the site containing the list (1) specified by *@ListId*. This result set MUST be returned. If the list (1) specified by *@ListId* exists, the result set MUST contain 1 row, otherwise, it MUST be empty. The T-SQL syntax for the result set is as follows:

```
tp_WebId        uniqueidentifier;
```

tp_WebId: The Web ID for a site that contains the list (1) specified by the *@ListId* parameter.

3.1.4.29 proc_GetWebUsageData

The **proc_GetWebUsageData** stored procedure is called to obtain usage data for a site. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_GetWebUsageData (
  @WebSiteId        uniqueidentifier,
  @WebDirName       nvarchar(256),
  @WebLeafName      nvarchar(128),
  @BlobType         tinyint
);
```

@WebSiteId: An identifier for the site collection that contains the site from which usage data is being requested.

@WebDirName: The directory name of the site from which usage data is being requested.

@WebLeafName: The leaf name of the site from which usage data is being requested.

@BlobType: Specifies the type usage data being requested. The possible values for this parameter MUST be listed in the following table:

Value	Description
1	Specifies that monthly usage data is being requested.
!= 1	Specifies that daily usage data is being requested.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.

Value	Description
!= 0	Failed execution.

Result Sets: MUST return the Monthly Usage result set when the *@BlobType* parameter is 1, and it MUST return the Daily Usage result set when the *@BlobType* parameter is not 1:

3.1.4.29.1 Monthly Usage Result Set

The Monthly Usage result set returns usage information for the past 31 months. The Monthly Usage result set will be returned when the value of the *@BlobType* parameter is set to 1. The Monthly Usage result set MUST contain 0 rows if the site identified by the *@WebSiteId*, *@WebDirName* and *@WebLeafName* parameters does not exist; otherwise, it MUST contain 1 row. The Monthly Usage result set is defined using T-SQL syntax, as follows:

```
MonthlyUsageData            image NULL,
MonthlyUsageDataVersion     int NOT NULL;
```

MonthlyUsageData: A binary value that stores a site's monthly usage data. The structure of the binary value is specified in the Usage Data Binary Field Structure section.

MonthlyUsageDataVersion: An integer that indicates the number of times that the MonthlyUsageData field has been modified. If this value is NULL, the row MUST be ignored and the call to this stored procedure MUST be considered a failed execution.

3.1.4.29.2 Daily Usage Result Set

The Daily Usage result set returns usage information for the past 31 days. The Daily Usage result set will be returned when the value of the *@BlobType* parameter is set to a value different than 1. The Daily Usage result set MUST contain 0 rows if the site identified by the *@WebSiteId*, *@WebDirName* and *@WebLeafName* parameters does not exist; otherwise, it MUST contain 1 row. The Daily Usage result set is defined using T-SQL syntax, as follows:

```
  DailyUsageData            image NULL,
  DailyUsageDataVersion     int NOT NULL;
```

DailyUsageData: A binary value that stores daily usage data for a site. The structure of the binary value is specified in the Usage Data Binary Field Structure section.

DailyUsageDataVersion: An integer that indicates the number of times that the DailyUsageData field has been modified. If this value is NULL, the row MUST be ignored and the call to this stored procedure MUST be considered a failed execution

3.1.4.30 proc_IsContentTypeGhosted

The **proc_IsContentTypeGhosted** stored procedure is called to determine if the specified content type is uncustomized. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_IsContentTypeGhosted (
  @SiteId            uniqueidentifier,
  @ContentTypeId     varbinary(512)
);
```

@SiteId: The site collection identifier for the site collection that contains the content type specified by *@ContentTypeId*.

@ContentTypeId: The content type identifier of the content type to be checked for uncustomized. This MUST be of type **tContentTypeId**.

Return Code Values: An **integer** that MUST be zero.

Result Sets: MUST return the **Content Type is Ghosted** result set.

3.1.4.30.1 Content Type is Ghosted Result Set

The **Content Type is Ghosted** result set returns an **integer** that indicates whether or not the content type is uncustomized. If *@SiteId* is not NULL, *@ContentTypeId* is not NULL, and the content type specified by *@ContentTypeId* exists, this result set MUST contain 1 row. Otherwise, this result set MUST contain zero rows. The result set is defined using T-SQL syntax, as follows:

```
{IsGhosted}      int;
```

{IsGhosted}: Contains an **integer** that MUST be in the following table.

Value	Description
"0"	The content type specified by *@ContentTypeId* is not uncustomized.
"1"	The content type specified by *@ContentTypeId* is uncustomized.

3.1.4.31 proc_ListAllFileUrls

The **proc_ListAllFileUrls** stored procedure is called to get the URLs of all documents in a site. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ListAllFileUrls(
  @SiteId           uniqueidentifier,
  @WebUrl           nvarchar(260),
  @IncludeGhosts    bit
);
```

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

@WebUrl: The store-relative form URL to the site.

@IncludeGhosts: A bit flag specifying whether to also get uncustomized documents in the site**.** When only documents in the site are requested, the *@IncludeGhosts* flag MUST be set to zero ("0"). When uncustomized documents in the site are also requested, the *@IncludeGhosts* flag MUST be set to "1".

Return Code Values: An **integer** that MUST be listed in the following table:

Value	Description
"0"	Successful completion.
"3"	The site specified by the *@WebUrl* parameter does not exist in the specified site collection.

Result Sets: MUST return the **All File URLs** result set if the site specified by the *@WebUrl* parameter exists in the specified site collection**.**

3.1.4.31.1 All File URLs Result Set

The **All File URLs** result set returns the directory name and leaf name of all documents in the site. The **All File URLs** result set MUST be returned if the site specified by the *@WebUrl* parameter exists in the specified site collection. It MUST return 1 row for each document in the specified site. If the *@IncludeGhosts* parameter is set to "1", the result set MUST also return 1 row for each uncustomized document. If the *@IncludeGhosts* parameter is set to zero ("0"), the result set MUST NOT return a row for any uncustomized document. The **All File URLs** result set is defined using T-SQL syntax, as follows:

```
DirName        nvarchar(256),
LeafName       nvarchar(128);
```

DirName: Contains the directory name of the document or uncustomized document to be returned.

LeafName: Contains the leaf name of the document or uncustomized document to be returned.

3.1.4.32 proc_ListAllWebsOfSite

The **proc_ListAllWebsOfSite** stored procedure is called to retrieve the list of sites in a site collection. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ListAllWebsOfSite (
  @SiteId         uniqueidentifier,
  @Collation      nvarchar(32)
);
```

@SiteId: The site collection identifier for the site collection for which the child sites are to be retrieved.

@Collation: The collation order to be used to order the result set.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.32.1 SiteWebs Result Set

SiteWebs returns the list of sites in the specified site collection. The SiteWebs result set MUST return 1 row for each site. If a *@Collation* value is specified, the result set is ordered on the FullURL field using the specified *@Collation*. The SiteWebs result set is defined using T-SQL syntax, as follows:

```
FullUrl               nvarchar(256),
Id                    uniqueidentifier,
{ParentWebFullUrl}    nvarchar(256),
Language              int,
{Title}               nvarchar(255);
```

FullUrl: Contains the store-relative form URL of the site.

Id: Contains the site identifier of the site.

{ParentWebFullUrl}: Contains the store-relative form URL of the parent site. If a parent does not exist, the site's FullUrl is returned.

Language: Contains the **language code identifier (LCID)** of the site.

{Title}: Contains the title of the site. If the site has no title, an empty string is returned.

3.1.4.33 proc_ListChildWebs

The **proc_ListChildWebs** stored procedure is called to retrieve the list of child sites for a given site parent. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ListChildWebs (
  @SiteId        uniqueidentifier,
  @WebUrl        nvarchar(256),
  @Collation     nvarchar(32)
);
```

@SiteId: The site collection identifier for the site collection from which the child sites are to be retrieved.

@WebUrl: The store-relative form URL of the site from which the child sites are to be retrieved..

@Collation: The **collation** to be used to order the result set.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
3	The specified *@WebUrl* was not found for the given *@SiteId*.

Result Sets: MUST return the following result set:

3.1.4.33.1 ChildWebs Result Set

ChildWebs returns the list of sites whose parent site is the site specified by *@WebUrl*. The ChildWebs result set will return 1 row for each site. If a *@Collation* value is specified, the result set is ordered on the FullURL field using the specified *@Collation*. The ChildWebs result set is defined using T-SQL syntax, as follows:

```
FullUrl     nvarchar(256),
Id          uniqueidentifier,
Language    int,
{Title}     nvarchar(255);
```

FullUrl: Contains the store-relative form URL of the site.

Id: Contains the site identifier of the site.

Language: Contains the language code identifier (LCID) of the site.

{Title}: Contains the title of the site. If the site has no title, an empty string is returned.

3.1.4.34 proc_ListChildWebsFiltered

The **proc_ListChildWebsFiltered** stored procedure is called to get a filtered list of data about subsites for a specified parent site; the list MAY be sorted based on a specified **Windows collation name**. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ListChildWebsFiltered(
@SiteId                     uniqueidentifier,
@ParentWebId                uniqueidentifier,
@UserId                     int,
@Collation                  nvarchar(32) = '',
@WebTemplate                int = NULL,
@ProvisionConfig            smallint = NULL,
@ToLinkRecurringMeeting     bit = NULL
);
```

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

@ParentWebId: The site identifier of the specified parent site.

@UserId: The user identifier of the user making the request to the front-end Web server. This MUST NOT be NULL.

@Collation: The Windows collation name that is used to sort the filtered list of subsites. This MAY be NULL. If *@Collation* is not NULL, then it MUST be used to sort the filtered list of subsites.

@WebTemplate: The value of the identifier of the site definition that contains the site definition configuration used to provision this site that is used to filter the list of subsites. This MAY be NULL.

@ProvisionConfig: The value of the identifier of the site definition configuration used to provision this site that is used to filter the list of subsites. This MAY be NULL.

@ToLinkRecurringMeeting: A bit flag indicating the number and kind of meetings configured, if these are **Meeting Workspace sites**. This is to indicate how subsites are filtered. This MAY be NULL.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.34.1 List Child Webs Filtered Result Set

The List Child Webs Filtered Result set contains a filtered list of subsites for the specified parent site. This list MAY be sorted based on the value of the specified *@Collation* parameter.

The List Child Webs Filtered result set MUST return 1 row of data associated with each subsite if it satisfies all of the following conditions:

- - It is an immediate subsite of the specified parent site.
- - *@WebTemplate* is NULL

  Or

@WebTemplate is not NULL, and the value of the identifier of the site definition used to provision this subsite is the same as the specified *@WebTemplate* value and *@ProvisionConfig* is NULL.

Or

@WebTemplate is not NULL, the value of the identifier of the site definition used to provision this subsite is the same as the specified *@WebTemplate* value, *@ProvisionConfig* is not NULL and the value of *@ProvisionConfig* is same as the value of the identifier of the site definition configuration used to provision this subsite.

- *@ToLinkRecurringMeeting* is NULL

  Or

  @ToLinkRecurringMeeting is set to 1 and the subsite is a Meeting Workspace site and it has no meetings configured.

  Or

  @ToLinkRecurringMeeting is set to 0 and the subsite is a Meeting Workspace site and it has no recurring meetings configured.

If the value of *@Collation* is NULL or the empty string, then the result set MUST NOT be sorted. Otherwise, the result set MUST be sorted on the **site title** of the subsites based on the collation order specified by the value of *@Collation*.

The List Child Webs Filtered result set is defined using T-SQL syntax, as follows:

```
FullUrl                      nvarchar(256),
Id                           uniqueidentifier,
{Title}                      nvarchar(255),
{Description}                ntext,
Language                     int,
WebTemplate                  int,
ProvisionConfig              smallint,
MeetingCount                 smallint,
{Acl}                        image,
AnonymousPermMask            bigint,
FirstUniqueAncestorWebId     uniqueidentifier,
SecurityProvider             uniqueidentifier,
TimeCreated                  datetime,
{TimeListLastModified}       datetime;
```

FullUrl: Contains the store-relative form URL of the subsite.

Id: Contains the site identifier of the subsite.

{Title}: The site title of the subsite. If the site title is NULL, then the empty string MUST be returned.

{Description}: The **site description** of the subsite. If the site description is NULL, then the empty string MUST be returned.

Language: The Language Code Identifier (LCID) of the subsite.

WebTemplate: The identifier of the site definition that contains the site definition configuration used to provision the subsite.

ProvisionConfig: The identifier of the site definition configuration used to provision this subsite.

MeetingCount: If this subsite is a meeting workspace site, this value indicates the number of meetings that are configured. Otherwise, this value MAY return 0. The front-end Web server MUST ignore this value for subsites that are not meeting workspaces.

{Acl}: The binary serialization of the ACL (see [MS-WSSFO2] section 2.2.4.6 - WSS ACL Format) for this subsite. This value MUST be NULL if the subsite inherits security settings from its parent site.

AnonymousPermMask: Contains a 64-bit mask that specifies the permissions granted to **anonymous users** in this subsite. The bit mask values are defined in [MS-WSSFO2] section 2.2.2.13 - WSS Rights Mask.

FirstUniqueAncestorWebId: The site identifier of the closest ancestor site that does not inherit security settings from its parent site.

SecurityProvider: The identifier of the **external security provider** for this subsite. This MUST be NULL for subsites using the native security implementation.

TimeCreated: The time this subsite was created. This MUST be in UTC format.

{TimeListLastModified}: The last time any list (1) contained in this subsite was modified. This MUST be in UTC format.

3.1.4.35 proc_ListContentTypeInUse

The **proc_ListContentTypeInUse** stored procedure is called to list all usages of the specified content type. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ListContentTypeInUse (
  @SiteId           uniqueidentifier,
  @ContentTypeId    varbinary(512)
);
```

@SiteId: The site collection identifier for the site collection that contains the content type specified by *@ContentTypeId*.

@ContentTypeId: The content type identifier to list usages for. This MUST be of type tContentTypeId.

Return Code Values: An integer which MUST be 0.

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

3.1.4.35.1 Content Type Descendants Result Set

Returns all descendant content types of the specified content type. This result set MUST be returned and MUST contain 1 row for each content type that is a descendant of the content type specified by *@ContentTypeId*. The result set is defined using T-SQL syntax, as follows:

```
ContentTypeId    varbinary(512),
Scope            nvarchar(256);
```

ContentTypeId: Contains the Content Type ID of this content type that is a descendant of the content type specified by *@ContentTypeId*. This MUST be of type tContentTypeId.

Scope: Contains a store-relative form URL of the site or list (1) root folder to which this content type is registered.

3.1.4.35.2 Content Type List Usage Result Set

Returns all content types that are used in lists (1) that are the specified content type or are a Descendant content type of the specified content type. This result set MUST be returned and MUST contain 1 row for each content type that is used in each list (1) and is the content type specified by *@ContentTypeId* or is a Descendant content type of the content type specified by *@ContentTypeId*. The result set is defined using T-SQL syntax, as follows:

```
ContentTypeId    varbinary(512),
{Scope}      nvarchar(256);
```

ContentTypeId: Contains the content type identifier, in tContentTypeId format (see tContentTypeId), of this content type that is used in a list (1) and is a type or subtype of the content type specified by *@ContentTypeId*.

{Scope}: Contains a store-relative form URL of the root folder of the list (1) that this content type is used in.

3.1.4.36 proc_ListContentTypesInScope

The **proc_ListContentTypesInScope** stored procedure is called to list the site content types or site columns available to a given site (in other words, those registered to the site and those registered to the site's ancestor sites). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ListContentTypesInScope(
  @SiteId      uniqueidentifier,
  @Class       tinyint,
  @Scope       nvarchar(256)
);
```

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

@Class: The type of record to be retrieved. The parameter MUST be in the following table:

Value	Description
0	Site columns.
1	Site content types.

@Scope: The scope to retrieve site content types or site columns from.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.

Value	Description
144	*@Scope* refers to a site that is not within the site collection designated by the *@SiteId* parameter.

Result Sets: MUST return the following result set:

3.1.4.36.1 Result Set

If *@Class* is equal to 0 then the stored procedure returns a list of all site columns registered in the site designated by the *@SiteId* and *@Scope* parameters or to the site's ancestor sites. This result set MUST contain 1 row for each site column registered in the site or its ancestor sites.

If *@Class* is equal to 1 then the stored procedure returns a list of all site content types registered in the site designated by the *@SiteId* and *@Scope* parameters or to the site's ancestor sites. This result set MUST contain 1 row for each site content type registered in the site or its ancestor sites.

For both values of *@Class* listed in the preceding paragraphs, the result set is defined using T-SQL syntax, as follows:

```
ContentTypeId    varbinary(512),
Scope            nvarchar(256),
Definition       ntext,
NextChildByte    tinyint,
Version          int,
ResourceDir      nvarchar(128);
```

ContentTypeId: Contains a content type identifier for the site content type or site column.

Scope: Contains the store-relative form URL of the site to which this site content type or site column is registered.

Definition: MUST contain the XML fragment of the site content type or site column or NULL if the site content type or site column has no XML fragment. The XML schemas for these structures are defined in [MS-WSSCAML] section TPContentTypeReferences, and [MS-WSSCAML] section TPFields, respectively.

NextChildByte: If *@Class* is equal to zero, this value MUST be "0x00". If *@Class* is equal to "1", this value MUST be a number between 0x00 and 0xFF.

Version: Contains the version of the site content type or site column.

ResourceDir: If *@Class* is equal to zero, this value MUST be NULL. If *@Class* is equal to "1", this value MUST contain the leaf name of the content type resource folder for this site content type.

3.1.4.37 proc_ListContentTypesInWeb

The **proc_ListContentTypesInWeb** stored procedure is called to list all the site content types or site columns in the specified site. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ListContentTypesInWeb(
  @SiteId              uniqueidentifier,
  @Class               tinyint,
  @Scope               nvarchar(256)
);
```

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

@Class: The type of record that should be retrieved. The parameter MUST be in the following table:

Value	Description
0	Site columns.
1	Site content types.

@Scope: The store-relative form URL of the site to retrieve site content types or site columns from.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.37.1 Result Set

If *@Class* is equal to 0 then the stored procedure returns a list of all site columns registered in the site designated by the *@SiteId* and *@Scope* parameters. This result set MUST contain 1 row for each site column registered in the site.

If *@Class* is equal to 1 then the stored procedure returns a list of all site content types registered in the site designated by the *@SiteId* and *@Scope* parameters. This result set MUST contain 1 row for each site content type registered in the site.

For both values of *@Class* listed in the preceding paragraphs, the result set is defined using T-SQL syntax, as follows:

```
ContentTypeId         varbinary(512),
Scope                 nvarchar(256),
Definition            ntext,
NextChildByte         tinyint,
Version               int,
ResourceDir           nvarchar(128);
```

ContentTypeId: Contains an identifier of type tContentTypeId for the site content type or site column.

Scope: Contains the store-relative form URL of the site to which this site content type or site column is registered.

Definition: MUST contain the XML fragment that defines the site content type or site column or NULL if the site content type or site column has no XML fragment. The XML schemas for these structures are defined in [MS-WSSCAML] section TPContentTypeReferences, and [MS-WSSCAML] section TPFields, respectively.

NextChildByte: If *@Class* is equal to 0, this value MUST be 0x00. If *@Class* is equal to 1, this value MUST be a number between 0x00 and 0xFF.

Version: Contains the version of the site content type or site column.

ResourceDir: If *@Class* is equal to 0, then this value MUST be NULL. If *@Class* is equal to 1, then this value contains the leaf name of the content type’s resource folder.

3.1.4.38 proc_ListDerivedContentTypes

The **proc_ListDerivedContentTypes** stored procedure is called to list all site content types and list content types that are derived from a given site content type. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ListDerivedContentTypes(
  @SiteId           uniqueidentifier,
  @ContentTypeId    varbinary(512)
);
```

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

@ContentTypeId: The identifier of the site content type from which the requested site content types are derived. This MUST be of type tContentTypeId.

Return Code Values: An integer which MUST be 0.

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

3.1.4.38.1 Derived Site Content Types Result Set

Returns a list of site content types that are derived from the site content type designated by the *@ContentTypeId* parameter. This result set MUST be returned and MUST contain 1 row for each site content type derived from the given parent site content type. The result set is defined using T-SQL syntax, as follows:

```
ContentTypeId    varbinary(512),
Scope            nvarchar(256),
Definition       ntext,
NextChildByte    tinyint,
Version          int,
ResourceDir      nvarchar(128);
```

ContentTypeId: contains an identifier of type tContentTypeId for the site content type.

Scope: contains the store-relative form URL of the site to which this site content type is registered.

Definition: MUST contain the XML fragment of the site content type or NULL if the site content type does not have an XML fragment. The XML schema for this structure is defined in [MS-WSSCAML] section TPContentTypeReferences.

NextChildByte: This value MUST be a number between 0x00 and 0xFF.

Version: Contains the version of the site content type.

ResourceDir: This value contains the leaf name of the content type resource folder.

3.1.4.38.2 Derived Content Types Result Set

Returns a list of content types associated with lists (1) that are derived from the site content type designated by the *@ContentTypeId* parameter. This result set MUST be returned and MUST contain 1 row for each content type derived from the given parent site content type. The result set is defined using T-SQL syntax, as follows:

```
ContentTypeId    varbinary(512),
WebId            uniqueidentifier,
ListId           uniqueidentifier;
```

ContentTypeId: contains an identifier for the content type. This MUST be of type tContentTypeId.

WebId: contains a site identifier that identifies the site to which the content type is registered.

ListId: contains a list identifier that identifies the list (1) to which the content type is registered.

3.1.4.39 proc_ListsUsingFieldTemplate

The **proc_ListsUsingFieldTemplate** stored procedure is called to get a list of lists (1) in a site collection that include a specific field (2). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ListsUsingFieldTemplate(
  @SiteId          uniqueidentifier,
  @FieldId         varbinary(512),
  @BaseTypes       int
);
```

@SiteId: The site collection identifier of the site collection in which to look for the field (2).

@FieldId: The field identifier of the field (2) in varbinary format.

@BaseTypes: An **integer** bit pattern indicating which base types use the field (2). (For more information about base types, see List Base Type Pattern.) The bit pattern is described in the List Base Type Pattern section.

Return Code Values: An **integer** that MUST be zero.

Result Sets: MUST return the **Lists Using Field** result set.

3.1.4.39.1 Lists Using Field Result Set

The **Lists Using Field** result set contains a list of GUID pairs for the sites and lists (1) under a site collection using a particular field (2). The **Lists Using Field** result set MUST contain 1 row per list (1) that uses the specified field (2). The **Lists Using Field** result set is defined using T-SQL syntax, as follows:

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

WebId: The site identifier containing the list (1) specified by ListId.

ListId: The list (1) that uses the field (2) specified by *@FieldId*.

3.1.4.40 proc_ListUnghostedFieldTemplatesInList

The **proc_ListUnghostedFieldTemplatesInList** stored procedure is called to get the customized (2) field (2) definitions associated with a specific list (1). The stored procedure is defined using T-SQL syntax, as follows:

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

@SiteId: The site collection identifier in which the specified list (1) exists.

@WebId: The site identifier of the site that contains the specified list (1).

@ListId: The list identifier for which the field (2) definitions are being requested.

Return Code Values: An **integer** that MUST be zero.

Result Sets: MUST return the **Unghosted List Fields** result set.

3.1.4.40.1 Unghosted List Fields Result Set

The **Unghosted List Fields** result set contains 1 row for each customized (2) field (2) that is associated with the specified list (1). The result set MUST contain zero rows if there are no customized (2) fields (2) associated with the list (1). The **Unghosted List Fields** result set is defined using T-SQL syntax, as follows:

```
{FieldId}       uniqueidentifier,
Definition      ntext;
```

{FieldId}: The field identifier of the field (2).

Definition: The XML fragment of the field (2). The XML schema for this structure is defined in [MS-WSSCAML] section TPFields.

3.1.4.41 proc_MakeViewDefaultForContentType

The **proc_MakeViewDefaultForContentType** stored procedure is called to assign a default view for a content type in a specific list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_MakeViewDefaultForContentType(
  @ListId            uniqueidentifier,
  @ViewId            uniqueidentifier,
  @ContentTypeId     varbinary(512)
);
```

@ListId: A list identifier for the list (1) whose default view for a content type is being set.

@ViewId: A **view identifier** for the view that will be set as the default view for the content type. If *@ViewId* is NULL the stored procedure MUST NOT change the back-end database server.

@ContentTypeId: A content type identifier for the content type whose default view is being set. This value MUST be the identifier of a content type that descends from the folder content type. This MUST be of type tContentTypeId.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.42 proc_MakeViewDefaultForList

The **proc_MakeViewDefaultForList** stored procedure is used to set the default list view for a list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_MakeViewDefaultForList (
  @ListId       uniqueidentifier,
  @ViewId       uniqueidentifier
);
```

@ListId: The list identifier of the list (1) whose default list view will be set.

@ViewId: The view identifier of the view that will become the default list view for the specified list (1).

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.43 proc_MakeViewMobileDefaultForList

The **proc_MakeViewMobileDefaultForList** stored procedure is called to set the default mobile list view for a list (1) when the view is being displayed on a **Mobile device**. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_MakeViewMobileDefaultForList(
  @ListId      uniqueidentifier,
  @ViewId       uniqueidentifier
);
```

@ListId: The list identifier of the list (1) whose default mobile list view will be set.

@ViewId: The view identifier of the view that will become the default mobile list view for the specified list (1).

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.44 proc_MapContentTypeToList

The **proc_MapContentTypeToList** stored procedure is called to record that a list content type or list column is being used in a particular list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_MapContentTypeToList(
  @SiteId           uniqueidentifier,
  @WebId            uniqueidentifier,
  @ListId           uniqueidentifier,
  @ContentTypeId    varbinary(512),
  @Class            bit
);
```

@SiteId: The site collection identifier of the site collection in which the list (1) resides. This MUST NOT be NULL.

@WebId: The site identifier of the site in which the list (1) resides. This MUST NOT be NULL.

@ListId: The list identifier of the list (1) to which the list content type is to be associated. This MUST NOT be NULL.

@ContentTypeId: The content type identifier of the list content type being associated. This MUST be of type tContentTypeId. This MUST NOT be NULL.

@Class: This parameter MUST be in the following table:

Value	Description
0	*@ContentTypeId* refers to a list column and it MUST be 16 bytes long.
1	*@ContentTypeId* refers to a content type.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.45 proc_MapFieldToContentType

The **proc_MapFieldToContentType** stored procedure is called to record that a site column is being used in a particular site content type. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_MapFieldToContentType(
  @SiteId             uniqueidentifier,
  @WebId              uniqueidentifier,
  @ContentTypeId      varbinary(512),
  @FieldId            uniqueidentifier
);
```

@SiteId: The site collection identifier of the site collection in which the site content type resides. This MUST NOT be NULL.

@WebId: The site identifier of the site in which the site content type resides. This MUST NOT be NULL.

@ContentTypeId: The content type identifier of the site content type being associated. This MUST be of type tContentTypeId. This MUST NOT be NULL.

@FieldId: The field identifier of the site column being mapped. This MUST NOT be NULL.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.46 proc_MapUrlToListAndView

The **proc_MapUrlToListAndView** stored procedure is called to get the list identifier and view identifier of the specified view page. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_MapUrlToListAndView(
  @SiteID        uniqueidentifier,
  @WebID         uniqueidentifier,
  @DirName       nvarchar(256),
  @LeafName      nvarchar(128)
);
```

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

@WebID: The site identifier of the site for the document specified by *@DirName* and *@LeafName*.

@DirName: The directory name of the view page.

@LeafName: The leaf name of the view page.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.1.4.46.1 Map URL to List and View Result Set

The Map URL to list and view result set returns information about the view for the specified view page. The Map URL to list and view result set MUST be returned. The Map URL to list and view result set MUST return 1 row if the view page was found. If the view page was not found, then Map URL to list and view result set MUST return 0 rows. The Map URL to list and view result set is defined using T-SQL syntax, as follows:

```
tp_Id          uniqueidentifier,
tp_ListId      uniqueidentifier;
```

tp_Id: Contains the view identifier of the view for the view page specified by the *@DirName* and *@LeafName* stored procedure parameters.

tp_ListId: Contains the list identifier of the list (1) which contains the view specified by tp_Id.

3.1.4.47 proc_MapV2FieldToList

The **proc_MapV2FieldToList** stored procedure is called to associate a field (2) to a list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_MapV2FieldToList(
  @SiteId           uniqueidentifier,
  @WebId            uniqueidentifier,
  @ListId           uniqueidentifier,
  @ContentTypeId    varbinary(512)
);
```

@SiteId: The site collection identifier in which the list (1) exists. This MUST NOT be NULL.

@WebId: The site identifier in which the list (1) exists. This MUST NOT be NULL.

@ListId: The list identifier. This MUST NOT be NULL.

@ContentTypeId: The field identifier. This MUST be of type tContentTypeId. This MUST be 16 bytes long.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.48 proc_markWebAsProvisioned

The **proc_markWebAsProvisioned** stored procedure is called to remove the 'unprovisioned' bit from the site property flags (see [MS-WSSFO2] section 2.2.2.10 - Site Property Flags) for a site. Once this bit is removed from site property flags, the site is considered to be provisioned and all steps to provision the site have been completed. The stored procedure is defined using T-SQL syntax, as follows:

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

@WebId: The site identifier of the site that will be set as provisioned.

Return Code Values: An integer value of 0, which the protocol client MUST ignore.

Result Sets: MUST NOT return any result sets.

3.1.4.49 proc_MergeWeb

The **proc_MergeWeb** stored procedure is called to convert a site into a folder on its parent site. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_MergeWeb(
  @WebSiteId      uniqueidentifier,
  @WebUrl         nvarchar(260)
);
```

@WebSiteId: The site collection identifier of the site collection that contains the site to be converted.

@WebUrl: The store-relative form URL of the site to be converted.

Return Code Values: An **integer** that MUST be listed in the following table:

Value	Description
"0"	Successful completion.
"3"	The site to be converted does not exist in the specified site collection.
"5"	The site to be converted is a top-level site and cannot be converted.
"133"	The site to be converted contains 1 or more lists (1) or **document libraries**.

Result Sets: MUST return the **Audit Mask** result set.

3.1.4.49.1 Audit Mask Result Set

The **Audit Mask** result set contains the information about the **audit flags** associated with this site. The **Audit Mask** result set MUST be returned and MUST contain 1 row.

The **Audit Mask** result set is defined in [MS-WSSFO2] section 3.1.5.21.1.

3.1.4.50 proc_MiniSproc

The **proc_MiniSproc** stored procedure is called to return specific metadata for a given site. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_MiniSproc(
    @WebSiteId         uniqueidentifier,
    @WebId             uniqueidentifier,
    @Url               nvarchar(260),
    @DGCacheVersion    bigint,
    @SystemId          varbinary(512) = NULL
)
```

@WebSiteId: The site identifier for a site collection. This MUST NOT be NULL.

@WebId: The site identifier for a site. If NULL, this will request additional result sets, otherwise the document specified by *@Url* MUST be in this specified site.

@Url: The **store-relative URL** for a document in the selected site collection. If *@WebId* is NOT NULL, this MUST NOT be NULL and MUST be contained in the selected site. If this is an invalid store-relative URL (including NULL) the procedure will return with an error code prior to producing the last result set.

@DGCacheVersion: The version number of the domain group map cache in the front-end Web server. It is used to compare with the domain group cache version of the back-end database server to determine if an update is needed. A special value of -2 (Skip) is specified to indicate that information about the domain group cache versions is not requested. This value is ignored if *@WebId* is NOT NULL, otherwise it MUST NOT be NULL.

@SystemId: The SystemID of the user. The user MUST have read access to the document specified by *@Url* to return the last result set.

Return Code Values: an integer which MUST be listed in the following table.

Value	Description
0	Successful completion.
2	The specified document is invalid or is not readable by the specified user.
3	Derived site identifier does not exist based on given site collection identifier.
1168	*@WebSiteId* does not specify a valid site collection identifier.
1271	Access to this site is blocked.

Result Sets: MUST return zero to 6 of the following result sets:

3.1.4.50.1 Site URL Result Set

The Site URL result set contains a string for the URL of a site. The stored procedure MUST return this result set when the *@WebId* is NULL and the specified site collection does exist. When returned it MUST return 1 row. The T-SQL syntax for the result set is as follows:

```
{Url}       nvarchar(385)
```

Url: The store-relative form URL of the site specified by *@WebSiteId* and *@Url*. If *@Url* is a site, this MUST match *@Url*, if *@Url* is not contained in site collection this MUST be the top-level site.

3.1.4.50.2 Domain Group Cache Versions Result Set

The stored procedure MUST return this result set when the *@WebId* is NULL and the specified site collection does exist**.**

See [MS-WSSFO2] section 2.2.5.4 Domain Group Cache Versions Result Set for details.

3.1.4.50.3 Domain Group Cache Back-End Database Server Update Result Set

The stored procedure MUST return this result set when the rules specified in the following reference are met and *@WebId* is NULL and the specified site collection exists.

See [MS-WSSFO2] section 2.2.5.3 - Domain Group Cache Back-End Database Server Update Result Set, for details.

3.1.4.50.4 Domain Group Cache Front-End Web Server Update Result Set

The stored procedure MUST return this result set when the rules specified in the following reference are met and *@WebId* is NULL and the specified site collection does exist**.**

See [MS-WSSFO2] section 2.2.5.5 - Domain Group Cache Front-End Web Server Update Result Set, for details.

3.1.4.50.5 Site Metadata Result Set

The Site Metadata result set contains specialized site metadata. The stored procedure MUST return this result set when the *@WebId* is NULL and the specified site collection does exist. This result set MUST contain one row if the site is found; otherwise 0 rows are returned.

See the [MS-WSSFO2] section 3.1.5.15.14 - Site Metadata Result Set, for details.

3.1.4.50.6 Event Receivers Result Set

This result set contains information about the event receivers defined for this **event host**. The stored procedure MUST return this result set when the *@WebId* is NULL and the specified site collection does exist.

There MUST be 1 row in this result set for each event receiver that is registered for this event host. The result set is ordered by SiteId, WebId, HostId, Type, HostType, SequenceNumber, Assembly.

See the [MS-WSSFO2] section 3.1.5.6.4 - Event Receivers Result Set, for schema details.

3.1.4.50.7 User Document Security Context Result Set

User Document Security Context contains security context for a given published user document. The stored procedure MUST return this result set when no return code error is encountered. The User Document Security Context result set MUST return 1 row.

The User Document Security Context result set is defined using T-SQL syntax as follows:

```
{PersonalPartsExist}    int,
{DraftOwnerId}          int,
{Lists_Flags}           bigint,
Acl                     image,
AnonymousPermMask      bigint,
{Level}                tinyint,
{IsListItem}           bit;
```

{PersonalPartsExist}: An int value specifying whether the document contains any personal Web Parts. It MUST be 1 if there exist personal Web Parts on the document. It MUST be 0 otherwise. This value MUST NOT be NULL.

{DraftOwnerId}: The user who published the document as a draft. This value MUST be 0 if the document does not exist or is not a draft version.

{Lists_Flags}: A bigint that specifies the list flags (see [MS-WSSFO2] section 2.2.2.5 - List Flags) on the list (1) which contains the document as a list item. If the document is not a list item in a list (1), List_Flags MUST be 0.

Acl: The binary serialization of the ACL (see [MS-WSSFO2] section 2.2.4.6 - WSS ACL Format) for the document. This is either explicitly defined or inherited from the document's parent object.

AnonymousPermMask: A WSS rights mask (see [MS-WSSFO2] section 2.2.2.13) indicating the rights granted to an anonymous user, or to a user who has no specific rights to the document.

{Level}: A tinyint value specifying the document's publishing level type ([MS-WSSFO2] section 2.2.2.6). This value MUST not be NULL.

{IsListItem}: A bit value that MUST be 1 if the document is a list item in a list (1), otherwise it MUST be 0. This value MUST NOT be NULL.

3.1.4.51 proc_ProvisionContentType

The **proc_ProvisionContentType** stored procedure is called to make a site content type or site column available on a particular scope. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ProvisionContentType(
  @SiteId           uniqueidentifier,
  @Scope            nvarchar(256),
  @Class            tinyint,
  @NextChildByte    tinyint,
  @ContentTypeId    varbinary(512),
  @ResourceDir      nvarchar(128) = NULL
);
```

@SiteId: The site collection identifier of the site collection that contains the scope where the site content type or site column is to be made available. This MUST NOT be NULL.

@Scope: The store-relative form URL of the scope that the site content type or site column is to be made available in. This MUST NOT be NULL.

@Class: MUST be 0 if a site column is being made available. MUST be 1 if a site content type is being made available.

@NextChildByte: If *@Class* is equal to 0, this value MUST be 0x00. If *@Class* is equal to 1, this value MUST be a number between 0x00 and 0xFF.

@ContentTypeId: The content type identifier of the site content type that is to be made available. This MUST be of type tContentTypeId. This MUST NOT be NULL.

@ResourceDir: The store-relative form URL identifying the content type resource folder of the site content type that is being made available. This MUST be NULL if a site column is being made available.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.52 proc_RenameListItemContentType

The **proc_RenameListItemContentType** stored procedure is called to set the content type display name field (2) for the list items, documents or folders in a particular list (1) in the back-end database server. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_RenameListItemContentType(
  @SiteId              uniqueidentifier,
  @ListId              uniqueidentifier,
  @ContentTypeId       varbinary(512),
  @ContentTypeName     nvarchar(255)
);
```

@SiteId: The site collection identifier of the site collection that contains the list (1) where the content type is to be changed.

@ListId: The list identifier of the list (1) that uses the content type whose display name is to be changed.

@ContentTypeId: The content type identifier of the content type whose display name is to be changed. This MUST be of type tContentTypeId.

@ContentTypeName: The new name of the content type whose display name is to be changed. This MUST NOT be NULL.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.53 proc_SetListFormToUrl

The **proc_SetListFormToUrl** stored procedure is called to set the default form for the **display form**, edit form, or **new form** for a list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_SetListFormToUrl(
  @SiteId       uniqueidentifier,
  @WebId        uniqueidentifier,
  @ListId       uniqueidentifier,
  @PageUrl      nvarchar(260),
  @PageType     tinyint
);
```

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

@WebId: This stored procedure parameter MUST be ignored.

@ListId: The list identifier of the list (1) containing the form specified by the *@PageUrl* parameter.

@PageUrl: The store-relative form URL that will become the default form for the list (1).

@PageType: The type of form of the URL specified by the *@PageUrl* parameter. This parameter MUST be one of the following:

Value	Description
"4"	The form specified by the *@PageUrl* parameter is a display form.
"6"	The form specified by the *@PageUrl* parameter is an edit form.
"8"	The form specified by the *@PageUrl* parameter is a new form.

Return Code Values: An **integer** that MUST be listed in the following table:

Value	Description
"0"	Successful completion.
"126"	This return code MUST be returned if any of the following conditions are met: ▪ The URL specified by the *@PageUrl* parameter does not exist for the site collection specified by the *@SiteId* parameter. ▪ The specified list (1) does not already contain a form matching the specified *@PageType* parameter. ▪ The Web Part Page specified by *@PageUrl* is a customized (2) Web Part Page.
127	This return code MUST be returned if any of the following conditions are met: ▪ More than one URL specified by the *@PageUrl* parameter exists matching the specified *@PageType* parameter for the specified list (1). ▪ The URL specified by the *@PageUrl* parameter does not match the specified *@PageType* stored procedure.

Result Sets: MUST NOT return any result sets.

3.1.4.54 proc_SetSiteFlags

The **proc_SetSiteFlags** stored procedure is called to set the **site collection flags** of a site collection. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_SetSiteFlags(
  @SiteId       uniqueidentifier,
  @bitsToSet    int,
  @bitMask      int
);
```

@SiteId: The site collection identifier for a site collection

@bitsToSet: Specifies the value of the bit flag. The valid values of the flag MUST be specified in [MS-WSSFO2], section 2.2.2.21 - Site Collection Flags.

@bitMask: Specifies the mask of bits to set.

Return Code Values: An integer value which MUST be in the following table:

Value	Description
0	Successful completion.
3	The site collection specified by *@SiteId* was not found

Result Sets: MUST NOT return any result sets.

3.1.4.55 proc_SetSitePortalProps

The **proc_SetSitePortalProps** stored procedure is called to specify the **portal site** of a site collection. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_SetSitePortalProps (
  @WebSiteId         uniqueidentifier,
  @SitePortalURL     nvarchar(260),
  @SitePortalName    nvarchar(255)
);
```

@WebSiteId: The site collection identifier for a site collection

@SitePortalURL: The absolute URL of the portal site

@SitePortalName: The name of a site in the site collection

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
3	The site collection specified by *@WebSiteId* was not found

Result Sets: MUST NOT return any result sets.

3.1.4.56 proc_SetSiteProps

The **proc_SetSiteProps** stored procedure is called to set OwnerID and SecondaryContactID of a site collection. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_SetSiteProps(
  @SiteId                uniqueidentifier,
  @OwnerID               int,
  @SecondaryContactID    int
);
```

@SiteId: The site collection identifier for a site collection whose values are to be updated.

@OwnerID: The user identifier for the owner of the site collection.

@SecondaryContactID: The user identifier for the secondary contact for the site collection.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
3	The site collection specified by *@SiteId* was not found

Result Sets: MUST NOT return any result sets.

3.1.4.57 proc_SetWebMetainfo

The **proc_SetWebMetainfo** stored procedure is called to set metadata information for a given **Central Administration site** or user-defined site. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_SetWebMetainfo(
  @WebSiteId                uniqueidentifier,
  @WebUrl                   nvarchar(260),
  @MetaInfo                 image,
  @Flags                    int,
  @DefTheme                 nvarchar(64),
  @IncrementSiteTimeStamp   bit,
  @MasterUrl                nvarchar(260),
  @CustomMasterUrl          nvarchar(260),
  @@WebId                   uniqueidentifier OUTPUT
);
```

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

@WebUrl: The URL for a given site

@MetaInfo: A metadictionary (see [MS-FPSE], section 2.2.2.5.4) for the document. This value MUST be NULL if the document does not exist.

@Flags: A site collection flags value that indicates the settings for the site collection that contains this site.

@DefTheme: The name of a **theme** that is used as part of the display of the site.

@IncrementSiteTimeStamp: 0 if updating the metadata or 1 if creating it.

@MasterUrl: The URL of **master pages** for a given site

@CustomMasterUrl: The URL of custom master pages for a given site

@@WebId: The site identifier of a site returned to caller based on a given site URL.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
3	The site collection specified by *@WebSiteId* or *@WebUrl* was not found, or a given *@WebUrl* is not a top-level site
212	When one of the following condition is met for a given site collection: 1. When adding content to the site is prevented or accessing to the site is blocked 2. When adding content to the site is prevented or access to the site is blocked, and disk space used by the site is over allowed disk space limit 3. When disk space used by the site is over allowed disk space limit and user is creating a new site collection

Result Sets: MUST NOT return any result sets.

3.1.4.58 proc_SetWebUsageData

The **proc_SetWebUsageData** stored procedure is called to store usage data for a site. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_SetWebUsageData(
  @WebSiteId           uniqueidentifier,
  @WebUrl              nvarchar(260),
  @BlobTypeToUpdate    int,
  @UsageData           image,
  @BWUsed              bigint,
  @UsageDataVersion    int,
  @DayLastAccessed     smallint
);
```

@WebSiteId: The site colleciton identifier for the site collection that contains the site for which usage data is being stored. This parameter MUST NOT be NULL.

@WebUrl: The store-relative form URL of the site for which usage data is being stored. This parameter MUST NOT be NULL.

@BlobTypeToUpdate: Specifies the type of usage information being stored. The possible values for this parameter MUST be listed in the following table:

Value	Description
0	Specifies that daily usage data is being stored.
!= 0	Specifies that monthly usage data is being stored.

@UsageData: A binary structure containing usage data. The structure of the binary value is specified in the Usage Data Binary Field Structure section.

@BWUsed: The number of bandwidth bytes consumed by the usage data being stored. MUST be 0 if *@BlobTypeToUpdate* is different than 0.

@UsageDataVersion: The number of times that usage data has been stored for the site**.** It MUST be 0 if no previous usage data has been stored. It MUST be incremented by one after each successful completion. This parameter MUST NOT be NULL.

@DayLastAccessed: The number of days since January 1, 1899 until the date the usage data is stored. This parameter MUST NOT be NULL.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
!= 0	Failed execution.

Result Sets: MUST NOT return any result sets.

3.1.4.59 proc_StoreUserInfoListInfo

The **proc_StoreUserInfoListInfo** stored procedure is called to establish the row and column of the UserData View ([MS-WSSFO2] section 2.2.5.8) that will store the **user activity status** for the specified list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_StoreUserInfoListInfo(
  @SiteId           uniqueidentifier,
  @ListId           uniqueidentifier,
  @RowOrdinal       int,
  @ColName          nvarchar(64)
);
```

@SiteId: The site collection identifier of the site collection that contains the specified list (1).

@ListId: The list identifier of the list (1).

@RowOrdinal: A **zero-based index** integer number that identifies the row of the UserData View in which the user activity status will be stored for the specified list (1).

@ColName: The SQL column name of the UserData View in which the user activity status will be stored for the specified list (1).

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST NOT return any result sets.

3.1.4.60 proc_UnmapContentTypeFromList

The **proc_UnmapContentTypeFromList** stored procedure is called to delete an association between a content type and a list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDUREproc_UnmapContentTypeFromList(
  @SiteId           uniqueidentifier,
  @ContentTypeId    varbinary(512)
);
```

@SiteId: The site collection identifier of the site collection in which the content type resides.

@ContentTypeId: The content type identifier of the list content type being unmapped. This MUST be of type tContentTypeId.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.61 proc_UnmapFieldFromList

The **proc_UnmapFieldFromList** stored procedure is called to delete the association of a field (2) with a list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_UnmapFieldFromList(
  @SiteId           uniqueidentifier,
  @WebId            uniqueidentifier,
  @ListId           uniqueidentifier,
  @ContentTypeId    varbinary(512)
);
```

@SiteId: The site collection identifier in which the list (1) exists.

@WebId: The site identifier in which the list (1) exists.

@ListId: The list identifier.

@ContentTypeId: The field identifier in binary format. This MUST be of type tContentTypeId.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.62 proc_UnmapFieldsFromContentType

The **proc_UnmapFieldsFromContentType** stored procedure is called to delete an association between a site content type and the site columns mapped to it. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_UnmapFieldsFromContentType(
  @SiteId           uniqueidentifier,
  @WebId            uniqueidentifier,
  @ContentTypeId    varbinary(512)
);
```

@SiteId: The site collection identifier of the site collection in which the content type resides.

@WebId: The site identifier of the site in which the content type resides.

@ContentTypeId: The content type identifier of the site content type from which the association is being removed. This MUST be of type tContentTypeId.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.1.4.63 proc_UpdateContentTypeInScope

The **proc_UpdateContentTypeInScope** stored procedure is called to update the definition of a site content type or site column. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_UpdateContentTypeInScope (
  @SiteId           uniqueidentifier,
  @Class            tinyint,
  @Scope            nvarchar(256),
  @ContentTypeId    varbinary(512),
  @Definition       ntext,
  @Version          int
);
```

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

@Class: The type of record to be retrieved. The parameter MUST be in the following table:

Value	Description
0	Site columns.
1	Site content types.

@Scope: The store-relative form URL of the site to update the site content type or site column in.

@ContentTypeId: The content type identifier of the site content type or site column to be updated. This MUST be of type tContentTypeId.

@Definition: The XML fragment of the site content type or site column. The XML schemas for these structures are defined in [MS-WSSCAML], section TPContentTypeReferences, and [MS-WSSCAML], section TPFields, respectively.

@Version: The version of the site content type or site column to update.

Return Code Values: An integer that MUST be listed in the following table:

Value	Description
0	Successful completion.

Value	Description
2	The requested site content type or site column does not exist.
212	The site collection is locked.
1150	The *@Version* parameter doesn't match the version of the existing record in the database.
1816	The site collection quota for the specified site collection has been exceeded.

Result Sets: MUST NOT return any result sets.

3.1.4.64 proc_UpdateFeatureProperties

The **proc_UpdateFeatureProperties** stored procedure is called to update the Feature Property Definitions of a feature marked as active. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_UpdateFeatureProperties(
  @SiteId           uniqueidentifier,
  @WebId            uniqueidentifier,
  @FeatureId        uniqueidentifier,
  @Flags            int = 0,
  @Properties       ntext = NULL
);
```

@SiteId: The site collection identifier of the site collection in which the feature is marked active.

@WebId: MUST be a site identifier containing all zeros if the feature is site collection scoped. Otherwise, this parameter MUST be the site identifier of a site that exists in the site collection in which the feature is marked active.

@FeatureId: The feature identifier of the feature for which the properties are updated.

@Flags: MUST be 0.

@Properties: An XML fragment, that MUST conform to the XML schema defined in Feature Property Definitions, for the feature. If the *@Properties* parameter is NULL, the Feature Property Definitions MUST be empty.

Return Code Values: An integer that MUST be listed in the following table:

Value	Description
0	Successful completion.
3	The site collection or site does not exist, or the feature is not marked activate in the site collection or site.

Result Sets: MUST NOT return any result sets.

3.1.4.65 proc_UpdateListContentTypes

The **proc_UpdateListContentTypes** stored procedure is called to update the list content types available on a given list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_UpdateListContentTypes(
  @SiteId       uniqueidentifier,
  @WebId        uniqueidentifier,
  @ListId       uniqueidentifier,
  @val          ntext,
  @Version      int
);
```

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

@WebId: The site identifier of the site that contains the requested list (1).

@ListId: The list identifier of the list (1) to be updated.

@val: The XML fragment that defines the list content types for the list (1) specified by *@ListId*. The XML schemas for this structure is defined in [MS-WSSCAML], section TPContentTypeReferences.

@Version: The version of the list (1) to update.

Return Code Values: An integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
3	The requested site or list (1) does not exist.
1150	*@Version* is not the current version of the list (1).

Result Sets: MUST NOT return any result sets.

3.1.4.66 proc_UpdateListFields

The **proc_UpdateListFields** stored procedure is called to add one or more fields (2) to a list (1) or update the field (2) definition of one or more fields (2) in a list (1). The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_UpdateListFields(
  @SiteId                 uniqueidentifier,
  @WebId                  uniqueidentifier,
  @ListId                 uniqueidentifier,
  @Fields                 ntext,
  @ContentTypes           ntext,
  @Version                int,
  @UpdateListFieldsFlags  int = 1,
  @FieldIdDeleted         uniqueidentifier = NULL,
  @VersionDelta           int = 1
);
```

@SiteId: The site collection identifier in which the list (1) exists.

@WebId: The site identifier of the site containing the list (1).

@ListId: The list identifier of the specified list (1).

@Fields: The updated **string** of implementation-specific version number followed by an XML fragment of the field (2) definitions. The XML schema for this structure is defined in [MS-WSSCAML] section TPFields.

@ContentTypes: The updated XML fragment of the content types used by the specified list (1). The content types **XML** is not modified if this is NULL. The XML schema for this structure is defined in [MS-WSSCAML] section TPContentTypeReferences.

@Version: The version of the lists metadata.

@UpdateListFieldsFlags: An **integer** value that MUST consists one or more bit flags defined in the following table<1>

Value	Description
"1"	Indicates if the field (2) schema for this list (1) is modified or not. If this flag is set, updates the list flags (see [MS-WSSFO2] section 2.2.2.5) for the list (1) specified by *@ListId,* indicating that this list (1) has had its schema customized (2) from the version that exists in the schema file stored on the front-end Web server that was used to create it.
2	Indicates the list field (2) schema has been updated, and the only change is the build number in the version **string**. The format of the list field (2) schema is defined in MS-WSSCAML section 2.4.1.12

@FieldIdDeleted: This parameter MUST be NULL.

@VersionDelta: The **integer** value by which the version of list metadata SHOULD be incremented.

Return Code Values: An **integer** that MUST be listed in the following table:

Value	Description
"0"	Successful completion.
"1150"	A concurrency violation occurred. The version specified by *@Version* does not exist for the list (1) specified by *@ListId*.

Result Sets: MUST NOT return any result sets.

3.1.4.67 proc_UpdateSiteHashKey

The **proc_UpdateSiteHashKey** stored procedure is called to update the random set of bytes used to generate the **form digest validation** of a site collection. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_UpdateSiteHashKey(
  @SiteId           uniqueidentifier,
  @SiteHashKey      binary(16)
);
```

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

@SiteHashKey: A random set of 16 bytes that will be used to generate the form digest validation. MUST NOT be NULL.

Return Code Values: An integer value of 0, which the protocol client MUST ignore.

Result Sets: MUST NOT return any result sets.

3.1.4.68 proc_UpdateTpWebMetaData

The **proc_UpdateTpWebMetaData** stored procedure is called to update metadata for an existing site. The T-SQL syntax for the stored procedure is as follows:

```
PROCEDURE proc_UpdateTpWebMetaData(
  @SiteId                     uniqueidentifier,
  @WebId                      uniqueidentifier,
  @Title                      nvarchar(255),
  @Description                ntext,
  @Version                    int,
  @WebTemplate                int,
  @Language                   int,
  @Locale                     int,
  @Collation                  smallint,
  @TimeZone                   smallint,
  @Time24                     bit,
  @CalendarType               smallint,
  @AdjustHijriDays            smallint,
  @AltCalendarType            tinyint,
  @CalendarViewOptions        tinyint,
  @WorkDays                   smallint,
  @WorkDayStartHour           smallint,
  @WorkDayEndHour             smallint,
  @Config                     smallint,
  @Flags                      int,
  @Author                     int,
  @AlternateCSSUrl            nvarchar(260),
  @CustomizedCss              nvarchar(260),
  @CustomJSUrl                nvarchar(260),
  @AlternateHeaderUrl         nvarchar(260),
  @ExternalSecurityProvider   uniqueidentifier,
  @MasterUrl                  nvarchar(260),
  @CustomMasterUrl            nvarchar(260),
  @SiteLogoUrl                nvarchar(260),
  @SiteLogoDescription        nvarchar(255),
  @TemplateVersion            smallint,
  @TimeCreated                datetime,
  @TimeLastModified           datetime
);
```

@SiteId: The site collection identifier for the site collection containing the site

@WebId: The site identifier for the site whose metadata is to be updated.

@Title: The title for the site. If the value is NULL, then the existing value is not updated.

@Description: The description of the site. If the value is NULL, then the existing value is not updated.

@Version: The version of the metadata for this site, before this update. If the caller does not specify the correct version, the update will fail.

@WebTemplate: The identifier for the site definition used in the site definition to define the base structure of this site. If the value is -1, then the existing value is not updated.

@Language: The Language Code Identifier (LCID) associated with the site. This specifies the current UI culture, which determines the language resources used to display text and images to the user of the front-end Web server. If the value is 0, then the existing value is not updated.

@Locale: The language code identifier (LCID) associated with the site which is used to determine the current culture for regional language specific data formatting such as currency or date/time settings. If the value is 0, then the existing value is not updated.

@Collation: The collation order of the site which indicates an additional sorting order that should be processed by the back-end database server. The collation method is an implementation-specific capability of the front-end Web server and back-end database server. If the value is -1, then the existing value is not updated.

@TimeZone: The Time Zone identifier for the time zone that SHOULD be used when displaying time values for this site**.** If the value is -1, then the existing value is not updated.

@Time24: If set to 1, a 24-hour time format SHOULD be used when displaying time values for this site; otherwise, a 12-hour time format SHOULD be used. If the value is -1 then the existing value is not updated.

@CalendarType: The calendar type that SHOULD be used when processing date values for this site. If the value is -1, then the existing value is not updated. (For more information about a calendar type, see [MS-WSSFO2] section 2.2.3.3 – Calendar Type.)

@AdjustHijriDays: If the CalendarType value is 6, this specifies the number of days to extend or reduce the current month in Hijri calendars for this site.

@AltCalendarType: The calendar type of an alternate calendar for processing date values for this site. If the value is NULL, only the CalendarType value is used for this site. If the value is -1, then the existing value is not updated. (For more information about a calendar type, see [MS-WSSFO2] section 2.2.3.3 – Calendar Type.)

@CalendarViewOptions: A Calendar View Options Type which specifies the calendar display options setting for this site.

@WorkDays: A set of Workdays Flags which specify the week days defined as the work week for this site.

@WorkDayStartHour: The start time of the work day, in minutes after midnight for this site.

@WorkDayEndHour: The end time of the work day, in minutes after midnight for this site

@Config: An identifier of the site definition used to provision this site. If the value is -1, then the existing value is not updated. The following reserved values and implementation-specific site definition identifiers are defined.

Value	Description
-1	This site has not had any site definition provisioned.
0	This site has the implementation-specific default site definition applied.
1	This site has the Team Collaboration site site definition applied.
2	This site has the Meeting Workspace site site definition applied.
3	This site has the Central Administration site site definition applied.

Value	Description
4	This site has the Wiki site site definition applied.
9	This site has the Blog site site definition applied.

@Flags: A **Site Property Flags** value describing the configuration of this site. If the value is NULL, then the existing value is not updated.

@Author: The user identifier of the user who is listed as the creator of this site. If the value is NULL, then the existing value is not updated.

@AlternateCSSUrl: The URL for a custom cascading style sheet file registered on the site for use in pages of the site.

@CustomizedCss: This contains an implementation-specific list of custom cascading style sheet files associated with this site

@CustomJSUrl: The URL for a custom JavaScript file registered on the site for use in pages of the site.

@AlternateHeaderUrl: The URL for a custom header **HTML** page registered on the site for use in pages of the site when rendered on the front-end Web server.

@ExternalSecurityProvider: The CLSID of the external security provider for this site. This MUST be NULL for sites using the native security implementation.

@MasterUrl: The URL of the master page for a given site. If the value is NULL, the existing value is not updated.

@CustomMasterUrl: The URL for an alternate master page for a given site. If the value is NULL, the existing value is not updated.

@SiteLogoUrl: The URL of an image that represents the site, used for display in the user interface.

@SiteLogoDescription: The description of the image that represents the site used for display in the user interface as an ALT tag on the image.

@TemplateVersion: A property that shows the revision of a site definition used in the site definition to define the base structure of the site. If the value is -1, the existing value is not updated.

@TimeCreated: the date and time that the site collection is created. If the value is NULL, then the existing value is not updated. MAY be omitted.

@TimeLastModified: The timestamp in UTC format that specifies the last time the metadata of this site was modified by any user. If *@TimeLastModified* is NULL then the LastMetadataChange timestamp is updated to the system time of the back-end database server.

Return Code Values: an integer which MUST be listed in the following table:

Value	Description
0	Successful completion.
3	The site specified by *@WebId* does not exist

Value	Description
1150	Failed to update, as the value specified by *@Version* does not match the version of the site metadata being updated.

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

No other local events impact the operation of this protocol.

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 requesting 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, but 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 completed as part of a response for a higher level event.

- Configuration
- Site collections
- Sites
- Lists (1)
- 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 procedure(s). 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 procedure(s).

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 Result 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. However, 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 or Configuration databases, 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

No other local events impact the operation of this protocol.

4 Protocol Examples

This section provides protocol examples for using features, content types and columns, and views.

4.1 Features

This section includes examples that show how to activate and deactivate a feature at a site.

4.1.1 Activate a Feature at a Site

This scenario is initiated when a feature is activated for a site.

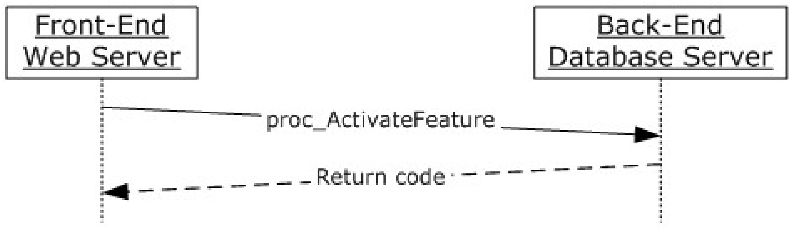

Figure 2: Activate a Feature at a Site

For simplicity's sake, this example assumes that:

1. The scope of the feature is that of a site.
2. The feature does not have any activation dependencies.

The following actions happen:

1. The front-end Web server attempts to activate the specified feature by calling the proc_ActivateFeature stored procedure.
2. The back-end database server returns an output Return Code indicating the result of the action.

4.1.2 Deactivate a Feature at a Site

This scenario is initiated when a feature is deactivated for a site.

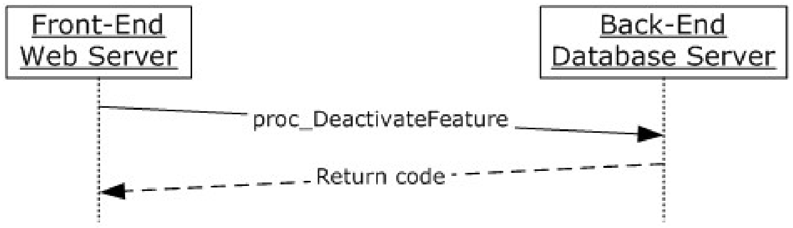

Figure 3: Deactivate a Feature at a Site

For simplicity's sake, this example assumes that:

1. The scope of the feature is that of a site.
2. The feature does not have any activation dependencies.

The following actions happen:

- The front-end Web server attempts to deactivate the specified feature by calling the proc_DeactivateFeature stored procedure.

The back-end database server returns an output Return Code indicating the result of the action.

4.2 Content Types and Columns

This section provides specific example scenarios for end-to-end content types and columns management in the back-end database server. These examples describe in detail the process of communication between the various server components involved. In conjunction with the detailed protocol documentation, this information is intended to provide an example of how a front-end Web server communicates with a back-end database server.

4.2.1 Create, Rename, and Delete a Text Column

Scenario 1: Jonathan is getting together a team for a 24 hour skiing event. Based on his experience from the previous year, he realized that snowboarders are more competitive than skiers. To keep track of the participants he created a list (1) on a site called "Ski Team". Jonathan created a "snowboarder" site content type and added it to the "Ski Team" list. The following communication between front-end Web server and the back-end database server illustrates an example of the communication that takes place:

1. The front-end Web server calls the [MS-WSSFO2] section 3.1.5.2 - **proc_CreateDir** (defined in [MS-WSSFO2] section 3.1.5.8) stored procedure with the following parameters to create a new directory under 'Lists/Ski Team' called 'snowboarder':

```
EXEC proc_CreateDir
    @DirSiteId = '59E2191D-FCA4-4061-8A2C-7EDB5FC9FECD',
    @DirWebId = 'DD1F4F41-8CF7-4F78-B41D-A3D5836900DA',
    @DirDirName = N'Lists/Ski Team',
    @DirLeafName = N'snowboarder',
    @DirLevel = 1,
    @AddMinorVersion = 0,
    @DocFlags = 0,
    @CreateDirFlags = 16;
```

2. The back-end database server returns a Return Code of 0, indicating success and no result sets.

3. The front-end Web server calls proc_UpdateListContentTypes stored procedure with the following parameters to update the list content types XML fragment for the "Ski Team" list:

```
EXEC proc_UpdateListContentTypes
    @SiteId = '59E2191D-FCA4-4061-8A2C-7EDB5FC9FECD',
    @WebId = 'DD1F4F41-8CF7-4F78-B41D-A3D5836900DA',
    @ListId = '44E6723A-9894-4763-9B7D-27210B80B73D',
@val = N'<ContentType ID="0x0100AFC3898D5EA38D4E83884A182F5AD5E7" Name="Item"
Group="List Content Types" Description="Create a new list item." Version="0"
FeatureId="{695b6570-a48b-4a8e-8ea5-26ea7fc1d162}"><Folder
TargetName="Item"/><FieldRefs><FieldRef ID="{c042a256-787d-4a6f-8a8a-cf6ab767f12d}"
Name="ContentType"/><FieldRef ID="{fa564e0f-0c70-4ab9-b863-0177e6ddd247}" Name="Title"
Required="TRUE" ShowInNewForm="TRUE"
ShowInEditForm="TRUE"/></FieldRefs><XmlDocuments><XmlDocument
NamespaceURI="http://schemas.microsoft.com/sharepoint/v3/contenttype/forms">PEZvcm1UZW
1wbGF0ZXMgeG1sbnM9Imh0dHA6Ly9zY2hlbWFzLm1pY3Jvc29mdC5jb20vc2hhcmVwb2ludC92My9jb250ZW50
dHlwZS9mb3JtcyI+PERpc3BsYXk+TGlzdEZvcm08L0Rpc3BsYXk+PEVkaXQ+TGlzdEZvcm08L0VkaXQ+PE5ldz
```

```
5MaXN0Rm9ybTwvTmV3PjwvRm9ybVRlbXBsYXRlcz4=</XmlDocument></XmlDocuments></ContentType><
ContentType ID="0x01200087D0207666989447AC0E70B27EDEE926" Name="Folder" Group="Folder
Content Types" Description="Create a new folder." Sealed="TRUE" Version="0"
FeatureId="{695b6570-a48b-4a8e-8ea5-26ea7fc1d162}"><FieldRefs><FieldRef ID="{c042a256-
787d-4a6f-8a8a-cf6ab767f12d}" Name="ContentType"/><FieldRef ID="{fa564e0f-0c70-4ab9-
b863-0177e6ddd247}" Name="Title" Required="FALSE" Hidden="TRUE"/><FieldRef
ID="{8553196d-ec8d-4564-9861-3dbe931050c8}" Name="FileLeafRef" Required="TRUE"
Hidden="FALSE"/></FieldRefs><XmlDocuments><XmlDocument
NamespaceURI="http://schemas.microsoft.com/sharepoint/v3/contenttype/forms">PEZvcm1UZW
1wbGF0ZXMgeG1sbnM9Imh0dHA6Ly9zY2hlbWFzLm1pY3Jvc29mdC5jb20vc2hhcmVwb2ludC92My9jb250ZW50
dHlwZS9mb3JtcyI+PERpc3BsYXk+TGlzdEZvcm08L0Rpc3BsYXk+PEVkaXQ+TGlzdEZvcm08L0VkaXQ+PE5ldz
5MaXN0Rm9ybTwvTmV3PjwvRm9ybVRlbXBsYXRlcz4=</XmlDocument></XmlDocuments></ContentType><
ContentType
ID="0x010030FB45D373D641489EE87FA33528FD4E0078487DD373026C438B946D85BC4F2788"
Name="snowboarder" Group="Custom Content Types" Version="11"><Folder
TargetName="snowboarder"/><FieldRefs><FieldRef ID="{c042a256-787d-4a6f-8a8a-
cf6ab767f12d}" Name="ContentType"/><FieldRef ID="{fa564e0f-0c70-4ab9-b863-
0177e6ddd247}" Name="Title" Required="TRUE" ShowInNewForm="TRUE"
ShowInEditForm="TRUE"/><FieldRef ID="{203fa378-6eb8-4ed9-a4f9-221a4c1fbf46}"
Name="Hobbies" Required="FALSE" Hidden="FALSE" ReadOnly="FALSE" PITarget=""
PrimaryPITarget="" PIAttribute="" PrimaryPIAttribute="" Aggregation=""
Node=""/><FieldRef ID="{1020c8a0-837a-4f1b-baa1-e35aff6da169}"
Name="_Photo"/></FieldRefs><XmlDocuments><XmlDocument
NamespaceURI="http://schemas.microsoft.com/sharepoint/v3/contenttype/forms">PEZvcm1UZW
1wbGF0ZXMgeG1sbnM9Imh0dHA6Ly9zY2hlbWFzLm1pY3Jvc29mdC5jb20vc2hhcmVwb2ludC92My9jb250ZW50
dHlwZS9mb3JtcyI+PERpc3BsYXk+TGlzdEZvcm08L0Rpc3BsYXk+PEVkaXQ+TGlzdEZvcm08L0VkaXQ+PE5ldz
5MaXN0Rm9ybTwvTmV3PjwvRm9ybVRlbXBsYXRlcz4=</XmlDocument></XmlDocuments></ContentType><
ContentType ID="0x01"/><ContentType ID="0x0120"/>',
    @Version = 69;
```

4. The back-end database server returns a Return Code of 0, indicating success and no result sets.

5. The front-end Web server calls proc_MapContentTypeToList stored procedure with the following parameters to associate the "snowboarder" list content type with the "Ski Team" list:

```
EXEC proc_MapContentTypeToList
    @SiteId = '59e2191d-fca4-4061-8a2c-7edb5fc9fecd',
    @WebId = 'dd1f4f41-8cf7-4f78-b41d-a3d5836900da',
    @ListId = '44e6723a-9894-4763-9b7d-27210b80b73d',
    @ContentTypeId =
0x010030FB45D373D641489EE87FA33528FD4E0078487DD373026C438B946D85BC4F2788,
    @Class = 1;
```

6. The back-end database server returns a Return Code of 0, indicating success and no result sets.

7. The front-end Web server calls the proc_UpdateListFields stored procedure with the following parameters to update the list columns in the "Ski Team" list:

```
EXEC proc_UpdateListFields
    @SiteId = '59E2191D-FCA4-4061-8A2C-7EDB5FC9FECD',
    @WebId = 'DD1F4F41-8CF7-4F78-B41D-A3D5836900DA',
    @ListId = '44E6723A-9894-4763-9B7D-27210B80B73D',
    @Fields = N'12.0.0.6219.0.0<FieldRef Name="ContentTypeId"/><FieldRef Name="Title"
ColName="nvarchar1"/><FieldRef Name="_ModerationComments" ColName="ntext1"/><FieldRef
Name="File_x0020_Type" ColName="nvarchar2"/><Field ID="{203FA378-6EB8-4ed9-A4F9-
221A4C1FBF46}" Name="Hobbies" DisplayName="Hobbies" Group="Core Contact and Calendar
Columns" Type="Text" Sealed="TRUE" AllowDeletion="TRUE" Customization=""
SourceID="{44e6723a-9894-4763-9b7d-27210b80b73d}" StaticName="Hobbies"
ColName="nvarchar3" RowOrdinal="0"/><Field Name="_Photo" ID="{1020c8a0-837a-4f1b-baa1-
e35aff6da169}" StaticName="_Photo"
```

```
SourceID="http://schemas.microsoft.com/sharepoint/v3" DisplayName="Contact Photo"
Group="Core Contact and Calendar Columns" Type="URL" Format="Image" Sealed="TRUE"
Sortable="FALSE" AllowDeletion="TRUE" Customization="" ColName="nvarchar4"
RowOrdinal="0" ColName2="nvarchar5" RowOrdinal2="0"/><Field RowOrdinal="0"
Type="Choice" Format="Dropdown" FillInChoice="FALSE" Sealed="FALSE" Name="ContentType"
ColName="tp_ContentType" SourceID="http://schemas.microsoft.com/sharepoint/v3"
ID="{c042a256-787d-4a6f-8a8a-cf6ab767f12d}" DisplayName="Content Type"
StaticName="ContentType" Group="_Hidden" PITarget="MicrosoftWindowsSharePointServices"
PIAttribute="ContentTypeID"><Default>Item</Default><CHOICES><CHOICE>Item</CHOICE><CHOI
CE>Folder</CHOICE><CHOICE>snowboarder</CHOICE></CHOICES></Field>',
    @ContentTypes = NULL,
    @Version = 70;
```

8. The back-end database server returns a Return Code of 0, indicating success and no result sets.

Scenario 2: One of the columns in the "snowboarder" site content type is "Hobbies". Jonathan thought it would be nice to know what else his teammates like besides snowboarding. Jonathan's friend David asked to make the "Hobbies" column a Required field (2). This was done to make sure that no one on the team likes to ski in their spare time from snowboarding. Jonathan agreed that it's a good idea and changed the properties of the "Hobbies" column in the "snowboarder" site content type to make it a Required field (2). The following communication between the front-end Web server and the back-end database server was used to do this:

1. The front-end Web server calls proc_UpdateContentTypeInScope stored procedure with the following parameters to update the XML fragment that defines the "snowboarder" site content type:

```
EXEC proc_UpdateContentTypeInScope
    @SiteId = '59E2191D-FCA4-4061-8A2C-7EDB5FC9FECD',
    @Class = 1,
    @Scope = N'',
    @ContentTypeId = 0x010030FB45D373D641489EE87FA33528FD4E,
    @Version = 15,
    @Definition = N'<ContentType ID="0x010030FB45D373D641489EE87FA33528FD4E"
Name="snowboarder" Group="Custom Content Types" Version="13"><Folder
TargetName="_cts/snowboarder" /><FieldRefs><FieldRef ID="{c042a256-787d-4a6f-8a8a-
cf6ab767f12d}" Name="ContentType" /><FieldRef ID="{fa564e0f-0c70-4ab9-b863-
0177e6ddd247}" Name="Title" Required="TRUE" ShowInNewForm="TRUE" ShowInEditForm="TRUE"
/><FieldRef ID="{203fa378-6eb8-4ed9-a4f9-221a4c1fbf46}" Name="Hobbies" Required="TRUE"
Hidden="FALSE" Customization="" ReadOnly="FALSE" PITarget="" PrimaryPITarget=""
PIAttribute="" PrimaryPIAttribute="" Aggregation="" Node="" /><FieldRef ID="{1020c8a0-
837a-4f1b-baa1-e35aff6da169}" Name="_Photo" /></FieldRefs><XmlDocuments><XmlDocument
NamespaceURI="http://schemas.microsoft.com/sharepoint/v3/contenttype/forms">PEZvcm1UZW
1wbGF0ZXMgeG1sbnM9Imh0dHA6Ly9zY2hlbWFzLm1pY3Jvc29mdC5jb20vc2hhcmVwb2ludC9yb250ZW50
dHlwZS9mb3JtcyI+PERpc3BsYXk+TGlzdEZvcm08L0Rpc3BsYXk+PEVkaXQ+TGlzdEZvcm08L0VkaXQ+PE5ldz
5MaXN0Rm9ybTwvTmV3PjwvRm9ybVRlbXBsYXRlcz4=</XmlDocument></XmlDocuments></ContentType>'
;
```

2. The back-end database server returns a Return Code of 0, indicating success and no result sets.

3. The front-end Web server calls the proc_UnmapFieldsFromContentType stored procedure with the following parameters to remove the association between the "snowboarder" site content type and all site columns that it uses.

```
EXEC proc_UnmapFieldsFromContentType
    @SiteId = '59e2191d-fca4-4061-8a2c-7edb5fc9fecd'
    @WebId = 'dd1f4f41-8cf7-4f78-b41d-a3d5836900da',
    @ContentTypeId = 0x010030FB45D373D641489EE87FA33528FD4E;
```

4. The back-end database server returns a Return Code of 0, indicating success and no result sets.

5. The front-end Web server calls the proc_MapFieldToContentType stored procedure four times as follows to add an association between the "snowboarder" site content type and each of the four site columns that it uses.

```
EXEC proc_MapFieldToContentType
    @SiteId = '59e2191d-fca4-4061-8a2c-7edb5fc9fecd',
    @WebId = 'dd1f4f41-8cf7-4f78-b41d-a3d5836900da',
    @ContentTypeId = 0x010030FB45D373D641489EE87FA33528FD4E,
    @FieldId = 'c042a256-787d-4a6f-8a8a-cf6ab767f12d';
EXEC proc_MapFieldToContentType
    @SiteId = '59e2191d-fca4-4061-8a2c-7edb5fc9fecd',
    @WebId = 'dd1f4f41-8cf7-4f78-b41d-a3d5836900da',
    @ContentTypeId = 0x010030FB45D373D641489EE87FA33528FD4E,
    @FieldId = 'fa564e0f-0c70-4ab9-b863-0177e6ddd247';
EXEC proc_MapFieldToContentType
    @SiteId = '59e2191d-fca4-4061-8a2c-7edb5fc9fecd',
    @WebId = 'dd1f4f41-8cf7-4f78-b41d-a3d5836900da',
    @ContentTypeId = 0x010030FB45D373D641489EE87FA33528FD4E,
    @FieldId = '203fa378-6eb8-4ed9-a4f9-221a4c1fbf46';
EXEC proc_MapFieldToContentType
    @SiteId = '59e2191d-fca4-4061-8a2c-7edb5fc9fecd',
    @WebId = 'dd1f4f41-8cf7-4f78-b41d-a3d5836900da',
    @ContentTypeId = 0x010030FB45D373D641489EE87FA33528FD4E,
    @FieldId = '1020c8a0-837a-4f1b-baa1-e35aff6da169';
```

6. The back-end database server returns a Return Code of 0, indicating success and no result sets each of the four times.

7. The front-end Web server calls the proc_ListDerivedContentTypes stored procedure with the following parameters to fetch all content types that inherit from the "snowboarder" site content type in order to propagate the change.

```
EXEC proc_ListDerivedContentTypes
    @SiteId = '59E2191D-FCA4-4061-8A2C-7EDB5FC9FECD',
    @ContentTypeId = 0x010030FB45D373D641489EE87FA33528FD4E;
```

8. The back-end database server returns a Return Code of 0, indicating success and two result sets including the Derived List Content Types result set, which contains the content type identifier of the "snowboarder" list content type used in "Ski Team" list.

ContentTypeId	WebId	ListId
0x010030FB45D373D641489EE87FA33528FD4E0078487DD373026C438B946D85BC4F2788	DD1F4F41-8CF7-4F78-B41D-A3D5836900DA	44E6723A-9894-4763-9B7D-27210B80B73D

1. The front-end Web server calls the proc_UpdateListContentTypes stored procedure with the following parameters to update the list content types XML fragment for the "Ski Team" list:

```
EXEC proc_UpdateListContentTypes
    @SiteId = '59E2191D-FCA4-4061-8A2C-7EDB5FC9FECD',
    @WebId = 'DD1F4F41-8CF7-4F78-B41D-A3D5836900DA',
    @ListId = '44E6723A-9894-4763-9B7D-27210B80B73D',
    @val = N'<ContentType ID="0x0100AFC3898D5EA38D4E83884A182F5AD5E7" Name="Item" 
Group="List Content Types" Description="Create a new list item." Version="0" 
FeatureId="{695b6570-a48b-4a8e-8ea5-26ea7fc1d162}"><Folder 
TargetName="Item"/><FieldRefs><FieldRef ID="{c042a256-787d-4a6f-8a8a-cf6ab767f12d}" 
Name="ContentType"/><FieldRef ID="{fa564e0f-0c70-4ab9-b863-0177e6ddd247}" Name="Title" 
Required="TRUE" ShowInNewForm="TRUE" 
ShowInEditForm="TRUE"/></FieldRefs><XmlDocuments><XmlDocument 
NamespaceURI="http://schemas.microsoft.com/sharepoint/v3/contenttype/forms">PEZvcm1UZW
1wbGF0ZXMgeG1sbnM9Imh0dHA6Ly9zY2hlbWFzLm1pY3Jvc29mdC5jb20vc2hhcmVwb2ludC92My9jb250ZW50
dHlwZS9mb3JtcyI+PERpc3BsYXk+TGlzdEZvcm08L0Rpc3BsYXk+PEVkaXQ+TGlzdEZvcm08L0VkaXQ+PE5ldz
5MaXN0Rm9ybTwvTmV3PjwvRm9ybVRlbXBsYXRlcz4=</XmlDocument></XmlDocuments></ContentType><
ContentType ID="0x01200087D0207666989447AC0E70B27EDEE926" Name="Folder" Group="Folder 
Content Types" Description="Create a new folder." Sealed="TRUE" Version="0" 
FeatureId="{695b6570-a48b-4a8e-8ea5-26ea7fc1d162}"><FieldRefs><FieldRef ID="{c042a256-
787d-4a6f-8a8a-cf6ab767f12d}" Name="ContentType"/><FieldRef ID="{fa564e0f-0c70-4ab9-
b863-0177e6ddd247}" Name="Title" Required="FALSE" Hidden="TRUE"/><FieldRef 
ID="{8553196d-ec8d-4564-9861-3dbe931050c8}" Name="FileLeafRef" Required="TRUE" 
Hidden="FALSE"/></FieldRefs><XmlDocuments><XmlDocument 
NamespaceURI="http://schemas.microsoft.com/sharepoint/v3/contenttype/forms">PEZvcm1UZW
1wbGF0ZXMgeG1sbnM9Imh0dHA6Ly9zY2hlbWFzLm1pY3Jvc29mdC5jb20vc2hhcmVwb2ludC92My9jb250ZW50
dHlwZS9mb3JtcyI+PERpc3BsYXk+TGlzdEZvcm08L0Rpc3BsYXk+PEVkaXQ+TGlzdEZvcm08L0VkaXQ+PE5ldz
5MaXN0Rm9ybTwvTmV3PjwvRm9ybVRlbXBsYXRlcz4=</XmlDocument></XmlDocuments></ContentType><
ContentType 
ID="0x010030FB45D373D641489EE87FA33528FD4E0078487DD373026C438B946D85BC4F2788" 
Name="snowboarder" Group="Custom Content Types" Version="12"><Folder 
TargetName="snowboarder"/><FieldRefs><FieldRef ID="{c042a256-787d-4a6f-8a8a-
cf6ab767f12d}" Name="ContentType"/><FieldRef ID="{fa564e0f-0c70-4ab9-b863-
0177e6ddd247}" Name="Title" Required="TRUE" ShowInNewForm="TRUE" 
ShowInEditForm="TRUE"/><FieldRef ID="{203fa378-6eb8-4ed9-a4f9-221a4c1fbf46}" 
Name="Hobbies" Required="TRUE" Hidden="FALSE" Customization="" ReadOnly="FALSE" 
PITarget="" PrimaryPITarget="" PIAttribute="" PrimaryPIAttribute="" Aggregation="" 
Node=""/><FieldRef ID="{1020c8a0-837a-4f1b-baa1-e35aff6da169}" 
Name="_Photo"/></FieldRefs><XmlDocuments><XmlDocument 
NamespaceURI="http://schemas.microsoft.com/sharepoint/v3/contenttype/forms">PEZvcm1UZW
1wbGF0ZXMgeG1sbnM9Imh0dHA6Ly9zY2hlbWFzLm1pY3Jvc29mdC5jb20vc2hhcmVwb2ludC92My9jb250ZW50
dHlwZS9mb3JtcyI+PERpc3BsYXk+TGlzdEZvcm08L0Rpc3BsYXk+PEVkaXQ+TGlzdEZvcm08L0VkaXQ+PE5ldz
5MaXN0Rm9ybTwvTmV3PjwvRm9ybVRlbXBsYXRlcz4=</XmlDocument></XmlDocuments></ContentType><
ContentType ID="0x01"/><ContentType ID="0x0120"/>',
    @Version = 71;
```

2. The back-end database server returns a Return Code of 0, indicating success and no result sets.

3. The front-end Web server calls the proc_UpdateListFields stored procedure with the following parameters to update the list columns in the "Ski Team" list:

```
EXEC proc_UpdateListFields
    @SiteId = '59E2191D-FCA4-4061-8A2C-7EDB5FC9FECD',
    @WebId = 'DD1F4F41-8CF7-4F78-B41D-A3D5836900DA',
    @ListId = '44E6723A-9894-4763-9B7D-27210B80B73D',
    @Fields = N'12.0.0.6219.0.0<FieldRef Name="ContentTypeId"/><FieldRef Name="Title" 
ColName="nvarchar1"/><FieldRef Name="_ModerationComments" ColName="ntext1"/><FieldRef 
Name="File_x0020_Type" ColName="nvarchar2"/><Field ID="{203FA378-6EB8-4ed9-A4F9-
221A4C1FBF46}" Name="Hobbies" DisplayName="Hobbies" Group="Core Contact and Calendar 
Columns" Type="Text" Sealed="TRUE" AllowDeletion="TRUE" Customization="" 
SourceID="{44e6723a-9894-4763-9b7d-27210b80b73d}" StaticName="Hobbies" 
ColName="nvarchar3" RowOrdinal="0"/><Field Name="_Photo" ID="{1020c8a0-837a-4f1b-baa1-
e35aff6da169}" StaticName="_Photo" 
SourceID="http://schemas.microsoft.com/sharepoint/v3" DisplayName="Contact Photo" 
Group="Core Contact and Calendar Columns" Type="URL" Format="Image" Sealed="TRUE" 
Sortable="FALSE" AllowDeletion="TRUE" Customization="" ColName="nvarchar4"
```

```
RowOrdinal="0" ColName2="nvarchar5" RowOrdinal2="0"/><Field RowOrdinal="0"
Type="Choice" Format="Dropdown" FillInChoice="FALSE" Sealed="FALSE" Name="ContentType"
ColName="tp_ContentType" SourceID="http://schemas.microsoft.com/sharepoint/v3"
ID="{c042a256-787d-4a6f-8a8a-cf6ab767f12d}" DisplayName="Content Type"
StaticName="ContentType" Group="_Hidden" PITarget="MicrosoftWindowsSharePointServices"
PIAttribute="ContentTypeID"><Default>Item</Default><CHOICES><CHOICE>Item</CHOICE><CHOI
CE>Folder</CHOICE><CHOICE>snowboarder</CHOICE></CHOICES></Field>',
    @ContentTypes = NULL,
    @Version = 72;
```

4. The back-end database server returns a return code of 0, indicating success and no result sets.

4.2.2 Create a Text Site Column

This scenario is initiated when a site column is created for a site.

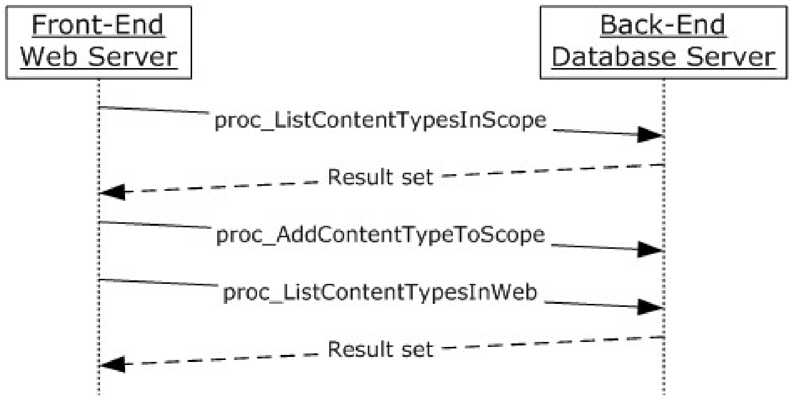
Front-End Web Server
Back-End Database Server
proc_ListContentTypesInScope
Result set
proc_AddContentTypeToScope
proc_ListContentTypesInWeb
Result set

Figure 4: Create a Text Site Column

The following actions happen:

1. The front-end Web server queries the site columns information by calling the proc_ListContentTypesInScope stored procedure.
2. The back-end database server returns result sets as listed in proc_ListContentTypesInScope
3. The front-end Web server calls proc_AddContentTypeToScope to add a new site column to the site.
4. The front-end Web server calls proc_ListContentTypesInWeb to query the site columns information for the specified site
5. The back-end database server returns 1 result set containing the requested information as listed in proc_ListContentTypesInWeb.

4.2.3 Add a Site Column to a List

This scenario is initiated when a site column is added to a list (1).

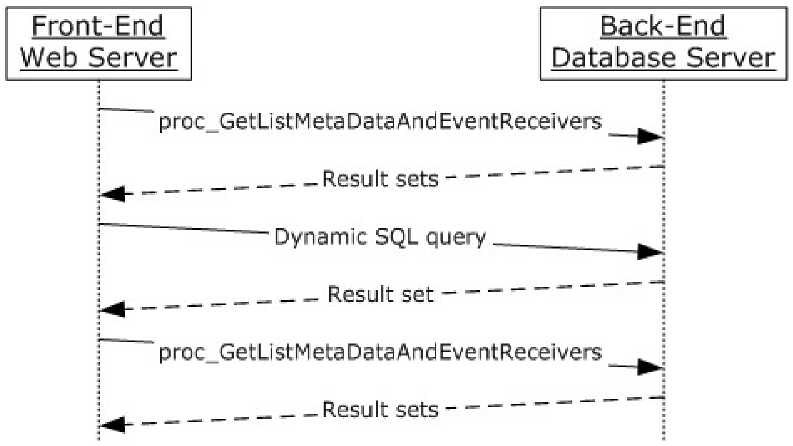

Figure 5: Add a Site Column to a list

For simplicity's sake, this example assumes that the site column being added to the list is contained in the site.

The following actions happen:

1. The front-end Web server queries all MetaData information and event receivers for the specified list by calling the **proc_GetListMetaDataAndEventReceivers** (defined in [MS-WSSFO2] section 3.1.5.26) stored procedure.
2. The back-end database server returns result sets as listed in [MS-WSSFO2] section 3.1.5.28.1 – [MS-WSSFO2] section 3.1.4.28.5.
3. The front-end Web server builds a transactional dynamic **query** in SQL syntax to add the site column to the list.
    1. The query begins a new **transaction (2)**.
    2. The query attempts to add the site column to the specified list using the proc_UpdateListFields stored procedure.
    3. The query then attempts to record that the site column is being used in the specified list.
    4. The query rolls back the transaction if the previous actions were not successful, or it commits the transaction if they were successful.
4. The front-end Web server queries all MetaData information and event receivers for the specified list by calling the **proc_GetListMetaDataAndEventReceivers** (defined in [MS-WSSFO2] section 3.1.5.26) stored procedure.
5. The back-end database server returns result sets as listed in [MS-WSSFO2] section 3.1.5.28.1 – [MS-WSSFO2] section 3.1.4.28.5.

4.2.4 Change the Name of a Site Column and Propagate to Lists

This scenario is initiated when the display name of a site column is changed and the change is pushed to lists (1).

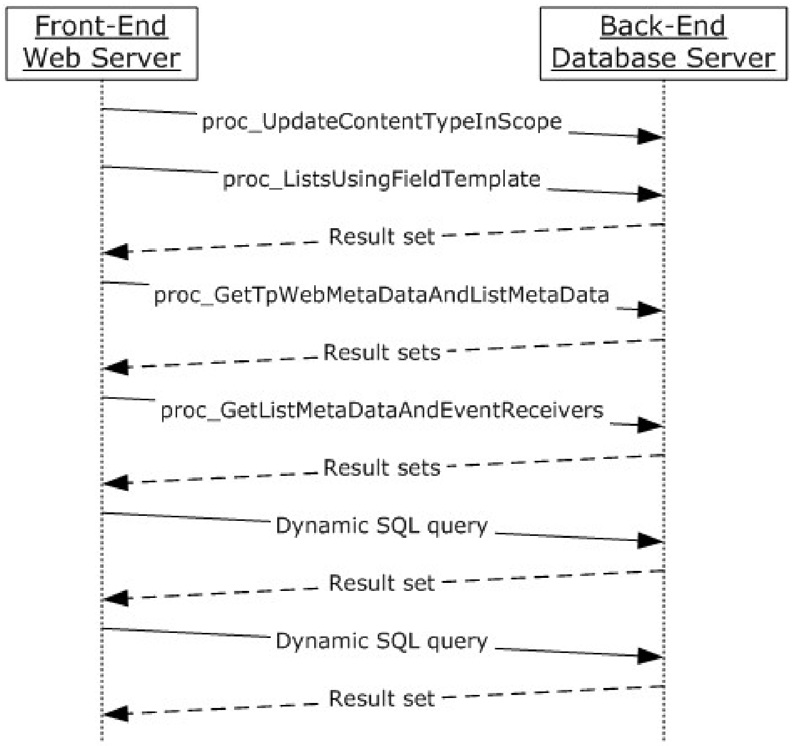
Front-End Web Server
Back-End Database Server
proc_UpdateContentTypeInScope
proc_ListsUsingFieldTemplate
Result set
proc_GetTpWebMetaDataAndListMetaData
Result sets
proc_GetListMetaDataAndEventReceivers
Result sets
Dynamic SQL query
Result set
Dynamic SQL query
Result set

Figure 6: Change the Name of a Site Column and Propagate to Lists

For simplicity's sake, this example assumes that:

1. The site column display name being changed is contained in the site.
2. The lists which the change of the site column display name are pushed to are contained in the site

The following actions happen:

1. The front-end Web server updates the definition (the display name in this scenario) of the site column by calling the proc_UpdateContentTypeInScope stored procedure.
2. The front-end Web server queries for a list of lists in the site which include the specified site column by calling the proc_ListsUsingFieldTemplate stored procedure.
3. The back-end database server returns result sets as listed in proc_ListsUsingFieldTemplate.
4. The front-end Web server queries for the MetaData for the specified site by calling the **proc_GetTpWebMetaDataAndListMetaData** (defined in [MS-WSSFO2] section 3.1.5.35) stored procedure.
5. The back-end database server returns result sets as listed in [MS-WSSFO2] section 3.1.5.35.1 – [MS-WSSFO2] section 3.1.4.35.27.
6. The front-end Web server queries all MetaData information and event receivers for the specified list by calling the **proc_GetListMetaDataAndEventReceivers** (defined in [MS-WSSFO2] section 3.1.5.26) stored procedure.

7. The back-end database server returns result sets as listed in [MS-WSSFO2] section 3.1.5.28.1 - [MS-WSSFO2] section 3.1.4.28.5.

8. The front-end Web server builds a transactional dynamic query in SQL syntax to update the definition (the display name in this scenario) of the field (2) in the list.

   - The query begins a new transaction.

   - The query attempts to update the field (2) definition of fields (2) in the specified list using the proc_UpdateListFields stored procedure.

   - The query rolls back the transaction if the previous actions were not successful, or it commits the transaction if they were successful.

9. The front-end Web server builds a transactional dynamic query in SQL syntax to update the list content types on the specified list.

10. The query begins a new transaction.

11. The query attempts to update the list content types on the specified list using the proc_UpdateListContentTypes stored procedure.

12. The query rolls back the transaction if the previous actions were not successful, or it commits the transaction if they were successful.

4.2.5 Create a New Site Content Type

This scenario is initiated when a new site content type is created.

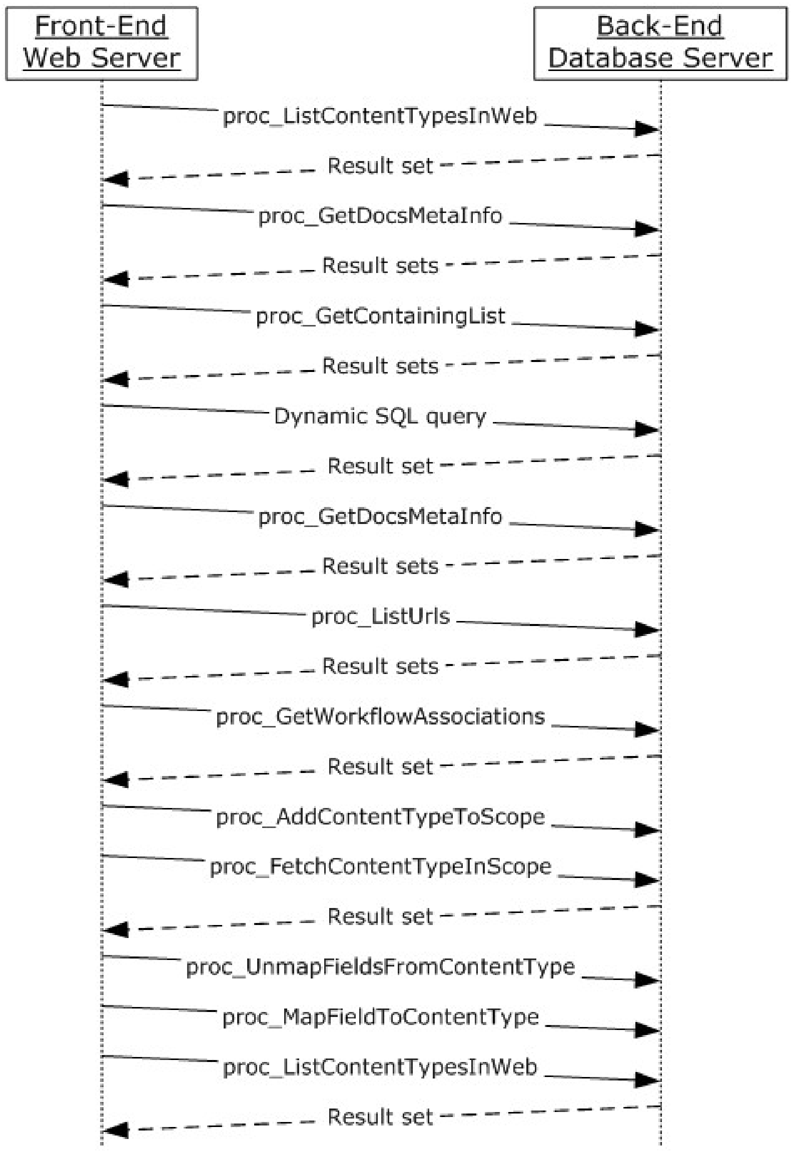

Figure 7: Create a new Site Content Type

For simplicity's sake, this example assumes that:

- The parent content type of the newly created content type is contained in the site.

The following actions happen:

1. The front-end Web server calls proc_ListContentTypesInWeb to query the content types information for the specified site
2. The back-end database server returns 1 result set containing the requested information as listed in proc_ListContentTypesInWeb.

3. The front-end Web server calls **proc_GetDocsMetaInfo** (defined in [MS-WSSFO2] section 3.1.5.24) to retrieve the metadata information for the specified content type

4. The back-end database server returns result sets containing the requested information as listed in [MS-WSSFO2] section 3.1.5.24.1 - [MS-WSSFO2] section 3.1.4.24.6.

5. The front-end Web server calls [MS-WSSFO2] section 3.1.5.19 - proc_GetContainingList to retrieve the metadata and event receiver information for the specified content type

6. The back-end database server returns result sets containing the requested information as listed in [MS-WSSFO2] section 3.1.4.19.1 - [MS-WSSFO2] section 3.1.4.19.3.

7. The front-end Web server builds a transactional dynamic query in SQL syntax to create a directory for the content types on the specified site.

   ▪The query begins a new transaction.

   ▪The query attempts to create a directory for the content types on the specified site using the [MS-WSSFO2] section 3.1.5.2 - **proc_CreateDir** (defined in [MS-WSSFO2] section 3.1.5.8) stored procedure.

   ▪The query rolls back the transaction if the previous actions were not successful, or it commits the transaction if they were successful.

8. The front-end Web server calls [MS-WSSFO2] section 3.1.5.20 - **proc_GetDocsMetaInfo** (defined in [MS-WSSFO2] section 3.1.5.24) to retrieve the metadata information for the specified content type

9. The back-end database server returns result sets containing the requested information as listed in [MS-WSSFO2] section 3.1.5.20.1 - [MS-WSSFO2] section 3.1.4.20.6.

10. The front-end Web server calls [MS-WSSFO2] section 3.1.5.37 - proc_ListUrls to query the metadata information for the specified site content type

11. The back-end database server returns result sets containing the requested information as listed in [MS-WSSFO2] section 3.1.5.37.1 - [MS-WSSFO2] section 3.1.5.37.7.

12. The front-end Web server calls **proc_GetWorkflowAssociations** (defined in [MS-WSSPROG] section 3.1.4.46) to retrieve the workflow associations information for the specified content type

13. The back-end database server returns result sets containing the requested information as listed in [MS-WSSPROG] section 3.1.4.46.1

14. The front-end Web server calls proc_AddContentTypeToScope to add the newly created content type to the specified site

15. The front-end Web server calls proc_FetchContentTypeInScope to retrieve information about the specified site content type registered to the specified site.

16. The back-end database server returns result sets containing the requested information as listed in proc_FetchContentTypeInScope

17. The front-end Web server calls proc_UnmapFieldsFromContentType to remove existing site columns reference from the specified site content type.

18. The front-end Web server calls proc_MapFieldToContentType to add site a column reference to the specified site content type.

19.The front-end Web server calls proc_ListContentTypesInWeb to query the content types information for the specified site

20.The back-end database server returns 1 result set containing the requested information as listed in proc_ListContentTypesInWeb.

4.2.6 Add Site Column to Content Type

This scenario is initiated when a site column is added to a site content type.

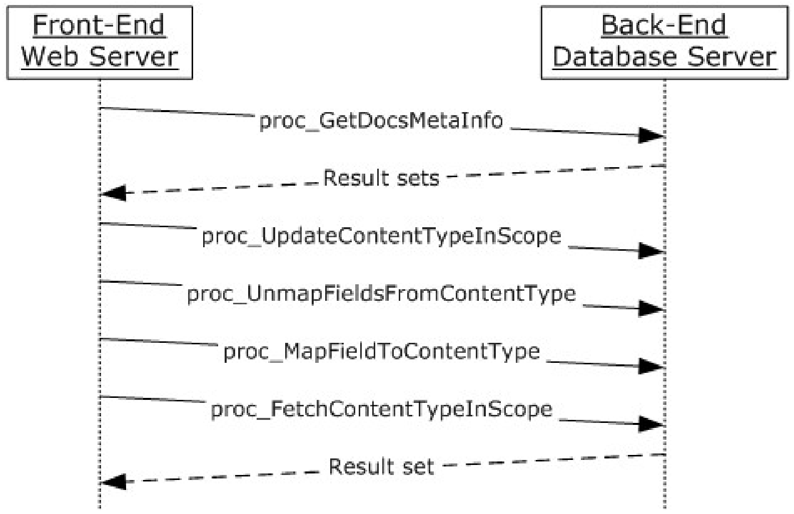
Front-End Web Server
Back-End Database Server
proc_GetDocsMetaInfo
Result sets
proc_UpdateContentTypeInScope
proc_UnmapFieldsFromContentType
proc_MapFieldToContentType
proc_FetchContentTypeInScope
Result set

Figure 8: Add Site Column to Content Type

For simplicity's sake, this example assumes that:

1. The site column to be added to the content type is contained in the site.

2. The content type which the site column is added to is contained in the site

The following actions happen:

1. The front-end Web server calls [MS-WSSFO2] section 3.1.5.20 - **proc_GetDocsMetaInfo** (defined in [MS-WSSFO2] section 3.1.5.24) to retrieve the metadata information for the specified content type

2. The back-end database server returns result sets containing the requested information as listed in [MS-WSSFO2] section 3.1.5.20.1 - [MS-WSSFO2] section 3.1.4.20.6.

3. The front-end Web server updates the definition of the site content type by calling the proc_UpdateContentTypeInScope stored procedure

4. The front-end Web server calls proc_UnmapFieldsFromContentType to remove existing site columns reference from the specified site content type.

5. The front-end Web server calls proc_MapFieldToContentType to add a site column reference to the specified site content type.

6. The front-end Web server calls proc_FetchContentTypeInScope to retrieve information about the specified site content type registered to the specified site.

7. The back-end database server returns result sets containing the requested information as listed in proc_FetchContentTypeInScope.

4.3 Views

Take for example a list (1) that contains list items. Often times, there are several different views created for a list to aggregate the list items in varying ways. For example, an implementer may create a view to display the list items in alphabetical order based on the list item's creator field (2). In addition, another view may be created to display only those list items whose creator field (2) is equal to "Contoso Managers". With multiple views per list, a dilemma arises such that the implementer needs to make a decision as to what view the end user will see when viewing the list items. The solution to this is as follows:

1. After provisioning a list and its views, call proc_MakeViewDefaultForList to set one of the views as the default list view. Use this default list view as the view that will be displayed to all end users when viewing list items.
2. The implementer also gives the end user the option to change the default list view. Once the end user selects which view among those views provisioned to become the default list view, proc_MakeViewDefaultForList is called again to change the default list view. Note that the implementer can use proc_FetchDocForHttpGet stored procedure defined in [MS-WSSFO2] to obtain the views defined for the given list. It is also possible to use proc_MapUrlToListAndView to obtain a specific view for the given list.

A similar dilemma arises with a list's forms. A list can have multiple forms to create, display, and change list items. For example, a list may have multiple edit forms and an implementer must have a mechanism to set one of the edit forms as the default form to be used when a list item is being edited (or changed) by the user. The solution to this is as follows:

1. After provisioning a list and its forms, call proc_SetListFormToUrl to designate which form is the default form for the edit form, new form, and display form.
2. If, at any time, a new edit form, new form, or display form is created, the implementer can call the proc_SetListFormToUrl stored procedure to designate the form as the default form.

5 Security

5.1 Security Considerations for Implementers

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

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 access rights, access will be denied when attempting to set up the [MS-TDS] connection to the content database, or when calling the stored procedures.

5.2 Index of Security Parameters

None.

6 Appendix A: Product Behavior

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

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

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

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

<1> Section 3.1.4.66: Section 3.1.4.66: Prior to the Infrastructure Update for Windows SharePoint Services 3.0 (KB951695), the parameter UpdateListFieldsFlags MUST NOT be used. Instead, this stored procedure uses a parameter *FieldSchemaModified*, which is a bit value indicating whether the field schema for this list has been modified or not.

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