Intellectual Property Rights Notice for Open Specifications Documentation

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

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

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

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

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

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

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

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

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

**Revision Summary**

<table>
<thead>
<tr>
<th>Date</th>
<th>Revision History</th>
<th>Revision Class</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/20/2012</td>
<td>0.1</td>
<td>New</td>
<td>Released new document.</td>
</tr>
<tr>
<td>04/11/2012</td>
<td>0.1</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
<tr>
<td>07/16/2012</td>
<td>0.1</td>
<td>No change</td>
<td>No changes to the meaning, language, or formatting of the technical content.</td>
</tr>
</tbody>
</table>
# Table of Contents

## 1 Introduction ................................. 9
1.1 Glossary ........................................ 9
1.2 References ....................................... 12
  1.2.1 Normative References .................. 12
  1.2.2 Informative References ................. 13
1.3 Overview ......................................... 13
  1.3.1 Content Types .............................. 13
    1.3.1.1 List Content Type Overview .......... 13
    1.3.1.2 Site Content Type Overview .......... 14
  1.3.2 Features ..................................... 14
  1.3.3 Custom Actions ............................ 14
  1.3.4 Views ........................................ 14
  1.3.5 List Schema .................................. 14
    1.3.5.1 List Column Overview ................. 15
    1.3.5.2 Site Column Overview ................. 15
  1.3.6 List/Web MetaInfo ......................... 15
  1.3.7 File Handling ............................... 15
  1.3.8 Provisioning ............................... 15
1.4 Relationship to Other Protocols ............ 16
1.5 Prerequisites/Preconditions ................. 16
1.6 Applicability Statement ...................... 16
1.7 Versioning and Capability Negotiation .... 16
1.8 Vendor-Extensible Fields .................... 16
1.9 Standards Assignments ....................... 16

## 2 Messages ..................................... 17
2.1 Transport ....................................... 17
2.2 Common Data Types ........................... 17
  2.2.1 Simple Data Types and Enumerations ... 17
    2.2.1.1 Upgrade Type .......................... 17
    2.2.1.2 Upgrade Status ....................... 17
  2.2.2 Bit Fields and Flags Structures ......... 17
  2.2.3 Binary Structures ......................... 17
    2.2.3.1 tContentTypeId ....................... 17
    2.2.3.2 List Identifier Packed Array ........ 18
    2.2.3.3 List Base Type Pattern ............... 19
    2.2.3.4 List Identifier Pattern ............... 19
  2.2.4 Common Result Sets ....................... 22
    2.2.4.1 List Content Types Result Set ....... 22
    2.2.4.2 Site Collection Upgrade Info Result Set 23
  2.2.5 Tables and Views ........................... 24
    2.2.5.1 AllListsAux ....................... .... 24
  2.2.6 XML Structures ............................. 24
    2.2.6.1 Namespaces ............................ 24
    2.2.6.2 Simple Types ......................... 24
    2.2.6.3 Complex Types ....................... 24
      2.2.6.3.1 Feature Property Definitions .... 24
      2.2.6.3.2 Site Health Check Result Message 25
3 Protocol Details ................................................................. 27
  3.1 Common Details .......................................................... 27
  3.2 Back-end Database Server Details .............................. 27
    3.2.1 Abstract Data Model ............................................. 27
      3.2.1.1 Content Types .............................................. 27
        3.2.1.1.1 List Content Type Data Model ..................... 27
        3.2.1.1.2 Site Content Type Data Model ..................... 28
      3.2.1.2 Features ...................................................... 29
      3.2.1.3 Views ........................................................ 29
      3.2.1.4 List Schema ................................................ 30
        3.2.1.4.1 List Column Data Model ............................. 30
        3.2.1.4.2 Site Column Data Model ............................. 30
      3.2.1.5 Provisioning ............................................... 31
      3.2.1.6 Custom Actions ............................................ 32
      3.2.1.7 Site Collection Health Checks ......................... 32
      3.2.1.8 Site Collection Upgrade ................................. 32
    3.2.2 Timers .................................................................. 32
    3.2.3 Initialization ...................................................... 32
    3.2.4 Message Processing Events and Sequencing Rules .... 32
      3.2.4.1 proc_AcquireSiteUpgradeSession .................... 33
      3.2.4.2 proc_ActivateFeature ................................. 33
      3.2.4.3 proc_AddContentTypeToScope ......................... 35
      3.2.4.4 proc_AddOrUpdateCustomAction ....................... 36
        3.2.4.4.1 Add or Update Custom Action Result Set ...... 38
      3.2.4.5 proc_AddToSiteUpgradeQueue .......................... 38
      3.2.4.6 proc_CloneFeature ....................................... 39
      3.2.4.7 proc_CopyResourceDir ................................. 39
      3.2.4.8 proc_CopyResourceDir ................................... 40
      3.2.4.9 proc_DeactivateFeature ............................... 42
      3.2.4.10 proc_DeleteContentTypeInScope ................... 42
      3.2.4.11 proc_DeleteFieldTemplateInScope ................. 44
      3.2.4.12 proc_DropListField .................................... 45
      3.2.4.13 proc_EnumListsWithMetadata .............. 46
        3.2.4.13.1 List Count Result Set .............................. 48
        3.2.4.13.2 List Metadata Result Set ......................... 48
        3.2.4.13.3 List Event Receivers Result Set ................ 49
        3.2.4.13.4 List Permissions Result Set ...................... 49
      3.2.4.14 proc_EnumWebAndSubwebsDTM ....................... 50
        3.2.4.14.1 WebsAndSubwebsDTM Result Set ............... 50
      3.2.4.15 proc_EstimateDocsSize .............................. 51
        3.2.4.15.1 EstimatedSize Result Set ........................ 52
      3.2.4.16 proc_FetchContentTypeInScope ..................... 52
        3.2.4.16.1 Content Type Result Set .......................... 53
      3.2.4.17 proc_FetchNextSiteForUpgrade ..................... 53
      3.2.4.18 proc_FixV2ContentTypeField ....................... 54
<table>
<thead>
<tr>
<th>Proc ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.2.4.42.1</td>
<td>All File URLs Result Set</td>
</tr>
<tr>
<td>3.2.4.43</td>
<td>proc_ListAllWebsOfSite</td>
</tr>
<tr>
<td>3.2.4.43.1</td>
<td>SiteWebs Result Set</td>
</tr>
<tr>
<td>3.2.4.44</td>
<td>proc_ListChildWebs</td>
</tr>
<tr>
<td>3.2.4.44.1</td>
<td>ChildWebs Result Set</td>
</tr>
<tr>
<td>3.2.4.45</td>
<td>proc_ListChildWebsFiltered</td>
</tr>
<tr>
<td>3.2.4.45.1</td>
<td>List Child Webs Filtered Result Set</td>
</tr>
<tr>
<td>3.2.4.46</td>
<td>proc_ListContentTypeInUse</td>
</tr>
<tr>
<td>3.2.4.46.1</td>
<td>Content Type Descendants Result Set</td>
</tr>
<tr>
<td>3.2.4.46.2</td>
<td>Content Type List Usage Result Set</td>
</tr>
<tr>
<td>3.2.4.47</td>
<td>proc_ListContentTypesInWeb</td>
</tr>
<tr>
<td>3.2.4.47.1</td>
<td>Result Set</td>
</tr>
<tr>
<td>3.2.4.48</td>
<td>proc_ListContentTypesInWebRecursive</td>
</tr>
<tr>
<td>3.2.4.49</td>
<td>proc_ListDerivedContentTypes</td>
</tr>
<tr>
<td>3.2.4.49.1</td>
<td>Derived Site Content Types Result Set</td>
</tr>
<tr>
<td>3.2.4.49.2</td>
<td>Derived Content Types Result Set</td>
</tr>
<tr>
<td>3.2.4.50</td>
<td>proc_ListsUsingFieldTemplate</td>
</tr>
<tr>
<td>3.2.4.50.1</td>
<td>Lists Using Field Result Set</td>
</tr>
<tr>
<td>3.2.4.51</td>
<td>proc_ListUnghostedFieldTemplatesInList</td>
</tr>
<tr>
<td>3.2.4.51.1</td>
<td>Unghosted List Fields Result Set</td>
</tr>
<tr>
<td>3.2.4.52</td>
<td>proc_MakeViewDefaultForContentType</td>
</tr>
<tr>
<td>3.2.4.53</td>
<td>proc_MakeViewDefaultForList</td>
</tr>
<tr>
<td>3.2.4.54</td>
<td>proc_MakeViewMobileDefaultForList</td>
</tr>
<tr>
<td>3.2.4.55</td>
<td>proc_MapContentTypeToList</td>
</tr>
<tr>
<td>3.2.4.56</td>
<td>proc_MapFieldToListAndView</td>
</tr>
<tr>
<td>3.2.4.57</td>
<td>proc_MapUrlToListAndView</td>
</tr>
<tr>
<td>3.2.4.58</td>
<td>proc_MapV2FieldToList</td>
</tr>
<tr>
<td>3.2.4.59</td>
<td>proc_markWebAsProvisioned</td>
</tr>
<tr>
<td>3.2.4.60</td>
<td>proc_MergeWeb</td>
</tr>
<tr>
<td>3.2.4.60.1</td>
<td>Audit Mask Result Set</td>
</tr>
<tr>
<td>3.2.4.61</td>
<td>proc_MiniSproc</td>
</tr>
<tr>
<td>3.2.4.61.1</td>
<td>Site URL Result Set</td>
</tr>
<tr>
<td>3.2.4.61.2</td>
<td>Domain Group Cache Versions Result Set</td>
</tr>
<tr>
<td>3.2.4.61.3</td>
<td>Domain Group Cache Back-End Database Server Update Result Set</td>
</tr>
<tr>
<td>3.2.4.61.4</td>
<td>Domain Group Cache Front-End Web Server Update Result Set</td>
</tr>
<tr>
<td>3.2.4.61.5</td>
<td>Site Metadata Result Set</td>
</tr>
<tr>
<td>3.2.4.61.6</td>
<td>Event Receivers Result Set</td>
</tr>
<tr>
<td>3.2.4.61.7</td>
<td>User Document Security Context Result Set</td>
</tr>
<tr>
<td>3.2.4.62</td>
<td>proc_ProvisionContentType</td>
</tr>
<tr>
<td>3.2.4.62.1</td>
<td>Content Type Exists Result Set</td>
</tr>
<tr>
<td>3.2.4.63</td>
<td>proc_ReleaseSiteUpgradeSession</td>
</tr>
<tr>
<td>3.2.4.64</td>
<td>proc_RemoveFromSiteUpgradeQueue</td>
</tr>
<tr>
<td>3.2.4.65</td>
<td>proc_RenameListItemContentType</td>
</tr>
<tr>
<td>3.2.4.66</td>
<td>proc_ResolveWikiLinkItem</td>
</tr>
<tr>
<td>3.2.4.66.1</td>
<td>Resolve Wiki Link Item Result Set</td>
</tr>
<tr>
<td>3.2.4.67</td>
<td>proc_RetrieveSiteHealthCheckResults</td>
</tr>
<tr>
<td>3.2.4.67.1</td>
<td>Retrieve Site Health Check Results Result Set</td>
</tr>
<tr>
<td>3.2.4.68</td>
<td>proc_RetrieveSiteUpgradeSession</td>
</tr>
<tr>
<td>3.2.4.69</td>
<td>proc_SetListFormToUrl</td>
</tr>
<tr>
<td>3.2.4.70</td>
<td>proc_SetSitePlatformVersion</td>
</tr>
<tr>
<td>3.2.4.71</td>
<td>proc_SetSiteFlags</td>
</tr>
<tr>
<td>3.2.4.72</td>
<td>proc_SetSitePortalProps</td>
</tr>
</tbody>
</table>
3.2.4.73 proc_SetSiteProps ................................................................. 103
3.2.4.74 proc_SetTpView ................................................................. 104
3.2.4.75 proc_SetWebMetaInfo ...................................................... 105
3.2.4.76 proc_SetWebUsageData .................................................... 106
3.2.4.77 proc_StoreSiteHealthCheckResults ................................ 107
3.2.4.78 proc_StoreUserInfoListInfo ............................................. 107
3.2.4.79 proc_UnmapContentTypeFromList .................................... 108
3.2.4.80 proc_UnmapFieldFromList ................................................ 108
3.2.4.81 proc_UnmapFieldsFromContentType ............................... 109
3.2.4.82 proc_UpdateContentTypeInScope ...................................... 109
3.2.4.83 proc_UpdateFeatureProperties ......................................... 111
3.2.4.84 proc_UpdateFeatureVersion ............................................. 112
3.2.4.85 proc_UpdateListContentTypes .......................................... 112
3.2.4.86 proc_UpdateListFields .................................................... 113
3.2.4.87 proc_UpdateSiteHashKey .................................................. 114
3.2.4.88 proc_UpdateSiteUpgradeSession ...................................... 115
3.2.4.89 proc_UpdateTpWebMetaInfo ........................................... 116
3.2.4.90 proc_GetFollowableLists ................................................ 119
3.2.4.90.1 Followed List Result Set .............................................. 120
3.2.4.91 proc_GetLastItemModifiedDates ..................................... 120
3.2.4.91.1 List Result Set ............................................................ 121
3.2.5 Timer Events ................................................................. 121
3.2.6 Other Local Events ........................................................... 121
3.3 Front-End Web Server Client Details ....................................... 121
3.3.1 Abstract Data Model ............................................................ 121
3.3.2 Timers ................................................................. 122
3.3.3 Initialization ................................................................. 122
3.3.4 Message Processing Events and Sequencing Rules ............... 122
3.3.5 Timer Events ................................................................. 122
3.3.6 Other Local Events ........................................................... 123
4 Protocol Examples ............................................................... 124
4.1 Features ................................................................. 124
4.2 Content Types and Columns ................................................... 125
4.2.1 Create, Rename, and Delete a Text Column ...................... 125
4.2.2 Create a Text Site Column ................................................ 130
4.2.3 Add a Site Column to a List .............................................. 130
4.2.4 Change the Name of a Site Column and Propagate to Lists.... 131
4.2.5 Create a New Site Content Type ....................................... 133
4.2.6 Add Site Column to Content Type ..................................... 136
4.3 Views ................................................................. 137
4.4 Custom Actions ............................................................. 137
4.4.1 Add a Custom Action ...................................................... 137
4.4.2 Retrieve a Custom Action ................................................ 138
4.4.3 Delete a Custom Action ................................................... 139
4.5 Metadata Information ....................................................... 140
4.5.1 proc_SetWebMetaInfo ...................................................... 140
5 Security ................................................................. 142
5.1 Security Considerations for Implementers ............................ 142
5.2 Index of Security Parameters .............................................. 142
6 Appendix A: Product Behavior ................................................ 143
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)
NULL GUID
XML

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

absolute URL
activation
app principal
app web domain identifier
audit flag
back-end database server
base type
CAML
cascading style sheet (CSS)
class identifier (CLSID)
Collaborative Application Markup Language (CAML)
collation
collation order
column
content database
content type
content type identifier
content type order
content type resource folder
CSS
custom action
customized
default form
default list view
default mobile list view
default view
descendant content type
directory name
display form
display name
personal Web Part
portal site
provision
provisioned
publishing level
query
record
relationship lookup field
request identifier
resource folder
resource identifier
resource token
result set
return code
root folder
row
sandboxed solution
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 property flag
site solution
site subscription identifier
site template
site title
site-relative URL
solution identifier
sort order
stored procedure
store-relative form
store-relative URL
Structured Query Language (SQL)
subsite
SystemID
theme
time zone
top-level site
transaction
Transact-Structured Query Language (T-SQL)
UI culture
uncustomized
Uniform Resource Locator (URL)
upgrade evaluation site collection
usage data
The following terms are specific to this document:

**app web domain:** The subdomain for a site(2) that belongs to an app instance.

**followable list:** A list in the content database that can be queried by using the followable list APIs.

**list flag:** An 8-byte unsigned integer bit mask that provides metadata about a SharePoint list.

**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.


---

[MS-WSSCCSP3] — v20120630
Windows SharePoint Services: Content Database Core List Schema and Site Provisioning Communications Version 3 Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
1.2.2 Informative References


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

1.3 Overview

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. 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 client, and the back-end database server, acting as a server.

1.3.1 Content Types

1.3.1.1 List Content Type Overview

List content types are objects defined at list level that specify list item behaviors in the list. A list can contain multiple list content types, which allows the list to contain list items with different behaviors. Every list item on a list 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 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 level that can be used to share common list 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 sites. 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.

A feature may be upgraded to change its behavior.

1.3.3 Custom Actions

Custom actions provide the ability to include pieces of dynamic functionality. Custom actions can be defined at the site collection, site, and list levels. Custom actions are often used to extend functionality to the user for a specific site collection, site, or list.

1.3.4 Views

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

1.3.5 List Schema

A list 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.
1.3.5.1 List Column Overview

List columns are objects defined at list 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 (sort order, filtering, grouping, and so on) based on their values set on list columns.

1.3.5.2 Site Column Overview

Site columns are objects defined at site level that can be used to share common list column definitions across lists. 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 hierarchy and can block deletion of a site column when there are still list columns referencing it. Site columns are destroyed when the site on which they are defined is deleted.

1.3.6 List/Web Metainfo

The list 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 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.

1.3.7 File Handling

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

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

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

The 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.

1.3.8 Provisioning

A list can be provisioned from a list template hosted either on a front-end Web server or saved in the list template gallery of the site. 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 of the parent site. Site provisioning calls list provisioning to create predefined lists for the site.

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:

![Diagram showing transport stack]

**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

2.2.1.1 Upgrade Type

Upgrade Type is a 4-byte integer value used to indicate the type of site collection upgrade to be performed.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Build to build site collection upgrade</td>
</tr>
<tr>
<td>1</td>
<td>Version to version site collection upgrade</td>
</tr>
</tbody>
</table>

2.2.1.2 Upgrade Status

Upgrade Status is a 4-byte integer value used to indicate the status of the site collection upgrade.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>site collection upgrade has not started</td>
</tr>
<tr>
<td>1</td>
<td>site collection upgrade in progress</td>
</tr>
<tr>
<td>2</td>
<td>site collection upgrade finished with error</td>
</tr>
<tr>
<td>3</td>
<td>site collection upgrade finished without error</td>
</tr>
</tbody>
</table>

2.2.2 Bit Fields and Flag Structures

None.

2.2.3 Binary Structures

2.2.3.1 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).

The 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 convention 1.

0x01

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

0x010077745d60-fb5d-4415-b722-f63181fb6e9d

2.2.3.2 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 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 22 | 21 | 20 | 19 | 18 | 17 | 16 | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7  | 6  | 5  | 4  | 3  | 2  | 1  | 0 |
| 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:

```sql
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-01234bbcdff0'
SET @bin2 = CAST(@GUID2 as binary)

SET @packedarray = @bin1 + @bin2
```
2.2.3.3 List Base Type Pattern

A bit pattern used to indicate which base types an operation will operate on.

The pattern is in the following format.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0</td>
<td>Generic list</td>
</tr>
<tr>
<td>Bit 1</td>
<td>Document Library</td>
</tr>
<tr>
<td>Bit 2</td>
<td>Unused</td>
</tr>
<tr>
<td>Bit 3</td>
<td>Discussion Board</td>
</tr>
<tr>
<td>Bit 4</td>
<td>Survey</td>
</tr>
<tr>
<td>Bit 5</td>
<td>Issue</td>
</tr>
<tr>
<td>Bit 6 – 31</td>
<td>Ignored</td>
</tr>
</tbody>
</table>

2.2.3.4 Usage Data Binary Field Structure

A structure that contains usage data for a site. The structure begins 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:

<table>
<thead>
<tr>
<th>Usage Data Header (100 bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page Data (Variable)</td>
</tr>
<tr>
<td>User Data (Variable)</td>
</tr>
<tr>
<td>Operating System Data (Variable)</td>
</tr>
<tr>
<td>Browser Data (Variable)</td>
</tr>
<tr>
<td>Referrer Data (Variable)</td>
</tr>
<tr>
<td>Reserved (290 bytes)</td>
</tr>
</tbody>
</table>

Usage Data Header (100 bytes): Defined in section 2.2.3.4.1.

Page Data (Variable): A series of Usage Records that specify the pages that have been requested from a site. 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.

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

### 2.2.3.4.1 Usage Data Header Structure

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

<table>
<thead>
<tr>
<th>byte1</th>
<th>byte2</th>
<th>byte3</th>
<th>byte4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Update Counter</td>
<td>Page Data Offset</td>
<td>User Data Offset</td>
<td>Operating System Data Offset</td>
</tr>
<tr>
<td>Browser Data Offset</td>
<td>Referrer Data Offset</td>
<td>Reserved 1</td>
<td>...</td>
</tr>
<tr>
<td>Page Data Count</td>
<td>User Data Count</td>
<td>Operating System Data Count</td>
<td>Browser Data Count</td>
</tr>
<tr>
<td>Referrer Data Count</td>
<td>Reserved 2</td>
<td>...</td>
<td>Last Accessed Day</td>
</tr>
<tr>
<td>Rollover Day</td>
<td>Reserved 3</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 beginning of the structure to the beginning of the Page Data.

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

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

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

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

2.2.3.4.2 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.

<table>
<thead>
<tr>
<th>Record Description (variable)</th>
<th>Record Data (variable)</th>
</tr>
</thead>
</table>

**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:

<table>
<thead>
<tr>
<th>byte1</th>
<th>byte2</th>
<th>byte3</th>
<th>byte4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bytes</td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>R</td>
<td>R</td>
<td>R</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>A</td>
<td></td>
</tr>
<tr>
<td>Hit Vector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hit Values (variable)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.

### 2.2.4 Common Result Sets

#### 2.2.4.1 List Content Types Result Set

The List Content Types Result Set returns 0 or more rows of content types. It is defined using T-SQL syntax, as follows:

```sql
ContentTypeId varbinary(512),
```
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 2.4.6 Content Type References.

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.

SolutionId: MUST contain the solution identifier of the solution used to deploy the content type. If the content type was not deployed via a solution it MUST be NULL.

2.2.4.2 Site Collection Upgrade Info Result Set

The site collection upgrade info Result Set returns 0 or more rows of site upgrade information. It is defined using T-SQL syntax, as follows:

SiteId uniqueidentifier,
SessionId uniqueidentifier,
SessionXML nvarchar(max),
UpgradeStatus int,
UpgradeType int,
ErrorCount int,
WarningCount int,
RequestDate datetime,
StartTime datetime,
LastUpdated datetime,
RetryCount int,
LogFileLocation nvarchar(260);

SiteId: The site collection identifier of the site collection.

SessionId: A unique identifier that is associated with the site collection upgrade.

SessionXML: Contains implementation-specific data about the current status of the site collection upgrade, but otherwise opaque to the back-end database server.

UpgradeStatus: A 4-byte Upgrade Status indicating the status of the site collection upgrade.

ErrorCount: number of errors for this site collection upgrade.

WarningCount: number of warnings for this site collection upgrade.

RequestDate: The UTC date and time when the site collection upgrade was requested.
**StartTime:** The UTC date and time when the site collection upgrade started.

**LastUpdated:** The UTC date and time when the site collection upgrade was last updated.

**RetryCount:** The number of site collection upgrade attempted on this site collection.

**LogFileLocation:** The site-relative URL of the file that contains the site collection upgrade log. MUST be empty string if there is no log file.

### 2.2.5 Tables and Views

#### 2.2.5.1 AllListsAux

This table contains the information for lists in the back-end database server. The AllListsAux Table MUST contain the ListID column. The client uses additional columns to store other metadata about the list in an implementation specific way. The AllListsAux Table contains the ListID column using T-SQL syntax, as follows:

```sql
TABLE AllListsAux {
    ListID     uniqueIdentifier   NOT NULL
};
```

**ListID:** The list identifier of the list.

### 2.2.6 XML Structures

#### 2.2.6.1 Namespaces

None.

#### 2.2.6.2 Simple Types

This specification does not define any common XML Schema simple type definitions.

#### 2.2.6.3 Complex Types

##### 2.2.6.3.1 Feature Property Definitions

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

```xml
<xsd:element name="Properties" type="FeaturePropertyDefinitions" minOccurs="0" maxOccurs="1" />
<xsd:complexType name="FeaturePropertyDefinitions">
    <xsd:sequence>
        <xsd:element name="Property" type="FeaturePropertyDefinition" minOccurs="0" maxOccurs="unbounded" />
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="FeaturePropertyDefinition">
    <xsd:attribute name="Key" type="xsd:string" />
    <xsd:attribute name="Value" type="xsd:string" />
</xsd:complexType>
```
The **Properties** element represents a collection of user-defined name/value pairs (represented by **Property** elements, with the **Key** attribute representing the name, and the **Value** attribute representing the value).

Example:

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

### 2.2.6.3.2 Site Health Check Result Message

The following XML schema definition (XSD) defines the Site Health Check Result Message:

```xml
<xsd:complexType name="ParentType" mixed="true">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="Hyperlink" type="HyperlinkType"/>
    <xsd:element name="Children" type="ChildrenType"/>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="ChildrenType" mixed="true">
  <xsd:choice minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="Hyperlink" type="HyperlinkType"/>
    <xsd:element name="Child" type="ParentType"/>
  </xsd:choice>
</xsd:complexType>

<xsd:complexType name="HyperlinkType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="Url" type="xsd:string" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:element name="Message" type="ParentType"/>
```

A message formatted according to this schema represents the results of running a site collection health rule, using a hierarchical structure with optional embedded hyperlinks.

Example:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Message>
  The following files have been customized from their default and may present some unexpected visuals or behavior after upgrade:
  <Children>
    <Child>http://host/sites/ExampleSite/default.aspx -
      <Hyperlink Url="reghost.aspx?Url=http://host/sites/ExampleSite14a/default.aspx">Reset page to default</Hyperlink>
    </Child>
    <Child>http://host/sites/ExampleSite/default.aspx -
  </Children>
</Message>
```
Reset specific pages to default to make the page lose customizations and any embedded data. Normally, you should do this only if you are having difficulty using the page after upgrade.

2.2.6.4 Elements

This specification does not define any common XML Schema element definitions.

2.2.6.5 Attributes

This specification does not define any common XML Schema attribute definitions.

2.2.6.6 Groups

This specification does not define any common XML Schema group definitions.

2.2.6.7 Attribute Groups

This specification does not define any common XML Schema attribute group definitions.

2.2.7 User-Defined Table Types

2.2.7.1 tvpLastItemModified

The tvpLastItemModifiedTable Type represents an array of lists about which metadata is being requested. The tvpLastItemModified Table Type is defined using T-SQL syntax, as follows.

```
TYPE tvpLastItemModified AS TABLE (
  ArrayIndex int,
  SiteId uniqueidentifier,
  DirName nvarchar(256),
  LeafName nvarchar(128)
);
```

**ArrayIndex:** An integer value representing the list’s index in the array.

**SiteId:** The Site Collection Identifier of the site collection containing the list.

**DirName:** The directory name of the location that contains the list.

**LeafName:** The leaf name of the location that contains the list.
3 Protocol Details

3.1 Common Details

3.2 Back-end Database Server Details

This section provides details about the back-end database server.

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.

3.2.1.1 Content Types

3.2.1.1.1 List Content Type Data Model

To apply a site content type to a list:

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 by 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),
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_IsContentTypeInUseInList. 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).

3.2.1.1.2 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_ListContentTypesInWebRecursive.
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 site’s content type collection 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 content type returned from step 4, update it using list content type data model (see
   preceding example).

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.

3.2.1.2 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

3.2.1.3 Views

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

- **Field**: A data type definition.
- **List item**: A data unit that stores information for a custom set of fields.
- **List**: 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 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.

### 3.2.1.4 List Schema

#### 3.2.1.4.1 List Column Data Model

To add a new list column to a list, 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**.

#### 3.2.1.4.2 Site Column Data Model

To add a site column:

1. Check the identifier and **field internal name** of the new column for duplicate entries already on the site by calling **proc_ListContentTypesInWebRecursive**. 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 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_ListContentTypesInWebRecursive`. If so, fail with error.

2. Delete the site column from the site's column collection by calling `proc_DeleteFieldTemplateInScope`.

### 3.2.1.5 Provisioning

To **provision** a list on a site:

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’s 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_GetUnghostedBaseFieldTemplateInSite`. 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 higher than 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.
3.2.1.6 Custom Actions

Custom actions can be created and modified by using the `proc_AddOrUpdateCustomAction` (section 3.2.4.4) stored procedure. To retrieve custom actions, the `proc_GetCustomActionsFromScope` stored procedure is used. To delete custom actions, the `proc_DeleteCustomAction` stored procedure is used.

3.2.1.7 Site Collection Health Checks

The back-end database server stores the results of the most recent run of each site collection health rule on each site collection. These health rules perform an implementation-independent check on the health of a site collection.

Front-end Web server calls `proc_StoreSiteHealthCheckResults` to store the results of running a site collection health rule on a particular site collection, and calls `proc_RetrieveSiteHealthCheckResults` to retrieve previously-stored results (if any) for a particular site collection health rule on a particular site collection.

3.2.1.8 Site Collection Upgrade

The back-end database server maintains a queue of site collection upgrade requests.

Front-end Web server calls `proc_AddToSiteUpgradeQueue` to add the site collection to the queue, and calls `proc_RemoveFromSiteUpgradeQueue` to remove the site collection from the queue.

Front-end Web server calls `proc_FetchNextSiteForUpgrade` to get the first available site collection from the queue, and calls `proc_AcquireSiteUpgradeSession` to lock the site collection and start site collection upgrade.

The back-end database server maintains the last updated time for the site collection, and the lock expires after 60 seconds. During the site collection upgrade, front-end Web server periodically calls `proc_HeartbeatSiteUpgradeSession` to refresh the lock on the site collection, and calls `proc_UpdateSiteUpgradeSession` to save the site collection upgrade progress. When site collection upgrade finishes, it calls `proc_ReleaseSiteUpgradeSession` to release the lock.

Front-end Web server calls `proc_RetrieveSiteUpgradeSession` to retrieve upgrade status about a particular site collection, and calls `proc_GetSiteUpgradeSession` to retrieve upgrade status about all site collections.

3.2.2 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.

3.2.3 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].

3.2.4 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

---

[MS-WSSCCSP3] — v20120630
Windows SharePoint Services: Content Database Core List Schema and Site Provisioning Communications Version 3 Protocol Specification
Copyright © 2012 Microsoft Corporation.
Release: July 16, 2012
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\}.

### 3.2.4.1 proc_AcquireSiteUpgradeSession

The proc_AcquireSiteUpgradeSession stored procedure is called to start a site collection upgrade by associating a unique identifier with a site collection. The stored procedure is defined using T-SQL syntax, as follows:

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

- **@SiteId**: The site collection identifier of the site collection.
- **@SessionId**: A unique identifier to be associated with this site collection.
- **@RequestGuid**: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion. The @SessionId is associated with the site collection.</td>
</tr>
<tr>
<td>1</td>
<td>The site collection is already associated with a different unique identifier.</td>
</tr>
<tr>
<td>2</td>
<td>The site collection cannot be found.</td>
</tr>
<tr>
<td>3</td>
<td>There are too many site collection upgrades in this database. The operation failed.</td>
</tr>
</tbody>
</table>

**Result Sets:** MUST NOT return any result sets.

### 3.2.4.2 proc_ActivateFeature

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

```sql
PROCEDURE proc_ActivateFeature(
    @SiteId uniqueidentifier,
    @WebId uniqueidentifier,
    @FeatureId uniqueidentifier,
    @SolutionId uniqueidentifier,
    @SolutionWebId uniqueidentifier,
    @SolutionLevel int,
    @Flags int,
    @Version nvarchar(64),
    @Properties nvarchar(max) = NULL,
    @FeatureTitle nvarchar(max) = NULL,
);
```

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

@WebId: MUST be a site identifier containing the NULL GUID if the feature is scoped to a site collection. 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.

@SolutionId: The identifier of the sandboxed solution or site solution that installed the Web Part. If the feature was not deployed via a solution it MUST be NULL.

@SolutionWebId: The Site Identifier (as specified in [MS-WSSFO3] section 2.2.1.1.11) of the site which is associated with the solution. If the site is not associated with the Solution or the feature is not deployed via a solution it MUST be NULL.

@SolutionLevel: Specifies the installation state of the sandboxed solution. If the feature was not deployed via a solution it MUST be 1. The value MUST be one of the integers in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active</td>
<td>Solution is in installed active state.</td>
</tr>
<tr>
<td>2</td>
<td>Installing</td>
<td>Installation or upgrade of the solution is in progress.</td>
</tr>
<tr>
<td>3</td>
<td>Previous</td>
<td>Solution is from a previous installation.</td>
</tr>
</tbody>
</table>

@Flags: Specifies the deployed scope of the solution that contains the feature. It MUST contain one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The feature is not deployed via a solution or the solution is deployed at the farm scope.</td>
</tr>
<tr>
<td>1</td>
<td>The solution is deployed at the site collection scope.</td>
</tr>
<tr>
<td>2</td>
<td>The solution is deployed at the site scope.</td>
</tr>
</tbody>
</table>

@Version: Specifies the version of the feature.

@Properties: An XML fragment, that MUST conform to the XML schema for the feature as defined in Feature Property Definitions (section 2.2.6.3.1). If the @Properties parameter is NULL, the Feature Property Definitions (section 2.2.6.3.1) MUST be empty.

@FeatureTitle: Specifies the title of the feature.

@FeatureDesc: Specifies the description of the feature.

@FeatureFolder: Specifies the name of the subdirectory that contains the feature.
@RequestGuid: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site collection or site does not exist.</td>
</tr>
<tr>
<td>80</td>
<td>The feature is already marked active in the site or site collection.</td>
</tr>
<tr>
<td>1168</td>
<td>Failed to mark feature as active because of an internal error.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

### 3.2.4.3 proc_AddContentTypeToScope

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

```sql
PROCEDURE proc_AddContentTypeToScope(
    @SiteId                 uniqueidentifier,
    @Class                  tinyint,
    @ContentTypeId          varbinary(512),
    @Scope                  nvarchar(256),
    @Definition             nvarchar(max),
    @ParentContentTypeId    varbinary(512)   = NULL,
    @ParentScopeIn          nvarchar(256)    = NULL,
    @ResourceDir            nvarchar(128)    = NULL,
    @FeatureId              uniqueidentifier = NULL,
    @SolutionId             uniqueidentifier = NULL,
    @NextChildByte          tinyint = NULL,
    @RequestGuid          uniqueidentifier = NULL OUTPUT
);
```

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Site column.</td>
</tr>
<tr>
<td>1</td>
<td>Site content type.</td>
</tr>
</tbody>
</table>

@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 URL of the site to which the site content type or site column will be added.

@Definition: The XML fragment that defines the site content type or site column. The XML schemas for these structures are defined in the section 2.4.6 - Content Type References, of [MS-WSSCAML] and section 2.3.2.9 - FieldDefinitions, of [MS-WSSCAML].
@ParentContentTypeId: If @Class is equal to 0 then this MUST be NULL. If @Class is equal to 1 then this MUST be the identifier of the parent 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 URL to which the parent site content type is registered.

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

@FeatureId: The feature identifier of the feature used to deploy the site content type or site column. If the site content type or site column was not deployed via a feature, this MUST be NULL.

@SolutionId: Specifies the solution identifier of the solution used to deploy the site content type. If the site content type was not deployed via a solution, this MUST be NULL.

@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.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>80</td>
<td>The site content type or site column was not created.</td>
</tr>
<tr>
<td>85</td>
<td>The site content type or site column is already registered to the site designated by @SiteId and @Scope or an ancestor of that site.</td>
</tr>
<tr>
<td>144</td>
<td>@Scope refers to a site that is not within the site collection designated by the @SiteId parameter.</td>
</tr>
<tr>
<td>212</td>
<td>The site collection is locked.</td>
</tr>
<tr>
<td>1816</td>
<td>The site collection quota for the site collection has been exceeded.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.4 proc_AddOrUpdateCustomAction

The proc_AddOrUpdateCustomAction stored procedure creates or updates an existing custom action.

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

```sql
PROCEDURE proc_AddOrUpdateCustomAction(
    @Id             uniqueidentifier,
    @ScopeId        uniqueidentifier,
    @SiteId         uniqueidentifier,
    @WebId          uniqueidentifier,
    @FeatureId      uniqueidentifier,
    @SolutionId     uniqueidentifier,
    @SolutionWebId  uniqueidentifier,
    @SolutionLevel  int,
    @ScopeType      int,
    @Properties     nvarchar(max),
    @Version        nvarchar(64)
)
```
@Id: The custom action identifier. MUST NOT be a NULL GUID and MUST NOT be NULL. If an existing custom action with this GUID exists, this stored procedure will update the associated custom action. Otherwise, this stored procedure will create a new custom action with this specified GUID.

@ScopeId: MUST be the site collection identifier, site identifier, or list identifier corresponding to the specified @ScopeType.

@SiteId: MUST be the site collection identifier in which the custom action will reside.

@WebId: If the scope of the custom action being created is a site collection custom action, this parameter MUST be set to a NULL GUID. However, if the custom action is a site custom action or list custom action, this parameter MUST be set to a site identifier in which the custom action will exist.

@FeatureId: MUST be a feature identifier or a NULL GUID.

@SolutionId: The identifier of the sandboxed solution or site solution that installed the Web Part. If the custom action was not deployed via a solution it MUST be NULL.

@SolutionWebId: The Site Identifier (as specified in [MS-WSSFO3] section 2.2.1.1.11) of the site which is associated with the solution. If there is no site associated with the Solution or the custom action is not deployed via a solution it MUST be NULL.

@SolutionLevel: Specifies the installation state of the sandboxed solution or site solution. If the custom action was not deployed via a solution it MUST be 1. The value MUST be an integer which is listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active</td>
<td>Solution is in installed active state.</td>
</tr>
<tr>
<td>2</td>
<td>Installing</td>
<td>Installation or upgrade of the solution is in progress.</td>
</tr>
<tr>
<td>3</td>
<td>Previous</td>
<td>Solution is from a previous installation.</td>
</tr>
</tbody>
</table>

@ScopeType: The scope of the custom action. MUST be one of the values listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>This is the site collection custom action identifier. The custom action is a site collection custom action.</td>
</tr>
<tr>
<td>3</td>
<td>This is the site custom action identifier. The custom action is a site custom action.</td>
</tr>
<tr>
<td>4</td>
<td>This is the list custom action identifier. The custom action is a list custom action.</td>
</tr>
</tbody>
</table>

@Properties: The custom action data describing its functionality. MUST NOT be NULL.

@Version: A version for the custom action being created or updated. MUST NOT be NULL.

Return Code Values: Returns an integer return code which MUST be listed in the following table:

<table>
<thead>
<tr>
<th>Return Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>0</td>
<td>The custom action was successfully created or updated.</td>
</tr>
<tr>
<td>212</td>
<td>The custom action was not successfully created or updated because the site</td>
</tr>
<tr>
<td></td>
<td>collection is locked.</td>
</tr>
<tr>
<td>1816</td>
<td>The custom action was not successfully created or updated because the site</td>
</tr>
<tr>
<td></td>
<td>collection quota has been exceeded.</td>
</tr>
<tr>
<td>-2147467259</td>
<td>The custom action was not successfully created or updated.</td>
</tr>
</tbody>
</table>

**Result Sets:**

This stored procedure MUST return the **Add or Update Custom Action Result Set** (section 3.2.4.4.1).

### 3.2.4.4.1 Add or Update Custom Action Result Set

This result set MUST return 0 rows if a new custom action was created. The result set MUST return 1 row if an existing custom action is updated. This result set MUST be ignored. This result set is defined using T-SQL syntax, as follows:

```sql
{UnnamedColumn}    int;
{UnnamedColumn}: This column MUST be ignored.
```

### 3.2.4.5 proc_AddToSiteUpgradeQueue

The **proc_AddToSiteUpgradeQueue** stored procedure is called to add the site collection to the site collection upgrade queue. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_ActivateFeature(
    @SiteId               uniqueidentifier,
    @SubscriptionId       varbinary(16),
    @UpgradeType          int,
    @RequestGuid          uniqueidentifier = NULL OUTPUT
);
```

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

@SubscriptionId: The site subscription identifier of the site collection.

@UpgradeType: Specifies the upgrade type.

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

**Return Code Values:** An integer which the protocol client MUST ignore.

**Result Sets:** MUST NOT return any result sets.
3.2.4.6 proc_CloneFeature

The proc_CloneFeature stored procedure is called to make a copy of an active feature and custom actions that match the given @SiteId, @SolutionWebId, @SolutionId, and @FeatureId to a new installation state. The information associated with the copied Feature and Custom Actions are identical to the original with the exception of the new installation state.

```
PROCEDURE proc_CloneFeature(
    @SiteId               uniqueidentifier,
    @SolutionWebId        uniqueidentifier,
    @SolutionId           uniqueidentifier,
    @FeatureId            uniqueidentifier,
    @SolutionLevelCurrent  int,
    @SolutionLevelNew     int
);
```

@SiteId: The site collection identifier of the site collection of the feature to clone. This parameter MUST NOT be NULL.

@SolutionWebId: The Site Identifier (as specified in [MS-WSSFO3] section 2.2.1.1.11) of the site which is associated with the feature to clone.

@SolutionId: The identifier of the sandboxed solution or site solution associated with the feature to clone.

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

@SolutionLevelCurrent: Specifies the installation state of feature to clone.

@SolutionLevelNew: Specifies the installation state to set on the cloned feature. @SolutionLevelNew MUST be a different value than @SolutionLevelCurrent.

The value of @SolutionLevelCurrent and @SolutionLevelNew MUST be an integer which is listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active</td>
<td>Solution is in installed active state.</td>
</tr>
<tr>
<td>2</td>
<td>Installing</td>
<td>Installation or upgrade of the solution is in progress.</td>
</tr>
<tr>
<td>3</td>
<td>Previous</td>
<td>Solution is from a previous installation.</td>
</tr>
</tbody>
</table>

Return Code Values: MUST be 0, indicating a successful completion.

Result Sets: MUST NOT return any result sets.

3.2.4.7 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,
    Preliminary
```
@ContentTypeId varbinary(512),
@Scope nvarchar(256),
@TargetDir nvarchar(256),
@RequestGuid uniqueidentifier = NULL OUTPUT
);

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

@WebId: The site identifier of the site in which the content type resource folder to be copied resides.

@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 URL of the site that contains the content type whose content type resource folder is to be copied.

@TargetDir: The store-relative URL of the folder where the specified content type resource folder is to be copied.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>2</td>
<td>The specified target destination was not found.</td>
</tr>
<tr>
<td>3</td>
<td>The content type does not exist.</td>
</tr>
<tr>
<td>5</td>
<td>User is not authorized to make this change</td>
</tr>
<tr>
<td>15</td>
<td>Attempt to rename an excluded directory type.</td>
</tr>
<tr>
<td>87</td>
<td>There is an inconsistency between the expected number of documents to be modified and the observed number which would be modified. The only way this happens is if there are concurrent attempts made to change affected objects.</td>
</tr>
<tr>
<td>206</td>
<td>Attempted to move folders that exceed file name range</td>
</tr>
<tr>
<td>212</td>
<td>Write Lock Error when creating a file or folder.</td>
</tr>
<tr>
<td>1359</td>
<td>Internal execution error occurred.</td>
</tr>
<tr>
<td>1816</td>
<td>Disk quota error. The quota for the site collection has reached the maximum allowable limit.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.8 proc_DeactivateContentTypeInScope

The proc_DeactivateContentTypeInScope stored procedure is called to deactivate a site content type or site column in a specific site. 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,
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);

@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 MUST NOT be deleted.

@UserId: The user identifier of the user who is initiating this procedure. If @WebId is NULL, this parameter MUST be ignored.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Site column.</td>
</tr>
<tr>
<td>1</td>
<td>Site content type.</td>
</tr>
</tbody>
</table>

@Scope: The store-relative 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 value of the parameter MUST be in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not deactivate specified site content type or site column if it is in use or is part of a feature.</td>
</tr>
<tr>
<td>1</td>
<td>Do not deactivate specified site content type or site column if it is in use.</td>
</tr>
<tr>
<td>2</td>
<td>Force deactivation even if the site content type or site column is in use or is read-only.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>2</td>
<td>The site content type or site column is not activated on this site.</td>
</tr>
<tr>
<td>4307</td>
<td>The site content type or site column is in use.</td>
</tr>
<tr>
<td>6009</td>
<td>The site content type or site column is part of a feature and, therefore, is read-only.</td>
</tr>
</tbody>
</table>
Result Sets: MUST NOT return any result sets.

3.2.4.9 proc_DeactivateFeature

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

```sql
PROCEDURE proc_DeactivateFeature(
    @SiteId uniqueidentifier,
    @WebId uniqueidentifier,
    @FeatureId uniqueidentifier,
    @SolutionLevel int,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

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

@WebId: MUST be a NULL GUID if the feature is site collection feature scoped. Otherwise, this parameter MUST be set to 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. This parameter MUST NOT be NULL.

@SolutionLevel: Specifies the installation state of the sandboxed solution or site solution to be marked inactive. If the feature was not deployed via a solution it MUST be 1. The value MUST be an integer which is listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active</td>
<td>Solution is in installed active state.</td>
</tr>
<tr>
<td>2</td>
<td>Installing</td>
<td>Installation or upgrade of the solution is in progress.</td>
</tr>
<tr>
<td>3</td>
<td>Previous</td>
<td>Solution is from a previous installation.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site collection or site does not exist, or the feature is not currently marked active in the site collection or site.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.10 proc_DeleteContentTypeInScope

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

```sql
PROCEDURE proc_DeleteContentTypeInScope(
    ...
);
```
@SiteId: The site collection identifier of the site collection that contains the requested site.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Site column.</td>
</tr>
<tr>
<td>1</td>
<td>Site content type.</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not delete specified site content type or site column if it is in use, or is part of a feature.</td>
</tr>
<tr>
<td>1</td>
<td>Do not delete specified site content type or site column if it is in use.</td>
</tr>
<tr>
<td>2</td>
<td>Force delete even if the site content type or site column is in use or is read-only.</td>
</tr>
</tbody>
</table>

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

This stored procedure calls another stored procedure, `proc_DeactivateContentTypeInScope` with the parameters listed in the following table:

<table>
<thead>
<tr>
<th>External Parameter</th>
<th>Value Passed to the Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>@SiteId</td>
<td>@SiteId</td>
</tr>
<tr>
<td>@WebId</td>
<td>NULL</td>
</tr>
<tr>
<td>@UserId</td>
<td>0</td>
</tr>
<tr>
<td>@Class</td>
<td>@Class</td>
</tr>
<tr>
<td>@Scope</td>
<td>@Scope</td>
</tr>
<tr>
<td>@ContentTypeId</td>
<td>@ContentTypeId</td>
</tr>
<tr>
<td>@IsDeactivatingFeature</td>
<td>@IsDeactivatingFeature</td>
</tr>
<tr>
<td>@RequestGuid</td>
<td>@RequestGuid</td>
</tr>
</tbody>
</table>

Return Code Values: An integer which MUST be one of the values from the following table:
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>2</td>
<td>The site content type or site column is not activated on this site.</td>
</tr>
<tr>
<td>4307</td>
<td>The site content type or site column is in use.</td>
</tr>
<tr>
<td>6009</td>
<td>The site content type or site column is part of a feature and, therefore, is read-only.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

### 3.2.4.11 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:

```sql
PROCEDURE proc_DeleteFieldTemplateInScope(
    @SiteId          uniqueidentifier,
    @Scope           nvarchar(256),
    @FieldId         uniqueidentifier,
    @BaseTypes       int,
    @RequestGuid     uniqueidentifier = NULL OUTPUT
);
```

- **@SiteId**: The site collection identifier of the site that has the site column to be deleted.
- **@Scope**: The store-relative URL of the site from which to delete the site column.
- **@FieldId**: The identifier 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.)
- **@RequestGuid**: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>2</td>
<td>The site column specified by the @FieldId parameter does not exist.</td>
</tr>
<tr>
<td>144</td>
<td>The site collection or scope specified by the parameters @SiteId or @Scope respectively does not exist.</td>
</tr>
<tr>
<td>4307</td>
<td>Cannot be deleted because the site column is being used in a content type or there exists a list whose base type is the type specified by the @BaseTypes parameter.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.
3.2.4.12 proc_DropListField

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

```t-sql
PROCEDURE proc_DropListField(
    @SiteId                uniqueidentifier,
    @WebId                 uniqueidentifier,
    @ListId                uniqueidentifier,
    @FieldId               uniqueidentifier,
    @ColName               nvarchar(64),
    @RowOrdinal            int,
    @ColName2              nvarchar(64),
    @RowOrdinal2           int,
    @IsIndexedField        bit,
    @NeedRemoveLinks       bit,
    @Fields                varbinary(max),
    @FieldsSize            int,
    @ContentTypes          varbinary(max),
    @ContentTypesSize      int,
    @Version               int,
    @RequestGuid           uniqueidentifier = NULL OUTPUT
);
```

@SiteId: The site collection identifier in which the site containing the list 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 2.2.4.3, that contains the data for the field being deleted. This value MUST be a valid column name in the [MS-WSSFO2], section 2.2.4.3, or MUST be NULL.

@RowOrdinal: Among a set of rows representing a list item for this list, the 0-based ordinal of the row containing the column indicated by @ColName that represents the field to be deleted.

@ColName2: The column name of the additional column in the [MS-WSSFO2], section 2.2.4.3, that contains data for this field if the field requires two columns to store data. This MUST be a valid column name in the [MS-WSSFO2], section 2.2.4.3, or MUST be NULL.

@RowOrdinal2: Among a set of rows representing a list item for this list, the 0-based ordinal of the row containing the column indicated by @ColName2 that represents the field to be deleted.

@IsIndexedField: This indicates whether the field to be deleted is an indexed field. The value MUST be 1 if the field is an indexed field, and MUST NOT be 1 if the field is not indexed.

@NeedRemoveLinks: MUST be 1 if the indicated field to be deleted can contain links.

@Fields: The implementation-specific version number followed by an XML fragment of the field definitions of the list where the definition of the field being deleted has been removed. The XML schema for this structure is defined in [MS-WSSCAML], section 2.3.2.9 - FieldDefinitions.

@FieldsSize: The size of the @Fields parameter in bytes.
@ContentTypes: An XML fragment specifying the updated content types registered for this list where the field being deleted has been removed. The XML schema for this structure is defined in [MS-WSSCAML], section 2.4.6 - Content Type References.

@ContentTypesSize: The size of the @ContentTypes parameter in bytes.

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>@SiteId, @WebId or @ListId do not specify a valid site collection identifier, site identifier or list identifier.</td>
</tr>
<tr>
<td>1150</td>
<td>Version conflict because the version passed in as @Version does not match the list’s metadata version.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.13 proc_EnumListsWithMetadata

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

```sql
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,  
    @ThresholdRowCount     int,  
    @MaxLists              int,  
    @NumLists              int,  
    @ListIds               varbinary(2^31-1),  
    @RequestGuid           uniqueidentifier = NULL OUTPUT )
);  
```

@SiteId: A site collection identifier.

@WebId: The site identifier of a site. This parameter is ignored when @ListsFilter = 3.

@WebsFilter: Contains a value that specifies the site filter type. If @ListsFilter is 0, 1 or 2, then @WebsFilter MUST be one of the values in the following table:
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The metadata is returned only for lists in the site specified by @WebId.</td>
</tr>
<tr>
<td>1</td>
<td>The metadata is returned for lists that belong to the site specified by @WebId and all of its child sites.</td>
</tr>
<tr>
<td>2</td>
<td>The metadata is returned for all lists that belong to the site collection given by @SiteId.</td>
</tr>
</tbody>
</table>

When @ListsFilter = 3 this parameter is ignored.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Metadata is only returned for lists that match the specified @BaseType and @ServerTemplate.</td>
</tr>
<tr>
<td>1</td>
<td>Metadata is only returned for lists with a match when searching in the field identified by @FieldId with the value specified by @FieldValue, or in all fields if @FieldId is NULL. The filtering based on @BaseType and @ServerTemplate is also applied.</td>
</tr>
<tr>
<td>2</td>
<td>Metadata is only returned for the lists that are specified by their identifying list identifier in @ListIds.</td>
</tr>
<tr>
<td>3</td>
<td>Metadata is only returned for the lists that are specified by their identifying list identifier in @ListIds and have at least one event receiver of type 3 or 10003 registered on it. See [MS-WSSFO3] section 2.2.1.2.6 - Event Receiver Type, for event receiver types.</td>
</tr>
</tbody>
</table>

@BaseType: If @ListsFilter is 0 or 1 then it MUST be:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Ignored</td>
</tr>
<tr>
<td>Other value</td>
<td>The [MS-WSSFO3] section 2.2.1.2.11 - List Base Type, for the list.</td>
</tr>
</tbody>
</table>

@ServerTemplate: If @ListsFilter is 0 or 1 then it MUST be:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>Ignored</td>
</tr>
<tr>
<td>Other value</td>
<td>The [MS-WSSFO3] section 2.2.1.2.12 - List Server Template, used to create the list.</td>
</tr>
</tbody>
</table>

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

@FieldId: If @ListsFilter is 1 then @FieldId identifies a field that will be used to search the list based on the value specified in @FieldValue. If @FieldId is NULL then the query is done based on the @FieldValue in all fields.

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

@FieldType: If @ListsFilter is 1 then it MUST be:
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Specifies that the field used in the search has type TEXT. The search is done based on the site collation order.</td>
</tr>
<tr>
<td>5</td>
<td>Specifies that the field used in the search has type CHOICE. The search is done based on the site collation order.</td>
</tr>
<tr>
<td>Other value</td>
<td>The search is done ignoring the site collation order.</td>
</tr>
</tbody>
</table>

If `@ListsFilter` is not 1, then this parameter is ignored.

`@PrefetchListScopes`: If set to 1 then the List Permissions Result Set (section 3.2.4.13.4) MUST be returned by this stored procedure. Otherwise the List Permissions Result Set MUST NOT be returned by this stored procedure.

`@ThresholdRowCount`: The threshold number of the unique list security scopes for the List Permissions result set. It MUST be greater than 0.

`@MaxLists`: If it is 0 or less, then it is ignored. If it is greater than 0, then it specifies the maximum number of lists for which metadata should be retrieved. If this stored procedure finds more lists than specified in `@MaxLists` it returns an error (see the table in Return Codes).

`@NumLists`: Specifies the number of entries in `@ListIds`. Used only when `@ListsFilter` = 2 or `@ListsFilter` = 3.

`@ListIds`: A List Identifier Packed Array (section 2.2.3.2) of list identifiers. Used only when `@ListsFilter` = 2 or `@ListsFilter` = 3.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>68</td>
<td>The number of lists that matched the criteria given by the input parameters exceeded the specified <code>@MaxLists</code> parameter.</td>
</tr>
</tbody>
</table>

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

3.2.4.13.1 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 that matched the selection criteria given by the input parameters.

3.2.4.13.2 List Metadata Result Set

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

The result set is defined using T-SQL syntax, as follows:
Note: If the List Metadata result set has no rows, then this stored procedure MUST NOT return the List Event Receivers Result Set (section 3.2.4.13.3) and MUST NOT return the List Permissions Result Set (section 3.2.4.13.4).

### 3.2.4.13.3 List Event Receivers Result Set

The List Event Receivers result set contains information about the event receivers registered for the lists that were returned in the List Metadata result set (see section 3.2.4.13.2).

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-WSSFO3] section 3.1.5.19.21 for details.

### 3.2.4.13.4 List Permissions Result Set

The List Permissions result set identifies permissions associated with the lists returned in the List Metadata result set (see section 3.2.4.13.2).

This result set can contain more than 1 row per list and the total number of rows is limited by the @ThresholdRowCount parameter. There MUST be as many rows per list as permissions that are associated with that list.

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:

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

**ListId**: The list identifier that identifies the list.

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

- If the list inherits permissions from the site and has no specific permissions, then a row is produced per list and its ScopeId MUST be 0x00.
- If the list has unique permission settings, then 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.
- If there are list items that have their own permissions then for each list item’s 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 inherits permissions from the site, then a row is produced per list, and Acl MUST be NULL.

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

**AnonymousPermMask:**

• If the list inherits permissions from the site and has no unique permissions, then a row is produced per list and **AnonymousPermMask** MUST be 0x00.

• If the list has its own permissions, then a row is produced per permission and **AnonymousPermMask** is a permission level that indicates the permissions granted to an anonymous user on this list.

• If there are list items that have their own permissions then 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 an anonymous user or a user who has no specific rights on that list item.

### 3.2.4.14 procEnumWebAndSubwebsDTM

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

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

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

**@WebId:** The site identifier of the site. This MUST NOT be NULL.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion. Result set MUST have 1 or more records.</td>
</tr>
<tr>
<td>3</td>
<td>No rows match the given @WebId. The result set MUST NOT have any records and MUST be empty.</td>
</tr>
</tbody>
</table>

**Result Sets:** MUST return the following result set:

### 3.2.4.14.1 WebsAndSubwebsDTM Result Set

WebsAndSubwebsDTM returns date and time properties of the site (2) specified in the @WebID parameter, and of all its child sites (2).

The WebsAndSubwebsDTM result set MUST contain exactly one row if the site (2) exists, and MUST contain no rows if there is no such site. The result set MUST contain more than one row if there are subsites for the specified @WebID parameter.

The **WebsAndSubwebsDTM** result set is defined using T-SQL syntax, as follows:
Id: This will be either the site identifier that was passed in as the parameter @WebID OR the site
identifiers of all the child sites (2) of the site (2) specified by @WebID. The value MUST NOT be
NULL.

FullUrl: The store-relative URL of the site (2). Each URL MUST have a leading '/'. The datatype is a
string with a maximum length of 257 characters. The value MUST NOT be NULL.

LastMetadataChange: The time stamp, in Coordinated Universal Time (UTC) format,
describing the last time that metadata properties where modified on the site (2) specified by @Id.
The value MUST NOT be NULL.

ListsMaxModified: The time stamp, in UTC, describing the last time when any of the lists
contained within the site (2) were modified. The value MUST NOT be NULL.

ListsMaxLastSecurityChange: The timestamp, in UTC, describing the last time when any of the
permissions on the lists contained within the site (2) were modified. The value MUST NOT be NULL.

3.2.4.15 proc_EstimateDocsSize

The proc_EstimateDocsSize stored procedure is called to provide an estimate of the total size of a
list or site in bytes. The stored procedure is defined using T-SQL syntax, as follows:

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

@ListId: The list identifier for the list. This parameter 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 site
collection specified by @SiteId 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 or equal to @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 the values in the following table:
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Do not include documents from the specified list or any other list in the site if @ListId is NULL, in the size estimate.</td>
</tr>
<tr>
<td>1</td>
<td>Include documents from the specified list, or site if @ListId is NULL, in the size estimate.</td>
</tr>
</tbody>
</table>

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

### 3.2.4.15.1 EstimatedSize Result Set

EstimatedSize returns a rough estimate of the total size for of a list 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:

```sql
{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:

<table>
<thead>
<tr>
<th>@ListID</th>
<th>@IncludeListDocs</th>
<th>{Size}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defines a list</td>
<td>0</td>
<td>a rough estimate of the list excluding documents</td>
</tr>
<tr>
<td>Defines a list</td>
<td>1</td>
<td>a rough estimate of the list including documents</td>
</tr>
<tr>
<td>NULL</td>
<td>0</td>
<td>a rough estimate of the site excluding documents</td>
</tr>
<tr>
<td>NULL</td>
<td>1</td>
<td>a rough estimate of the site including documents that are not part of any list</td>
</tr>
</tbody>
</table>

### 3.2.4.16 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:

```sql
PROCEDURE proc_FetchContentTypeInScope(
    @SiteId           uniqueidentifier,
    @Class            tinyint,
    @Scope            nvarchar(256),
    @ContentTypeId    varbinary(512),
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);
```

@SiteId: The site collection identifier of the site collection that contains the requested site content type or site column.
@Class: The type of record to be retrieved. The parameter MUST be one of the values in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Site column.</td>
</tr>
<tr>
<td>1</td>
<td>Site content type.</td>
</tr>
</tbody>
</table>

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>2</td>
<td>The requested site content type or site column was not found.</td>
</tr>
</tbody>
</table>

Result Sets: MUST return the following result set:

3.2.4.16.1 Content Type Result Set

Returns the definition and version of the specified site content type or site column. It MUST contain one row if the site content type or site column specified by the input parameters exists, or 0 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 2.4.6, and [MS-WSSCAML] section 2.3.2.9.

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

3.2.4.17 proc_FetchNextSiteForUpgrade

The proc_FetchNextSiteForUpgrade stored procedure is called to return one site collection from the site collection upgrade queue. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_FetchNextSiteForUpgrade(
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

@RequestGuid: The optional request identifier for the current request.
Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST return the following SiteUpgradeType result set. The result set MUST contain 0 or 1 rows.

3.2.4.17.1 SiteUpgradeType Result Set

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

```sql
{SiteId}          uniqueidentifier;
{UpgradeType}     int;
```

{SiteId}: The site collection identifier of the site collection.

{UpgradeType}: The Upgrade Type of the site collection.

3.2.4.18 proc_FixV2ContentTypeField

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

```sql
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.

3.2.4.19 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:

```sql
PROCEDURE proc_GetContentTypeIdFromUrl(
    @DocSiteID        uniqueidentifier,
    @DocDirName       nvarchar(256),
    ...
)
```
@DocSiteID: The site collection identifier for the site collection in which the document, list item or folder whose content type identifier is sought resides.

@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.

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.2.4.19.1 Object Content Type Identifier Result Set

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

- 1 row with 1 column.
- 0 rows with 1 column

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

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

<table>
<thead>
<tr>
<th>Unnamed Column Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ContentTypeId</td>
<td>If the passed in site collection identifier, directory name and leaf name correspond to a document, folder, or list item in the system, then the content type identifier of the current version of that document, folder, or list item is returned. This MUST be of type tContentTypeId.</td>
</tr>
<tr>
<td>0x012001</td>
<td>This is returned if the document, folder, or list item specified by the site collection identifier, directory name and leaf name, that are passed in does not have an explicitly assigned content type identifier.</td>
</tr>
</tbody>
</table>

If the directory name and leaf name do not correspond to a document, folder or list item, a result set with an unnamed column and zero rows MUST be returned.

3.2.4.20 proc_GetFeatureProperties

The proc_GetFeatureProperties stored procedure is called to retrieve the Feature Property Definitions (section 2.2.6.3.1) 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,
  @SolutionLevel    int,
  @RequestGuid      uniqueidentifier = NULL OUTPUT
);

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

@WebId: MUST be a site identifier containing the NULL GUID if the feature is scoped to a site collection. 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.

@SolutionLevel: Specifies the installation state of the feature.

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active</td>
<td>Solution is in installed active state.</td>
</tr>
<tr>
<td>2</td>
<td>Installing</td>
<td>Installation or upgrade of the solution is in progress.</td>
</tr>
<tr>
<td>3</td>
<td>Previous</td>
<td>Solution is from a previous installation.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The target site collection or site does not exist, or the feature is not marked active in the site collection or site.</td>
</tr>
</tbody>
</table>

Results Sets: The stored procedure MUST return one Feature Property Definitions Result Set (section 3.2.4.20.1) when the return code is 0. Otherwise, it MUST NOT return any result sets.

3.2.4.20.1 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 active.

Flags: Specifies the scope at which the solution used to deploy the feature is deployed. It MUST contain one of the following values:
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The feature is not deployed via a solution or the solution is deployed at the farm scope.</td>
</tr>
<tr>
<td>1</td>
<td>The solution is deployed at the site collection scope.</td>
</tr>
</tbody>
</table>

**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.

### 3.2.4.21 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:

```sql
PROCEDURE proc_GetFolderContentTypeOrder(
    @SiteId              uniqueidentifier,
    @CurrentFolderUrl    nvarchar(260),
    @RequestGuid         uniqueidentifier = NULL OUTPUT
);
```

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>87</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

#### 3.2.4.21.1 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:

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

- **{Empty}**: Contains an empty string.
3.2.4.21.2  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.

3.2.4.21.3  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.)

3.2.4.22  proc_GetListContentTypes

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

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

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

@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.

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:
3.2.4.22.1 Content Types Result Set

Returns a list of all content types in the specified list. If @WebId is not NULL, @ListId is not NULL, the site specified by @WebId exists, and the list specified by @ListId exists within that site 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 Content Type References.

3.2.4.23 proc_GetListIdsToSync

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

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

@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.

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

Return Code Values: An integer which MUST be 0 and MUST be ignored by the protocol client.

Result Sets: MUST return the following result set:

3.2.4.23.1 Lists Result Set

The Lists result set returns all lists (1) in the specified site that use a customized (2) field or a customized (2) list column. This result set MUST contain 1 row for each distinct list (1) in the site specified by @WebId that uses a customized (2) list column or an 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 (1) that uses a customized (2) list column or a customized (2) content type.

3.2.4.24 proc_GetParentWebUrl

The proc_GetParentWebUrl stored procedure is called to return the store-relative URL of the parent site of a specified site. The stored procedure is defined using T-SQL syntax, as follows:
PROCEDURE proc_GetParentWebUrl(
    @SiteId uniqueidentifier,
    @WebId uniqueidentifier,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);

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

@WebId: The site identifier for a site.

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

Return Code Values: An integer which MUST be 0 and MUST be ignored by the protocol client.

Result Sets: MUST return the following result set:

3.2.4.24.1 Parent Site URL Result Set

The Parent Site URL Result Set MUST be returned and MUST contain 1 row (if the site exists and has a parent) or 0 rows (if the site identifier specified by @WebId is invalid or if the site has no parent) defined using T-SQL syntax as follows:

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

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

3.2.4.25 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:

```sql
PROCEDURE proc_GetSiteProps(
    @WebSiteId uniqueidentifier,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

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

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

Return Code Values: An integer which MUST be 0 and MUST be ignored by the protocol client.

Result Sets: MUST return the following result set:

3.2.4.25.1 Site Props Result Set

The Site Props Result Set returns a set of properties of a site collection whose site collection identifier is specified by the @WebSiteId parameter.

The Site Props Result Set MUST contain 1 row if the site collection exists, and MUST contain 0 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,
PlatformVersion          nvarchar(64),
SourceSiteId             uniqueidentifier,
ExpirationDate           datetime,
EvalSiteId               uniqueidentifier,
UpgradeFlags             int,
UpgradeReminderDate      datetime;
```

**OWNERID**: A user identifier or site group identifier that owns the site collection. The value MUST NOT be NULL.

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

**PortalURL**: The **absolute URL** of the **portal site** which contains this site collection.

**PortalName**: The name of the portal site which contains this 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.

**PlatformVersion**: An implementation-specific string that represents the most current version (for example 4.0.25.0 or 15.0.10.0) of upgrade actions that have been performed in site collection upgrade.

**SourceSiteId**: If this site collection is an **upgrade evaluation site collection**, specifies the site collection identifier of the original site collection. The value MUST be NULL if this site collection is not an upgrade evaluation site collection.

**ExpirationDate**: The timestamp in UTC format that specifies the time after which this upgrade evaluation site collection will expire. The value MUST be NULL if this site collection is not an upgrade evaluation site collection.

**EvalSiteId**: The site collection identifier for the upgrade evaluation site collection that is associated with this site collection. The value MUST be NULL if this site collection is an upgrade evaluation site collection.

**UpgradeFlags**: The site collection upgrade flags for this site collection. Valid values are defined in [MS-WSSFO3] section 2.2.3.10.

**UpgradeReminderDate**: The timestamp in UTC format that specifies the time after which the end user SHOULD be reminded to perform site collection upgrade.
3.2.4.26  proc_GetSiteUpgradeSession

The proc_GetSiteUpgradeSession stored procedure is called to retrieve information about a site collection upgrade. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_GetSiteUpgradeSession (  
    @SubscriptionId                   varbinary(16) = NULL,
    @HideWaiting                      bit = 0,
    @ShowInProgress                   bit = 0,
    @ShowCompleted                    bit = 0,
    @ShowFailed                       bit = 0
);  
```

@SubscriptionId: If not NULL, the result set MUST only return information for site collections with the site subscription identifier equal to @SubscriptionId.

@HideWaiting: A bit flag specifying whether the result set will contain information about site collections where the site collection upgrade has not yet started. If 0, information about site collections where the site collection upgrade has not started MUST be returned, otherwise 1.

@ShowInProgress: A bit flag specifying whether information about site collections where a site collection upgrade is currently being performed will be returned. If 1, information about site collections where the site collection upgrade is currently being performed MUST be returned, otherwise 0.

@ShowCompleted: A bit flag specifying whether information about site collections where the site collection upgrade has finished without error will be returned. If 1, information about site collections where that site collection upgrade has finished without error MUST be returned, otherwise 0.

@ShowFailed: A bit flag specifying whether information about site collections where the site collection upgrade has finished with error will be returned. If 1, information about site collections where that site collection upgrade has finished with error MUST be returned, otherwise 0.

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST return a Site Upgrade Info result set. The result set MUST contain 0 or more rows.

3.2.4.27  proc_GetTpWebMetaData

The proc_GetTpWebMetaData stored procedure is called to request the metadata for a particular site. The stored procedure is defined using T-SQL syntax, as follows:

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

@WebSiteId: The site collection identifier for the site collection that contains the site whose metadata is requested. The value MUST NOT be NULL.
@WebDirName: The directory name of the site whose metadata is requested. The value MUST NOT be NULL.

@WebLeafName: The leaf name of the site whose metadata is requested. The value MUST NOT be NULL.

@DGCacheVersion: The version of the domain group cache as seen by 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. 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.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion. Result set MUST have one or more records.</td>
</tr>
<tr>
<td>3</td>
<td>No site matched the given information. MUST NOT return a result set.</td>
</tr>
<tr>
<td>1271</td>
<td>Access to this site is blocked.</td>
</tr>
</tbody>
</table>

Result Sets: proc_GetTpWebMetaData MUST return 0 to 5 result sets. All result sets returned will be sent in the following order:

3.2.4.27.1 Domain Group Cache Versions Result Set

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

3.2.4.27.2 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.

3.2.4.27.3 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.

3.2.4.27.4 Site MetaData Result Set

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

3.2.4.27.5 Site Event Receivers Result Set

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

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

```sql
PROCEDURE proc_GetUnghostedBaseFieldTemplateInSite(
    @SiteId           uniqueidentifier,
    @Scope            nvarchar(256),
    @FieldIdData      tvpArrayOfGuids READONLY,
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);
```

@SiteId: The site collection identifier.

@Scope: The scope of the site specified with a store-relative URL.

@FieldIdData: The tvpArrayOfGuids table type as defined in [MS-WSSFO3] section 2.2.8.1 containing field identifiers for which to return the field definition.

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.2.4.28.1 Field Definition Result Set

The Field Definition Result Set returns the field identifier of the field and the XML fragment that defines the customized field definition. The XML schema for this structure is defined in [MS-WSSFO3] section 2.2.7.3.3.

If the field exists in the specified site collection and in the specified scope, the Field Definition Result Set MUST contain 1 row. If the field 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:

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

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

Definition: Contains the customized XML definition of the field. This MUST be NULL if the field is uncustomized. The XML schema for this structure is defined in [MS-WSSFO3] section 2.2.7.3.3.

3.2.4.29 proc_GetUniqueListMetaData

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

```sql
PROCEDURE proc_GetUniqueListMetaData(
    @SiteId                   uniqueidentifier,
    @WebId                    uniqueidentifier,
    Preliminary
    Preliminary
)
```
@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 Unique Permissions result set is returned. If this bit is 1 and the permissions exist, proc_GetUniqueListMetaData MUST return the Unique Permissions result set. If the bit is 1 and the permissions do NOT exist, proc_GetUniqueListMetaData MUST return the NULL unique permissions result set.

@ThresholdScopeCount: The threshold number of the unique list security scopes for Unique Permissions result set. It MUST be greater than 0.

@PrefetchRelatedFields: A bit flag specifying whether the Related Fields result set is returned. If this bit is 1, proc_GetUniqueListMetaData MUST return the list related fields result set.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion. At least 2 and at most 4 result sets MUST be returned.</td>
</tr>
<tr>
<td>13</td>
<td>No list matched the parameters supplied. MUST NOT return any result set.</td>
</tr>
</tbody>
</table>

Result Sets: MUST return at least 2 and at most 4 result sets in case of Successful completion depending upon input parameters. Result sets that are returned will be sent in the order of the following sections.

3.2.4.29.1 List Metadata Result Set

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

The List Metadata result set is defined in [MS-WSSFO2], section 3.1.5.29.1 - List Metadata Result Set.

3.2.4.29.2 Unique Permissions Result Set

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

The Unique Permissions result set is defined in [MS-WSSFO2], section 3.1.5.34.2 - Unique Permissions Result Set.
3.2.4.29.3 NULL Unique Permissions Result Set

The NULL Unique Permissions result set contains an unlabeled result set with 1 row. This result set MUST 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 3.1.5.34.3 – NULL Unique Permissions Result Set.

3.2.4.29.4 Event Receivers Result Set

The Event Receivers result set contains information about the event receivers defined for this list. There MUST be 1 row in this result set for each 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 3.1.5.34.4 – Event Receivers Result Set.

3.2.4.29.5 Related Fields Result Set

The Related Fields result set contains information about all the relationship lookup fields in lists in the specified site collection whose target list is the specified list. This result set will be returned if @PrefetchRelatedFields parameter is set to 1 and the relationship lookup fields exist.

The Related Fields result set is defined in [MS-WSSDLIM2], section 3.1.4.63.1 – List Related Fields Result Set.

3.2.4.30 proc_GetWebExtendedMetaData

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

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

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

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

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.2.4.30.1 Creation and Modification Result Set

The result set returns the creation date and time and most recent modification date and time of a site whose site identifier is specified in the @WebId parameter.

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. The format of the date and time is: yyyy-mm-dd hh:mm:ss:mmm(24h).

**{Modified}**: Contains the most recent site modification date and time. The format of the date and time is:  yyyy-mm-dd hh:mm:ss:mmm(24h).

### 3.2.4.31 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:

```sql
PROCEDURE proc_GetWebFeatureList(
@SiteId                     uniqueidentifier,
@WebId                      uniqueidentifier,
@SolutionLevel             int,
@IsUserSolutionActivated    bit = NULL,
@RequestGuid               uniqueidentifier = NULL OUTPUT
);
```

**@SiteId**: The site collection identifier or the site collection whose active features are requested.

**@WebId**: If site scope features are being requested then this MUST be the site identifier of the site whose features are requested. Else if site collection scope features are being requested then this MUST be set to the NULL GUID.

**@SolutionLevel**: Specifies the installation state of the sandboxed solution or site solution. If the feature was not deployed via a solution it MUST be 1. The value MUST be an integer which is listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active</td>
<td>Solution is in installed active state.</td>
</tr>
<tr>
<td>2</td>
<td>Installing</td>
<td>Installation or upgrade of the solution is in progress.</td>
</tr>
<tr>
<td>3</td>
<td>Previous</td>
<td>Solution is from a previous installation.</td>
</tr>
</tbody>
</table>

**@IsUserSolutionActivated**: Specifies if any sandboxed solution or site solution is activated on the specified site collection. It MUST contain one of the following values.

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No sandboxed solution or site solution is activated on the specified site collection.</td>
</tr>
<tr>
<td>1</td>
<td>A sandboxed solution or site solution is activated on the specified site collection.</td>
</tr>
<tr>
<td>NULL</td>
<td>The stored procedure MUST determine whether any sandboxed solution or site solution is activated on the specified site collection.</td>
</tr>
</tbody>
</table>

**@RequestGuid**: The optional request identifier for the current request.
Return Code Values: An integer which MUST be 0.

Result Sets: This stored procedure MUST return the Get Web Feature List Result Set (section 3.2.4.31.1)

3.2.4.31.1 Get Web Feature List Result Set

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

```sql
FeatureId uniqueidentifier,
Version nvarchar(64),
SolutionId uniqueidentifier,
SolutionWebId uniqueidentifier,
SolutionLevel int,
Hash nvarchar(50),
Flags int;
```

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

**Version**: MUST contain the version of the feature.

**SolutionId**: If the feature was deployed using a sandboxed solution it MUST contain the identifier of the solution used to deploy the feature. Else it MUST be NULL.

**SolutionWebId**: The Site Identifier (as specified in [MS-WSSFO3] section 2.2.1.1.11) of the Site which is associated with the solution. If the no Site associated with the Solution or the feature is not deployed via a solution it MUST be NULL.

**SolutionLevel**: Must match the @SolutionLevel paramater.

**Hash**: If the feature was deployed using a solution it MUST contain the implementation-specific hash of the content of the sandboxed solution. Else it MUST be NULL.

**@Flags**: Specifies the deployed scope of the solution that contains the feature. It MUST contain one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The feature is not deployed via a solution or the solution is deployed at the farm scope.</td>
</tr>
<tr>
<td>1</td>
<td>The solution is deployed at the site collection scope.</td>
</tr>
<tr>
<td>2</td>
<td>The solution is deployed at the site scope.</td>
</tr>
</tbody>
</table>

3.2.4.32 proc_GetCustomActionsFromScope

This stored procedure retrieves the custom actions for the specified custom action identifier, site collection identifier, and site identifier. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_GetCustomActionsFromScope ( 
    @ScopeId uniqueidentifier,
    @SiteId uniqueidentifier,
);```
@WebId: uniqueidentifier
@SolutionLevel int
);

@ScopeId: MUST be the site collection identifier, site identifier, or list identifier for the custom actions being retrieved.

@SiteId: MUST be the site collection identifier for the custom actions being retrieved.

@WebId: If retrieving custom actions for the specified @SiteId, this parameter MUST be set to the NULL GUID. However, if retrieving the custom actions for a particular site within the specified @SiteId, this parameter MUST be set to a site identifier.

@SolutionLevel: Specifies the installation state of the custom action. The value MUST be an integer which is listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active</td>
<td>Solution is in installed active state.</td>
</tr>
<tr>
<td>2</td>
<td>Installing</td>
<td>Installation or upgrade of the solution is in progress.</td>
</tr>
<tr>
<td>3</td>
<td>Previous</td>
<td>Solution is from a previous installation.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The stored procedure finished successfully.</td>
</tr>
<tr>
<td>-2147467259</td>
<td>The stored procedure failed to complete successfully.</td>
</tr>
</tbody>
</table>

Result Sets:
This stored procedure MUST return the Custom Actions From Scope Result Set (section 3.2.4.32.1).

3.2.4.32.1 Custom Actions From Scope Result Set
This result set MUST return 1 row for each custom action retrieved. If there were no custom actions retrieved, this result set MUST NOT return any rows. This result set is defined in [MS-WSSFO2], section 2.2.5.2.

3.2.4.33 proc_DeleteCustomAction
This stored procedure removes the specified custom action. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_DeleteCustomAction ( 
@Id                  uniqueidentifier,
@ScopeId             uniqueidentifier,
@SiteId              uniqueidentifier,
@WebId               uniqueidentifier,
@SolutionLevel       int
@ResourceWebId       uniqueidentifier,
);```

[MS-WSSCCSP3] — v20120630
Windows SharePoint Services: Content Database Core List Schema and Site Provisioning Communications
Version 3 Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
@ScopeType: int
}

@Id: The custom action identifier that will be removed. MUST NOT be NULL.

@ScopeId: MUST be the site collection identifier, site identifier, or list identifier of the custom action that will be removed.

@SiteId: MUST be the site collection identifier in which the custom action resides.

@WebId: If the custom action being deleted is a site collection custom action, this parameter MUST be set to the NULL GUID. Otherwise this specifies the site identifier of the site in which the custom action exists.

@SolutionLevel: Specifies the installation state of the custom action being deleted.

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active</td>
<td>Solution is in installed active state.</td>
</tr>
<tr>
<td>2</td>
<td>Installing</td>
<td>Installation or upgrade of the solution is in progress.</td>
</tr>
<tr>
<td>3</td>
<td>Previous</td>
<td>Solution is from a previous installation.</td>
</tr>
</tbody>
</table>

@ResourceWebId: If the custom action being deleted is a site collection custom action, this parameter MUST be set to the site identifier of the root site of that site collection. Otherwise this specifies the site identifier of the site in which the custom action exists.

@ScopeType: MUST be the custom action scope of the custom action to be removed. MUST be one of the values listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>This specifies the custom action identifier @Id as a site collection custom action identifier. The custom action is a site collection custom action.</td>
</tr>
<tr>
<td>3</td>
<td>This specifies the custom action identifier @Id as a site custom action identifier. The custom action is a site custom action.</td>
</tr>
<tr>
<td>4</td>
<td>This specifies the custom action identifier @Id as a list custom action identifier. The custom action is a list custom action.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>If the custom action existed, it was successfully removed. If the custom action did not exist, it was not removed.</td>
</tr>
<tr>
<td>-2147467259</td>
<td>The custom action was not removed successfully.</td>
</tr>
</tbody>
</table>

Result Sets:

If the specified custom action was not removed, a result set MUST NOT be returned. However, if the specified custom action was removed, the Delete Custom Action Result Set One (section

[MS-WSSCCSP3] — v20120630
Windows SharePoint Services: Content Database Core List Schema and Site Provisioning Communications Version 3 Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
3.2.4.33.1) and Delete Custom Action Result Set Two (section 3.2.4.33.2) MUST be returned, in this order.

3.2.4.33.1 Delete Custom Action Result Set One

This result set MUST return the number of rows that are equal to the number of custom actions remaining that match the @SiteId, @WebId, and @ScopeId parameters after the custom action was removed. This result set MUST be ignored. This result set is defined using T-SQL syntax, as follows:

{UnnamedColumn} int;

{UnnamedColumn}: This column MUST be ignored.

3.2.4.33.2 Delete Custom Action Result Set Two

This result set MUST return 1 row. This result set is defined using T-SQL syntax, as follows:

ScopeCount int;
ParentScopeCount int;

ScopeCount: MUST contain the number of custom actions that exist for the specified @ScopeId, @WebId, and @SiteId.

ParentScopeCount: If the @ScopeType is set to the list custom action scope, this MUST contain the number of custom actions (including the custom action to be removed) that exist within the site as specified by @WebId. If the @ScopeType is not set to the list custom action scope, this MUST be 0.

3.2.4.34 proc_DeleteCustomActionForFeature

This stored procedure removes all custom actions that exist for the specified feature identifier, site collection identifier, site identifier, and SolutionLevel. The stored procedure is defined using T-SQL syntax, as follows:

PROCEDURE proc_DeleteCustomActionForFeature ( 
@FeatureId uniqueidentifier, 
@SiteId uniqueidentifier, 
@WebId uniqueidentifier, 
@SolutionLevel int, 
@ResourceWebId uniqueidentifier 
);

@FeatureId: The feature identifier of the custom action to be removed. This MUST NOT be NULL.

@SiteId: The site collection identifier of the custom action to be removed. This MUST NOT be NULL.

@WebId: This MUST be the site identifier of the custom action if the feature is site-scoped. If the feature is site collection-scoped then this MUST be an empty GUID. This MUST NOT be NULL.

@SolutionLevel: Specifies the installation state of the custom action to be deleted. The value MUST be one of the integers listed in the following table:
<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active</td>
<td>Solution is in installed active state.</td>
</tr>
<tr>
<td>2</td>
<td>Installing</td>
<td>Installation or upgrade of the solution is in progress.</td>
</tr>
<tr>
<td>3</td>
<td>Previous</td>
<td>Solution is from a previous installation.</td>
</tr>
</tbody>
</table>

@ResourceWebId: This MUST be the site identifier of the custom action if the feature is site-scoped. If the feature is site collection-scoped then this MUST be the site identifier of the root site of that site collection. This MUST not be NULL.

Return Code Values: An integer which MUST be 0.

Result Sets:
This stored procedure MUST NOT return a result set.

### 3.2.4.35 proc_GetWebIdOfListId

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

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

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

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

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

#### 3.2.4.35.1 WebId Result Set

This WebId Result Set returns the site identifier for the site containing the list specified by `@ListId`. If the list specified by `@ListId` exists, the result set MUST contain 1 row; otherwise, it MUST contain 0 rows. The WebId Result Set is defined using T-SQL syntax, as follows:

```sql
tp_WebId    uniqueidentifier;
```

tp_WebId: The site identifier for the site that contains the list specified by the `@ListId`.

### 3.2.4.36 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,
    @RequestGuid      uniqueidentifier = NULL OUTPUT
};

@WebSiteId: The site collection 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 of usage data being requested. The parameter MUST be in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Specifies that monthly usage data is being requested.</td>
</tr>
<tr>
<td>Not 1</td>
<td>Specifies that daily usage data is being requested.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>Failed execution.</td>
</tr>
</tbody>
</table>

Result Sets: MUST return the Monthly Usage Result Set when the @BlobType parameter is 1, otherwise it MUST return the Daily Usage Result Set when the @BlobType parameter is not 1.

3.2.4.36.1 Monthly Usage Result Set

The Monthly Usage Result Set returns usage data 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:

```
MONTHLYUSAGDATA    image,
MONTHLYUSAGDATAVERSION  int;
```

**MONTHLYUSAGDATA**: 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.

**MONTHLYUSAGDATAVERSION**: 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.
3.2.4.36.2 Daily Usage Result Set

The Daily Usage Result Set returns usage data 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:

```sql
CREATE TABLE DailyUsageData (image, DailyUsageDataVersion int);
```

**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.

3.2.4.37 proc_HeartbeatSiteUpgradeSession

The `proc_HeartbeatSiteUpgradeSession` stored procedure is called to refresh the lock on the site collection on the back-end database that was acquired via `proc_AcquireSiteUpgradeSession`. The front-end Web server periodically calls this stored procedure because the lock expires in 60 seconds. The stored procedure is defined using T-SQL syntax, as follows:

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

**@SiteId**: The site collection identifier for the site collection.

**@SessionId**: A unique identifier that was previously associated with this site collection by a call to `proc_AcquireSiteUpgradeSession`.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>1</td>
<td>Failed. The site collection is associated with a different unique identifier.</td>
</tr>
</tbody>
</table>

**Result Sets**: MUST NOT return any result sets.

3.2.4.38 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:

```sql
PROCEDURE proc_IsContentTypeGhosted ()
```
@SiteId (uniqueidentifier),
@Class (tinyint),
@ContentTypeId (varbinary(512)),
@RequestGuid (uniqueidentifier = NULL OUTPUT)
);

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

@Class: This MUST be set to 1.

@ContentTypeId: The content type identifier of the content type to be checked. This MUST be of type [ContentTypeId].

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

### 3.2.4.38.1 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, 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:

```sql
{IsGhosted} int;
```

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The content type specified by @ContentTypeId is customized.</td>
</tr>
<tr>
<td>1</td>
<td>The content type specified by @ContentTypeId is uncustomized.</td>
</tr>
</tbody>
</table>

### 3.2.4.39 proc_IncrementSiteClientTag

The proc_IncrementSiteClientTag stored procedure is called to increment the application file cache version for files in the site collection. The T-SQL syntax for the stored procedure is as follows:

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

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

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

Return values: An integer that MUST be zero.

Result sets: MUST NOT return any result sets.
3.2.4.40 proc_IsContentTypeInUseInList

The proc_IsContentTypeInUseInList stored procedure is called to determine whether there are list items with the specified list content type in the specified list. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_IsContentTypeInUseInList (  @SiteId           uniqueidentifier,  @ListId           uniqueidentifier,  @ContentTypeId    tContentTypeId,  @RequestGuid      uniqueidentifier = NULL OUTPUT  );
```

@SiteId: The site collection identifier for the site collection that contains the list 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.

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

Return Code Values: The stored procedure MUST return 1 if there are any list items with the specified list content type in the specified list. This MUST NOT include list items that have been deleted and MUST NOT include list item versions that are not the current version. If there are no such list items, or if @SiteId is NULL or if @ListId is NULL or if @ContentTypeId is NULL, then the stored procedure MUST return 0.

Result Sets: MUST NOT return any result sets.

3.2.4.41 proc_IsFieldTemplateUsedInContentTypeTemplate

The proc_IsFieldTemplateUsedInContentTypeTemplate stored procedure is called to determine whether there are any content types in the specified site collection which use the specified site column. The stored procedure is defined using T-SQL syntax, as follows:

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

@SiteId: The site collection identifier for the site collection that contains the content types being examined.

@FieldId: The GUID for the site column.

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

Return values: The stored procedure MUST return 1 if there are any content types in the site collection which use the site column as specified by @FieldId. Otherwise, the stored procedure MUST return 0.
Result Sets: MUST NOT return any result sets.

3.2.4.42 proc_ListAllFileUrls

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

```sql
PROCEDURE proc_ListAllFileUrls(
    @SiteId           uniqueidentifier,
    @WebUrl           nvarchar(260),
    @IncludeGhosts    bit,
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);
```

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

@WebUrl: The store-relative URL to the site. The length of the parameter @WebUrl SHOULD NOT exceed 260 characters; otherwise only the first 260 characters will be used.

@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 0. When uncustomized documents in the site are also requested, the @IncludeGhosts flag MUST be set to 1.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site specified by the @WebUrl parameter does not exist in the specified site collection.</td>
</tr>
</tbody>
</table>

Result Sets: MUST return the following result set if the site specified by the @WebUrl parameter exists in the specified site collection.

3.2.4.42.1 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, then the result set MUST also return 1 row for each ghosted document. If the @IncludeGhosts parameter is set to 0, then 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:

```sql
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.
3.2.4.43 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:

```sql
PROCEDURE proc_ListAllWebsOfSite (
    @SiteId uniqueidentifier,
    @Collation nvarchar(32),
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

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

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

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.2.4.43.1 SiteWebs Result Set

SiteWebs result set 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:

```sql
FullUrl nvarchar(256),
Id uniqueidentifier,
{ParentWebFullUrl} nvarchar(256),
Language int,
{Title} nvarchar(255),
UIVersion tinyint,
Flags int,
WebTemplate int,
ProvisionConfig smallint,
MasterUrl nvarchar,
CustomMasterUrl nvarchar;
```

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

Id: Contains the site identifier of the site.

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

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

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

UIVersion: A number that represents the visual state of the site.

Flags: A site collection flag value that indicates the settings for the site collection that contains this site.
**WebTemplate**: The identifier of the site definition that contains the site definition configuration used to provision a site.

**ProvisionConfig**: The identifier of the site definition configuration used to provision a site.

**MasterUrl**: The store-relative URL of the master page for a site.

**CustomMasterUrl**: The store-relative URL of custom master page for a site.

### 3.2.4.44 proc_ListChildWebs

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

```sql
PROCEDURE proc_ListChildWebs (  
    @SiteId uniqueidentifier,  
    @WebUrl nvarchar(256),  
    @Collation nvarchar(32),  
    @RequestGuid uniqueidentifier = NULL OUTPUT  
);
```

- **@SiteId**: The site collection identifier for the site collection from which the child sites are to be retrieved.
- **@WebUrl**: The store-relative URL of the site from which the child sites are to be retrieved.
- **@Collation**: The collation to be used to order the result set.
- **@RequestGuid**: The optional request identifier for the current request.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The specified @WebUrl was not found for the given @SiteId.</td>
</tr>
</tbody>
</table>

**Result Sets**: MUST return the following result set:

### 3.2.4.44.1 ChildWebs Result Set

ChildWebs result set 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:

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

- **FullUrl**: Contains the store-relative 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.

3.2.4.45 proc_ListChildWebsFiltered

The proc_ListChildWebsFiltered stored procedure is called to get a filtered list of data about child sites for a specified parent site; the list can 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,
    @Collation                 nvarchar(48) = '',
    @WebTemplate               int = NULL,
    @ProvisionConfig           smallint = NULL,
    @ToLinkRecurringMeeting    bit = NULL,
    @RequestGuid               uniqueidentifier = NULL OUTPUT
);
```

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

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

@Collation: The Windows collation name that is used to sort the filtered list of child sites. This MAY be NULL. If @Collation is not NULL, then the filtered list of child sites MUST be sorted using this value.

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

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

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

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.2.4.45.1 List Child Webs Filtered Result Set

The List Child Webs Filtered Result set contains a filtered list of child sites (2) for the specified parent site (2). This list can be sorted based on the value of the specified @Collation parameter.

The List Child Webs Filtered result set MUST return one row of data associated with each child site (2) if it satisfies all of the following conditions:

- It is a child site (2) of the specified parent site (2).
- @WebTemplate is NULL.

Or
@WebTemplate is not NULL, and the value of the identifier of the site definition used to provision this child site (2) 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 child site (2) 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 child site (2).

• @ToLinkRecurringMeeting is NULL

Or

@ToLinkRecurringMeeting is set to 1 and the child site (2) is a Meeting Workspace site and it has no meetings configured.

Or

@ToLinkRecurringMeeting is set to 0 and the child site (2) 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 cannot be sorted. Otherwise, the result set MUST be sorted on the site title of the child sites (2) based on the collation specified by the value of @Collation.

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

```sql
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 URL of the child site (2).

**Id**: Contains the site identifier of the child site (2).

**{Title}**: The site title of the child site (2). If the site title is NULL, the empty string MUST be returned.

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

**Language**: The language code identifier (LCID) of the child site (2).
WebTemplate: The identifier of the site definition that contains the site definition configuration used to provision the child site (2).

ProvisionConfig: The identifier of the site definition configuration used to provision this child site (2).

MeetingCount: If this child site (2) is a meeting workspace site, this value indicates the number of meetings that are configured. Otherwise, this value MAY return zero. The front-end Web server MUST ignore this value for child sites (2) that are not meeting workspaces.

{Acl}: The binary serialization of the ACL (see [MS-WSSFO2], section 2.2.4.6) for this child site (2). This value MUST be NULL if the child site (2) inherits security settings from its parent site.

AnonymousPermMask: Contains a 64-bit mask that specifies the permissions granted to anonymous users in this child site (2). The bit mask values are defined in [MS-WSSFO2], section 2.2.2.13.

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 child site (2). This MUST be NULL for a child site (2) using the default security implementation.

TimeCreated: The time this child site (2) was created. This MUST be in UTC format.

{TimeListLastModified}: The last time any list contained in this child site (2) was modified. This MUST be in UTC format.

3.2.4.46 proc_ListContentTypeInUse

The proc_ListContentTypeInUse stored procedure is called to list the usage of a specific content type. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_ListContentTypeInUse (
    @SiteId uniqueidentifier,
    @ContentTypeId varbinary(512),
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

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

@ContentTypeId: The content type identifier for which usage will be determined. This MUST be of type tContentTypeId.

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

Return Code Values: An integer which MUST be 0.

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

3.2.4.46.1 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: Contains the content type identifier 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 URL of the site or list root folder to which this content type is registered.

3.2.4.46.2 Content Type List Usage Result Set

Returns all content types that are used in lists in the specified site collection that are of the specified content type or content types that are 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 used in each list and is the content type specified by @ContentTypeId or for each content type that is a descendant content type of the content type specified by @ContentTypeId. The result set is defined using T-SQL syntax, as follows:

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

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

{Scope}: Contains a store-relative URL of the root folder of the list in which this content type is used.

3.2.4.47 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:

```sql
PROCEDURE proc_ListContentTypesInWeb(
  @SiteId uniqueidentifier,  
  @Class tinyint,  
  @Scope nvarchar(256),  
  @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

@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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Site column.</td>
</tr>
<tr>
<td>1</td>
<td>Site content type.</td>
</tr>
</tbody>
</table>

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

@RequestGuid: The optional request identifier for the current request.
**Return Code Values:** An integer which MUST be 0.

**Result Sets:** MUST return the following result set:

### 3.2.4.47.1 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 earlier, the result set MUST be List Content Types Result Set defined in Section 2.2.4.1.

### 3.2.4.48 proc_ListContentTypesInWebRecursive

The `proc_ListContentTypesInWebRecursive` stored procedure is called to list either all of the site content types or all of the site columns in a site. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_ListContentTypesInWebRecursive(
    @SiteId           uniqueidentifier,
    @Class            tinyint,
    @Scope            nvarchar(256),
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);
```

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

**@Class:** The type of return values. The parameter MUST be in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Site column.</td>
</tr>
<tr>
<td>1</td>
<td>Site content type.</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>144</td>
<td><code>@Scope</code> refers to a site that is not within the site collection designated by the <code>@SiteId</code> parameter.</td>
</tr>
</tbody>
</table>

**Result Sets:**
If @Class is equal to 0 then the stored procedure returns a list of all site columns registered in the site and all of the site’s ancestors as 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 and all of its ancestors. This result set MUST contain 1 row for each site content type registered in the site.

For both values of @Class listed earlier, the result set MUST be a List Content Types Result Set, as defined in Section 2.2.4.1.

3.2.4.49 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:

```sql
PROCEDURE proc_ListDerivedContentTypes(
    @SiteId           uniqueidentifier,
    @ContentTypeId    varbinary(512),
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);
```

@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.

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

Return Code Values: An integer which MUST be 0.

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

3.2.4.49.1 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 MUST be a List Content Types Result Set (Section 2.2.4.1).

3.2.4.49.2 Derived Content Types Result Set

Returns a list of content types associated with lists 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:

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

ContentTypeId: Contains a content type identifier. This MUST be of type tContentTypeId.

WebId: Contains a site identifier that specifies the site to which the content type is registered.
**ListId:** Contains a list identifier that specifies the list to which the content type is registered.

### 3.2.4.50  proc_ListsUsingFieldTemplate

The **proc_ListsUsingFieldTemplate** stored procedure is called to get a list of lists in a site collection which includes a specific field. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_ListsUsingFieldTemplate(
    @SiteId          uniqueidentifier,
    @FieldId         varbinary(512),
    @BaseTypes       int,
    @RequestGuid     uniqueidentifier = NULL OUTPUT
);
```

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

**@FieldId:** The field identifier of the field.

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

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

**Return Code Values:** An integer which MUST be 0.

**Result Sets:** MUST return the following result set:

#### 3.2.4.50.1 Lists Using Field Result Set

Lists Using Field Result Set contains a list of GUID pairs for the sites and lists under a site collection using a particular field. The Lists Using Field Result Set MUST contain 1 row per list which uses the specified field. The Lists Using Field Result Set is defined using T-SQL syntax, as follows:

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

**WebId:** The site identifier containing the list specified by `@ListId`.

**ListId:** The list identifier of the list which uses the field specified by `@FieldId`.

### 3.2.4.51  proc_ListUnghostedFieldTemplatesInList

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

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

Preliminary
@SiteId: The site collection identifier in which the specified list exists.

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

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

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.2.4.51.1 Unghosted List Fields Result Set

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

```sql
{FieldId} tinyint,
Definition ntext;
```

FieldId: The field identifier of the field.

Definition: The XML fragment of the field. The XML schema for this structure is defined in [MS-WSSFO2] section 2.2.8.3.3.

3.2.4.52 proc_MakeViewDefaultForContentType

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

```sql
PROCEDURE proc_MakeViewDefaultForContentType(
    @ListId uniqueidentifier,
    @ViewId uniqueidentifier,
    @ContentTypeId varbinary(512),
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

ListId: A list identifier for the list 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 content type identifier of a descendant content type from the folder content type. This MUST be of type tContentTypeId.

RequestGuid: The optional request identifier for the current request.

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.
3.2.4.53  proc_MakeViewDefaultForList

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

```sql
PROCEDURE proc_MakeViewDefaultForList (  
@SiteId           uniqueidentifier,  
@ListId           uniqueidentifier,  
@ViewId           uniqueidentifier,  
@RequestGuid      uniqueidentifier = NULL OUTPUT
);
```

@SiteId: The site identifier of the site that contains the list whose default list view will be set.

@ListId: The list identifier of the list 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.

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.4.54  proc_MakeViewMobileDefaultForList

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

```sql
PROCEDURE proc_MakeViewMobileDefaultForList (  
@ListId           uniqueidentifier,  
@ViewId           uniqueidentifier,  
@RequestGuid      uniqueidentifier = NULL OUTPUT
);
```

@ListId: The list identifier of the list 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.

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.4.55  proc_MapContentTypeToList

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

```sql
PROCEDURE proc_MapContentTypeToList (  
@SiteId           uniqueidentifier,
@ViewId           uniqueidentifier,
@RequestGuid      uniqueidentifier - NULL OUTPUT
);
```

@SiteId: The site identifier of the site that contains the list whose default list view will be set.

@ListId: The list identifier of the list 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.

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.
@SiteId: The site collection identifier of the site collection in which the list resides. This MUST NOT be NULL.

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

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

@ContentTypeId: The content type identifier of the content type or the binary representation of the field identifier of the field being associated. This MUST be of type tContentTypeId. This MUST NOT be NULL.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>@ContentTypeId refers to a field, and it MUST be 16 bytes long.</td>
</tr>
<tr>
<td>1</td>
<td>@ContentTypeId refers to a content type.</td>
</tr>
</tbody>
</table>

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

Return Code Values: An integer return code which MUST be one of the values in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>30</td>
<td>There was an IO error.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.56 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,
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);
```
@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.

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

Return Code Values: An integer return code which MUST be one of the values in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>30</td>
<td>There was an IO error.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.57 proc_MapUrlToListAndView

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

```sql
PROCEDURE proc_MapUrlToListAndView(
    @SiteID           uniqueidentifier,
    @WebID            uniqueidentifier,
    @DirName          nvarchar(256),
    @LeafName         nvarchar(128),
    @RequestGuid      uniqueidentifier = NULL OUTPUT
);
```

@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 list view page.

@LeafName: The leaf name of the list view page.

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.2.4.57.1 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 list view page. The result set MUST return 1 row if the specified view page exists. If the specified list...
The view page does not exist then the result set MUST return 0 rows. The Map URL to List and View Result Set is defined using T-SQL syntax, as follows:

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

**tp_Id:** MUST contain the view identifier of the view for the List view page specified by the @DirName and @LeafName parameters.

**tp_ListId:** MUST contain the list identifier of the list which contains the view specified by tp_Id.

### 3.2.4.58 proc_MapV2FieldToList

The `proc_MapV2FieldToList` stored procedure is called to associate a field to a list. The stored procedure is defined using T-SQL syntax, as follows:

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

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

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

### 3.2.4.59 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) 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 finished.

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

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

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

**@WebId:** The site identifier of the site that will be set as provisioned.
@RequestGuid: The optional request identifier for the current request.

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST return two Event Receivers Result Sets (see [MS-WSSFO3] section 2.2.4.11) in the following order, the first MUST be the Event Receivers Result Set for the site that is specified by @WebId, and the second MUST be the Event Receivers Result Set for its parent site.

3.2.4.60 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:

```sql
PROCEDURE proc_MergeWeb(
    @WebSiteId uniqueidentifier,
    @WebUrl nvarchar(260),
    @ThresholdRowCount int,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

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

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

@ThresholdRowCount: The maximum number of list items allowed to be touched by this operation (for example, if the user performing the operation has a limit imposed because of permissions), or 0 for no limit. This is for performance issues.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site to be converted does not exist in the specified site collection.</td>
</tr>
<tr>
<td>5</td>
<td>The site to be converted is a top-level site and cannot be converted.</td>
</tr>
<tr>
<td>36</td>
<td>The number of list items in the site to be converted exceeds @ThresholdRowCount. The operation failed.</td>
</tr>
<tr>
<td>133</td>
<td>The site to be converted contains 1 or more lists or document libraries. The operation failed.</td>
</tr>
</tbody>
</table>

Result Sets: MUST return the following result set:

3.2.4.60.1 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 result set MUST be an Audit Mask Result Set, [MS-WSSFO2], section 3.1.5.7.4 - Audit Mask Result Set.
3.2.4.61  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:

```sql
PROCEDURE proc_MiniSproc(
    @WebSiteId uniqueidentifier,
    @WebId uniqueidentifier,
    @Url nvarchar(260),
    @DGCacheVersion bigint,
    @SystemId varbinary(512) = NULL,
    @AppPrincipalName nvarchar(256),
    @IsHostHeaderAppPrincipalName bit,
    @RequestGuid uniqueidentifier = NULL OUTPUT
)
```

@WebSiteId: The site collection 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 the 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 site specified by @WebId. If this is an invalid store-relative URL (including NULL) the procedure MUST 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. 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.

@AppPrincipalName: The app principal identifier or the app web domain identifier associated with the request.

@IsHostHeaderAppPrincipalName: 1 indicates that the @AppPrincipalName parameter is the app web domain identifier. 0 indicates that the @AppPrincipalName parameter is the app principal identifier.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>2</td>
<td>The document specified by @Url is invalid or is not readable by the specified user, or the @SystemId is invalid (is NULL or does not specify an existing user).</td>
</tr>
<tr>
<td>3</td>
<td>The site identifier derived from @Url does not exist in the given site collection identifier.</td>
</tr>
<tr>
<td>1168</td>
<td>@WebSiteId does not specify a valid site collection identifier.</td>
</tr>
<tr>
<td>1271</td>
<td>Access to this site is blocked.</td>
</tr>
</tbody>
</table>
**Result Sets**: MUST return 0 to 6 of the following 7 result sets: Site URL Result Set (section 3.2.4.61.1), Domain Group Cache Versions Result Set (section 3.2.4.61.2), Domain Group Cache Back-End Database Server Update Result Set (section 3.2.4.61.3), Domain Group Cache Front-End Web Server Update Result Set (section 3.2.4.61.4), Site Metadata Result Set (section 3.2.4.61.5), Event Receivers Result Set (section 3.2.4.61.6), User Document Security Context Result Set (section 3.2.4.61.7).

### 3.2.4.61.1 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 contain 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 the parameter @Url. If the parameter @Url is not contained in site collection this MUST be the top-level site.

### 3.2.4.61.2 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], sections 2.2.6.4 - Domain Group Cache Versions Result Set for details.

### 3.2.4.61.3 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, @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.

### 3.2.4.61.4 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, @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.

### 3.2.4.61.5 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 1 row if the site is found; otherwise 0 rows MUST be returned.

See the [MS-WSSFO2], section 2.2.5.23 - Site Metadata Result Set, for details.
3.2.4.61.6 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 3.1.5.7.3 - Event Receivers Result Set, for schema details.

3.2.4.61.7 User Document Security Context Result Set


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 integer 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 identifier for the who published the document as a draft. This value MUST be 0 if the document does not exist or is not a draft.

{Lists_Flags}: A big integer value that specifies the list flags (see [MS-WSSFO2], section 2.2.2.5 - List Flags) on the list which contains the document as a list item. If the document is not a list item in a list, this 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 rights mask (see [MS-WSSFO2], section 2.2.2.14) indicating the rights granted to an anonymous user, or to a user who has no specific rights to the document.

{Level}: A tiny integer 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, otherwise it MUST be 0. This value MUST NOT be NULL.

3.2.4.62 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,
    @WebId            uniqueidentifier,
    @Scope            nvarchar(256),
    @Class            tinyint,
    @NextChildByte    tinyint,
    @Override         bit,
    @ContentTypeId    varbinary(512),
    @ResourceDir      nvarchar(128) = NULL,
    @RequestGuid      uniqueidentifier = NULL OUTPUT
)

@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.

@WebId: The site identifier of the site in which the content type is to be made available. If the value is NULL, the top-level site will be used.

@Scope: The store-relative 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.

@Override: If @Class is equal to 0, and this value is equal to 1, then this site column will replace any site column with the same content type identifier in the scope specified by the @Scope parameter, unless that site column was deployed via a feature. If @Class is equal to 1, this value MUST be 0. This MUST NOT be NULL.

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

@ResourceDir: The store-relative 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.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful execution.</td>
</tr>
<tr>
<td>29</td>
<td>There was an IO error.</td>
</tr>
<tr>
<td>80</td>
<td>The site content type or site column is already available in the scope.</td>
</tr>
</tbody>
</table>

Result Sets: If the return value is 0 or 29, this stored procedure MUST NOT return any result sets. If the return value is 80, MUST return the following result set:
3.2.4.62.1 Content Type Exists Result Set

This result set is returned when a site content type or site column exists in the scope specified by @Scope with the content type identifier specified by @ContentTypeId. This result set MUST contain exactly 1 row. The result set is defined using T-SQL syntax, as follows:

```
Class            tinyint,
ContentTypeId    varbinary(512);
```

**Class:** Contains the value @Class.

**ContentTypeId:** Contains the value @ContentTypeId. This MUST be of type tContentTypeId.

3.2.4.63 proc_ReleaseSiteUpgradeSession

The proc_ReleaseSiteUpgradeSession stored procedure is called to finish a site collection upgrade. This stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_ReleaseSiteUpgradeSession ( 
    @SiteId               uniqueidentifier, 
    @SessionId            uniqueidentifier, 
    @UpgradeStatus        int, 
    @RequestGuid          uniqueidentifier = NULL OUTPUT);
```

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

**@SessionId:** A unique identifier that was previously associated with this site collection by a call to proc_AcquireSiteUpgradeSession.

**@UpgradeStatus:** An integer which MUST be one of the values in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Indicates that the site collection upgrade finished with error.</td>
</tr>
<tr>
<td>3</td>
<td>Indicates that the site collection upgrade finished without error.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>1</td>
<td>Failure. The site collection is associated with a different unique identifier.</td>
</tr>
</tbody>
</table>

**Result Sets:** MUST NOT return any result sets.
3.2.4.64  proc_RemoveFromSiteUpgradeQueue

The proc_RemoveFromSiteUpgradeQueue stored procedure is called to remove the site collection upgrade request if it has not yet started. The stored procedure is defined using T-SQL syntax, as follows:

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Failure. Either the site collection upgrade was never requested for this site collection by using proc_AddToSiteUpgradeQueue, or the site collection upgrade has started by a call to proc_AcquireSiteUpgradeSession.</td>
</tr>
<tr>
<td>1</td>
<td>Successful completion. The site collection upgrade request for the site collection is removed.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.65  proc_RenameListItemContentType

The proc_RenameListItemContentType stored procedure is called to set the content type display name field for the list items, documents or folders in a particular list 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),
    @RequestGuid              uniqueidentifier = NULL OUTPUT
);
```

@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, section 2.2.3.1.

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

@RequestGuid: The optional request identifier for the current request.
Return Code Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.4.66 proc_ResolveWikiLinkItem

The proc_ResolveWikiLinkItem stored procedure is called to determine the URL to a list item. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_ResolveWikiLinkItem(
    @SiteId         uniqueidentifier,
    @LinkId         int,
    @ListId         uniqueidentifier,
    @ItemId         int,
    @RequestGuid    uniqueidentifier = NULL OUTPUT
);
```

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

@LinkId: This parameter is passed back in the Resolve Wiki Link Item Result Set result set unmodified, and otherwise ignored.

@ListId: The GUID of the list containing the list item specified by @ItemId.

@ItemId: the ID of the list item within the list specified by @ListId.

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

Return Code Values: An integer which MUST be 0.

Result Sets: MUST return the following result set:

3.2.4.66.1 Resolve Wiki Link Item Result Set

The Resolve Wiki Link Item Result Set is defined using T-SQL syntax, as follows:

```sql
LinkId      int;
ResolvedUrl nvarchar(V_STORE_MAX_FULLURL);
```

LinkId: This MUST be the same value as @LinkId.

ResolvedUrl: The URL to the list item indicated by @ItemId in the list indicated by @ListId, if the list item exists. Otherwise, it MUST be NULL.

3.2.4.67 proc_RetrieveSiteHealthCheckResults

The proc_RetrieveSiteHealthCheckResults stored procedure is called to retrieve results stored using the proc_StoreSiteHealthCheckResults (section 3.2.4.77) stored procedure. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_RetrieveSiteHealthCheckResults(
    @SiteId uniqueidentifier,
    @RuleId uniqueidentifier
);
```
@SiteId: The site collection identifier of the site collection on which the health rule was run.

@RuleId: The GUID specifying the health rule that was run.

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST return the following result set. If results for the specified health rule on the specified site collection have been stored, the result MUST contain 1 row. Otherwise, the result set must contain 0 rows.

3.2.4.67.1 Retrieve Site Health Check Results Result Set

The Retrieve Site Health Check Results Result Set is defined using T-SQL syntax, as follows:

```
Status tinyint,
Message varchar(max),
TimeStamp datetime
```

Status: The success or failure mode of the health rule that was run. MUST be one of the values in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Specifies that the site collection health rule passed. Site collection upgrade is not blocked by this site collection health rule.</td>
</tr>
<tr>
<td>1</td>
<td>Specifies that the site collection health rule failed with a warning. Site collection upgrade is not blocked by this site collection health rule.</td>
</tr>
<tr>
<td>2</td>
<td>Specifies that the site collection health rule failed with an error. Site collection upgrade is blocked by this site collection health rule.</td>
</tr>
</tbody>
</table>

Message: A message describing the results from running the health rule. This MUST NOT be NULL and MUST be an XML Document conforming to the format specified in section 2.2.6.3.2.

TimeStamp: The date and time when the health rule was run. This MUST NOT be NULL.

3.2.4.68 proc_RetrieveSiteUpgradeSession

The proc_RetrieveSiteUpgradeSession stored procedure is called to retrieve information about a site collection upgrade for a site collection. The stored procedure is defined using T-SQL syntax, as follows:

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

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

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

Return Code Values: An integer which the protocol client MUST ignore.
Result Sets: MUST return a site collection upgrade info result set as defined in section 2.2.4.2. The result set MUST contain 0 or 1 rows.

3.2.4.69 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. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_SetListFormToUrl(
    @SiteId         uniqueidentifier,
    @WebId          uniqueidentifier,
    @ListId         uniqueidentifier,
    @PageUrl        nvarchar(260),
    @PageType       tinyint,
    @RequestGuid    uniqueidentifier = NULL OUTPUT
);
```

@SiteId: The site collection identifier of the site collection containing the site that contains the list specified by @ListId.

@WebId: This parameter MUST be ignored.

@ListId: The list identifier of the list containing the form specified by the @PageUrl parameter.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>The form specified by the @PageUrl parameter is a display form.</td>
</tr>
<tr>
<td>6</td>
<td>The form specified by the @PageUrl parameter is an edit form.</td>
</tr>
<tr>
<td>8</td>
<td>The form specified by the @PageUrl parameter is a new form.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>126</td>
<td>This return code MUST be returned if any of the following conditions are met:</td>
</tr>
<tr>
<td></td>
<td>- The URL specified by the @PageUrl parameter does not exist for the site collection specified by the @SiteId parameter.</td>
</tr>
<tr>
<td></td>
<td>- The specified list does not already contain a form matching the specified @PageType parameter.</td>
</tr>
<tr>
<td></td>
<td>- The Web Part Page specified by @PageUrl is a customized (1) Web Part Page.</td>
</tr>
<tr>
<td>Value</td>
<td>Description</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
</tr>
<tr>
<td>127</td>
<td>This return code MUST be returned if any of the following conditions are met:</td>
</tr>
<tr>
<td></td>
<td> More than one URL specified by the @PageUrl parameter exists matching the specified @PageType parameter for the specified list.</td>
</tr>
<tr>
<td></td>
<td> The URL specified by the @PageUrl parameter does not match the specified @PageType parameter.</td>
</tr>
</tbody>
</table>

**Result Sets:** MUST NOT return any result sets.

### 3.2.4.70 proc_SetSitePlatformVersion

The **proc_SetSitePlatformVersion** stored procedure is called to set the platform version, an implementation-specific string that represents the most current version (for example 4.0.25.0 or 15.0.10.0) of upgrade actions that have been performed in site collection upgrade, to a site collection. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_SetListFormToUrl(
    @SiteId uniqueidentifier,
    @Version nvarchar(60),
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

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

@Version: The version string to set to the platform version value of the site collection.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>1249</td>
<td>This return code MUST be returned if any error happened during the execution.</td>
</tr>
</tbody>
</table>

**Result Sets:** MUST NOT return any result sets.

### 3.2.4.71 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:

```sql
PROCEDURE proc_SetListFormToUrl(
    @SiteId uniqueidentifier,
    @bitsToSet int,
    @bitMask int,
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

@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 2.2.1.9 - Site Collection Flags.

@bitMask: Specifies the mask of bits to set.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site collection specified by @SiteId was not found</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.72   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),
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

@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.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site collection specified by @WebSiteId was not found</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.73   proc_SetSiteProps

The proc_SetSiteProps stored procedure is called to set owners for a site collection. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_SetSiteProps(
    @SiteId                   uniqueidentifier,
    @OwnerID                  int,
    @RequestGuid              uniqueidentifier - NULL OUTPUT
);
```

Preliminary
@SecondaryContactID int,
@RequestGuid uniqueidentifier = NULL OUTPUT
);

@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 owner for the site collection.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site collection specified by @SiteId was not found.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.74 proc_SetTpView

The proc_SetTpView stored procedure is called to set the view information for a given view. The stored procedure is defined using T-SQL syntax, as follows:

PROCEDURE proc_SetTpView (
    @SiteId          uniqueidentifier,
    @ViewXml         tCompressedBinary,
    @ViewFlags       int,
    @ViewId          uniqueidentifier,
    @Type            int,
    @DisplayName     nvarchar(255)
);

@SiteID: The site identifier of the site collection that contains the view to be set.

@ViewXml: Information about the view to be set. This information is a compressed query expressed in Collaborative Application Markup Language (CAML). See [MS-WSSCAML] for more information about CAML. The query is compressed by the algorithm specified in [RFC1950].

@ViewFlags: The view flags of the view to be updated. Refer to [MS-WSSFO2] section 2.2.1.11 for valid values.

@ViewId: The view identifier of the view to be set.

@Type: The page type of the view to be updated. Refer to [MS-WSSFO2] section 2.2.2.14 for valid values.

@DisplayName: The display name of the view to be set.

Return values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.
3.2.4.75 proc_SetWebMetainfo

The proc_SetWebMetainfo stored procedure is called to set metadata information for a specified site. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_SetWebMetainfo(
    @WebSiteId               uniqueidentifier,
    @WebUrl                  nvarchar(260),
    @MetaInfo                varbinary(max),
    @Flags                   int,
    @DefTheme                nvarchar(64),
    @IncrementSiteTimeStamp  bit,
    @MasterUrl               nvarchar(260),
    @CustomMasterUrl         nvarchar(260),
    @@WebId                  uniqueidentifier OUTPUT,
    @RequestGuid             uniqueidentifier = NULL OUTPUT
);`

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

@WebUrl: The store-relative URL of the site for which the metadata is being set.

@MetaInfo: A metadict for the site.

@Flags: The site property flags value specified by [MS-WSSFO2], section 2.2.2.10 for the site for which the metadata is being set.

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

@IncrementSiteTimeStamp: If the bit is 1, the last modified timestamp for the site collection MUST be updated.

@MasterUrl: The store-relative URL of the master page for the specified site.

@CustomMasterUrl: The store-relative URL of the custom master page for the specified site.

@@WebId: The site identifier of the site returned to caller based on the @WebUrl parameter.

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

Return Code Values: An integer which MUST be one of the integers in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site collection specified by @WebSiteId or @WebUrl was not found, or a given @WebUrl is not a top-level site.</td>
</tr>
<tr>
<td>212</td>
<td>When adding content to the site is prevented or accessing to the site is blocked.</td>
</tr>
<tr>
<td>1816</td>
<td>The site collection quota for the site collection has been exceeded.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.
3.2.4.76 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:

```sql
PROCEDURE proc_SetWebUsageData(
    @WebSiteId    uniqueidentifier,
    @WebUrl       nvarchar(260),
    @BlobTypeToUpdate int,
    @UsageData    varbinary(max),
    @BWUsed       bigint,
    @UsageDataVersion int,
    @DayLastAccessed smallint,
    @RequestGuid  uniqueidentifier = NULL OUTPUT
);
```

@WebSiteId: The site collection 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 URL of the site for which usage data is being stored. This parameter MUST NOT be NULL.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Specifies that daily usage data is being stored.</td>
</tr>
<tr>
<td>Not 0</td>
<td>Specifies that monthly usage data is being stored.</td>
</tr>
</tbody>
</table>

@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. It MUST be 0 if @BlobTypeToUpdate is different than 0. This parameter MUST NOT be NULL.

@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.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site doesn't exist.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.
3.2.4.77 proc_StoreSiteHealthCheckResults

The proc_StoreSiteHealthCheckResults stored procedure is called to store the results of running a site collection health rule. These health rules perform an implementation-independent check on the health of a site collection. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_StoreSiteHealthCheckResults
    @SiteId uniqueidentifier,
    @RuleId uniqueidentifier,
    @Status tinyint,
    @Message varchar(max)
);
```

@SiteId: The site collection identifier of the site collection on which the health rule was run.

@RuleId: The GUID specifying the health rule that was run. MUST NOT be Guid.Empty.

@Status: The success or failure mode of the health rule that was run. MUST be one of the values in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Specifies that the site collection health rule passed. Site collection upgrade is not blocked by this site collection health rule.</td>
</tr>
<tr>
<td>1</td>
<td>Specifies that the site collection health rule failed with a warning. Site collection upgrade is not blocked by this site collection health rule.</td>
</tr>
<tr>
<td>2</td>
<td>Specifies that the site collection health rule failed with an error. Site collection upgrade is blocked by this site collection health rule.</td>
</tr>
</tbody>
</table>

@Message: A message describing the result of running the health rule. MUST NOT be NULL and MUST be an XML Document conforming to the format specified in section 2.2.6.3.2.

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST NOT return any result sets.

3.2.4.78 proc_StoreUserInfoListInfo

The proc_StoreUserInfoListInfo stored procedure is called to establish the row and column of the UserData View ([MS-WSSFO2], section 2.2.7.8 - UserData View) that will store the user activity status for the specified list. The stored procedure is defined using T-SQL syntax, as follows:

```
PROCEDURE proc_StoreUserInfoListInfo
    @SiteId uniqueidentifier,
    @ListId uniqueidentifier,
    @RowOrdinal int,
    @ColName nvarchar(64),
    @RequestGuid uniqueidentifier = NULL OUTPUT
);
```

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

@ListId: The list identifier of the list.
@RowOrdinal: A zero-based index integer number that identifies the row of the UserData View ([MS-WSSFO2], section 2.2.7.8 - UserData View) in which the user activity status will be stored for the specified list.

@ColName: The column name of the UserData View ([MS-WSSFO2], section 2.2.7.8 - UserData View) in which the user activity status will be stored for the specified list.

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

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST NOT return any result sets.

3.2.4.79 proc_UnmapContentTypeFromList

The proc_UnmapContentTypeFromList stored procedure is called to delete an association between a content type and a list. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_UnmapContentTypeFromList(
    @SiteId                   uniqueidentifier,
    @ListId                   uniqueidentifier,
    @ContentTypeId            varbinary(512),
    @RequestGuid              uniqueidentifier = NULL OUTPUT
);
```

@SiteId: The site collection identifier of the site collection in which the content type resides.

@ListId: The list identifier that identifies the list.

@ContentTypeId: The content type identifier of the list content type being unmapped. This MUST be of type tContentTypeId.

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

Return Values: An integer which MUST be 0.

Result Sets: MUST NOT return any result sets.

3.2.4.80 proc_UnmapFieldFromList

The proc_UnmapFieldFromList stored procedure is called to delete the association of a field with a list. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_UnmapFieldFromList(
    @SiteId                   uniqueidentifier,
    @WebId                    uniqueidentifier,
    @ListId                   uniqueidentifier,
    @ContentTypeId            varbinary(512),
    @RequestGuid              uniqueidentifier = NULL OUTPUT
);
```

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

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

@ListId: The list identifier for the list which contains the field.
@ContentTypeId: The field identifier in binary format. This MUST be of type tContentTypeId.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>30</td>
<td>There was an IO error.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.81 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:

```sql
PROCEDURE proc_UnmapFieldsFromContentType(
    @SiteId                   uniqueidentifier,
    @WebId                    uniqueidentifier,
    @ContentTypeId            varbinary(512),
    @RequestGuid              uniqueidentifier = NULL OUTPUT
);
```

@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.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>30</td>
<td>There was an IO error.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.82 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:

```sql
PROCEDURE proc_UpdateContentTypeInScope ( 
    @SiteId           uniqueidentifier,
    @Class            tinyint,
    @Scope            nvarchar(256),
);
```
@ContentTypeId    varbinary(512),
@Definition       ntext,
@Version          int,
@FeatureId        uniqueidentifier = NULL
@SolutionId       uniqueidentifier = NULL
@SetNextChildByteToZero     bit = 0
@RequestGuid      uniqueidentifier = NULL OUTPUT
);

@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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Site column.</td>
</tr>
<tr>
<td>1</td>
<td>Site content type.</td>
</tr>
</tbody>
</table>

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

@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 2.4.6 and [MS-WSSCAML] section 2.3.2.9.

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

@FeatureId: The feature identifier of the feature used to update the site content type or site column. If the site content type or site column was not updated via a feature it MUST be NULL.

@SolutionId: Specifies the solution identifier of the solution used to update the site content type or site column. If the site content type or site column was not updated via a solution, it MUST be NULL.

@SetNextChildByteToZero: Specifies if the NextChildByte column is set to 0 for the site content type. When the value of the parameter is 1, the NextChildByte column MUST be set to 0 for the site content type. When the value of the parameter is 0, the NextChildByte column MUST remain the current value for the site content type.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>2</td>
<td>The requested site content type or site column does not exist.</td>
</tr>
<tr>
<td>212</td>
<td>The site collection is locked.</td>
</tr>
<tr>
<td>1150</td>
<td>The @Version parameter doesn't match the version of the existing record in the back-end database server.</td>
</tr>
<tr>
<td>1816</td>
<td>The site collection quota for the specified site collection has been exceeded.</td>
</tr>
</tbody>
</table>
**Result Sets:** MUST NOT return any result sets.

### 3.2.4.83 proc_UpdateFeatureProperties

The `proc_UpdateFeatureProperties` stored procedure is called to update the Feature Property Definitions (section 2.2.6.3.1) of a feature marked as active. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_UpdateFeatureProperties(
    @SiteId                   uniqueidentifier,
    @WebId                    uniqueidentifier,
    @FeatureId                uniqueidentifier,
    @SolutionLevel            int,
    @Flags                    int = 0,
    @Properties               nvarchar(max) = NULL,
    @RequestGuid              uniqueidentifier = NULL OUTPUT
);
```

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

**@WebId:** MUST be a site identifier containing the NULL GUID if the feature is scoped to a site collection. Otherwise, this parameter is 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.

**@SolutionLevel:** Specifies the installation state of the feature. The value MUST be an integer which is listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active</td>
<td>Solution is in installed active state.</td>
</tr>
<tr>
<td>2</td>
<td>Installing</td>
<td>Installation or upgrade of the solution is in progress.</td>
</tr>
<tr>
<td>3</td>
<td>Previous</td>
<td>Solution is from a previous installation.</td>
</tr>
</tbody>
</table>

**@Flags:** MUST be 0.

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site collection or site (2) does not exist, or the feature is not marked active in the site collection or site (2).</td>
</tr>
</tbody>
</table>

**Result Sets:** MUST NOT return any result sets.
### 3.2.4.84 proc_UpdateFeatureVersion

This stored procedure updates the version of the specified feature. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_UpdateFeatureVersion (  
    @SiteId uniqueidentifier,  
    @WebId uniqueidentifier,  
    @FeatureId uniqueidentifier,  
    @SolutionLevel int,  
    @Version nvarchar(64)  
);  
```

**@SiteId:** The site collection identifier of the site collection in which the feature exists.

**@WebId:** If the feature is site collection feature scoped, this parameter MUST be a NULL GUID. Otherwise, this parameter MUST be set to the site identifier of the site in which the feature exists.

**@FeatureId:** The feature identifier of an existing feature whose version will be updated. This parameter MUST NOT be NULL.

**@SolutionLevel:** Specifies the installation state of the feature. The value MUST be one of the integers listed in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Active</td>
<td>Solution is in installed active state.</td>
</tr>
<tr>
<td>2</td>
<td>Installing</td>
<td>Installation or upgrade of the solution is in progress.</td>
</tr>
<tr>
<td>3</td>
<td>Previous</td>
<td>Solution is from a previous installation.</td>
</tr>
</tbody>
</table>

**@Version:** The new version for the feature.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The version for the specified feature was updated successfully.</td>
</tr>
<tr>
<td>3</td>
<td>The version for the specified feature failed to be updated.</td>
</tr>
</tbody>
</table>

**Result Sets:** MUST NOT return any result sets.

### 3.2.4.85 proc_UpdateListContentTypes

The `proc_UpdateListContentTypes` stored procedure is called to update the list content types available on a given list. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_UpdateListContentTypes(  
    @SiteId uniqueidentifier,  
    @WebId uniqueidentifier,  
    @ListId uniqueidentifier,  
    @ContentTypes tCompressedString,  
    @ContentTypesSize int,  
);  
```
@Version int,
@RequestGuid uniqueidentifier = NULL OUTPUT
);

@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.

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

@ContentTypes: The XML fragment that defines the list content types for the list specified by @ListId. The XML schemas for this structure is defined in [MS-WSSCAML], section Content Type References. The fragment is compressed by the algorithm specified in [RFC1950].

@ContentTypesSize: The size in bytes of the @ContentTypes parameter.

@Version: The version of the list to update.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The requested site or list does not exist.</td>
</tr>
<tr>
<td>29</td>
<td>There was an IO error.</td>
</tr>
<tr>
<td>1150</td>
<td>@Version is not the current version of the list.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.86 proc_UpdateListFields

The proc_UpdateListFields stored procedure is called to add one or more fields to a list, or to update the field definition of one or more fields in a list. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_UpdateListFields(
    @SiteId                      uniqueidentifier,
    @WebId                       uniqueidentifier,
    @ListId                      uniqueidentifier,
    @Fields                      varbinary(max),
    @fieldsSize                  int,
    @ContentTypes                varbinary(max),
    @contentTypesSize            int,
    @Version                     int,
    @UpdateListFieldsFlags       int = 1,
    @FieldIdDeleted              uniqueidentifier = NULL,
    @VersionDelta                int = 1
);
```

@SiteId: The site collection identifier in which the list exists.
@WebId: The site identifier of the site containing the list.

@ListId: The list identifier of the specified list.

@Fields: The version number followed by an XML fragment of the field definitions. The XML schema for this structure is specified in [MS-WSSCAML] section 2.3.2.9.

@fieldsSize: The size in bytes of the @Fields parameter.

@ContentTypes: The updated XML fragment of the content types used by the specified list. The content types XML is not modified if this is NULL. The XML schema for this structure is specified in [MS-WSSCAML] section 2.4.6.

@contentTypesSize: The size in bytes of the @ContentTypes parameter.

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

@UpdateListFieldsFlags: An integer value which MUST consist of one or more bit flags defined in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Indicates whether the field schema, as specified in [MS-WSSCAML], section 2.3.2.9 – FieldDefinitions Type, for this list has been modified.</td>
</tr>
<tr>
<td>2</td>
<td>Indicates that the field schema, as specified in [MS-WSSCAML], section 2.3.2.9 – FieldDefinitions Type, has been updated, and the only change is the build number in the version string.</td>
</tr>
</tbody>
</table>

@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 which MUST be one of the values in the following table:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site collection or site does not exist.</td>
</tr>
<tr>
<td>212</td>
<td>Adding content to the site is prevented or access to the site is blocked.</td>
</tr>
<tr>
<td>1150</td>
<td>A concurrency violation occurred. The version specified by @Version does not exist for the list specified by @ListId.</td>
</tr>
<tr>
<td>1816</td>
<td>The site collection quota for the site collection has been exceeded.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.87 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:

```sql
PROCEDURE proc_UpdateSiteHashKey(
    @SiteId uniqueidentifier,
)"
```
@SiteHashKey              binary(16),
@RequestGuid              uniqueidentifier = NULL OUTPUT
);

@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.

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

Return Code Values: An integer which the protocol client MUST ignore.

Result Sets: MUST NOT return any result sets.

3.2.4.88 proc_UpdateSiteUpgradeSession

The proc_UpdateSiteUpgradeSession stored procedure is called to update progress about a site collection upgrade. The stored procedure is defined using T-SQL syntax, as follows:

PROCEDURE proc_RetrieveSiteUpgradeSession (  
  @SiteId              uniqueidentifier,
  @SessionId           uniqueidentifier,
  @SessionXML          nvarchar(max),
  @UpgradeStatus       int,
  @ErrorCount          int,
  @WarningCount        int,
  @LogFileLocation     nvarchar(260),
  @RequestGuid         uniqueidentifier = NULL OUTPUT
);

@SiteId: The site collection identifier of the site collection whose site collection upgrade progress is being reported.

@SessionId: A unique identifier that is associated with the site collection with proc_AcquireSiteUpgradeSession.

@SessionXML: Contains implementation-specific data about the current status of the site collection upgrade, but otherwise opaque to the back-end database server.

@UpgradeStatus: Contains the current upgrade status of the site collection upgrade.

@ErrorCount: The number of errors for the site collection upgrade.

@WarningCount: The number of warnings for the site collection upgrade.

@LogFileLocation: The site-relative URL of the file that contains the site collection upgrade log. MUST be the empty string if there is no log file.

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

Return Code Values: An integer which MUST be one of the values in the following table:
<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>1</td>
<td>Failed. The site collection is associated with a different unique identifier for site collection upgrade.</td>
</tr>
</tbody>
</table>

**Result Sets:** MUST NOT return any result sets.

### 3.2.4.89 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:

```sql
PROCEDURE proc_UpdateTpWebMetaData(
    @SiteId uniqueidentifier,
    @WebId uniqueidentifier,
    @Title nvarchar(255),
    @Description nvarchar(max),
    @Version int,
    @WebTemplate int,
    @Language int,
    @Locale int,
    @Collation smallint,
    @TimeZone smallint,
    @Collation smallint,
    @CalendarType smallint,
    @AdjustHijriDays smallint,
    @AltCalendarType tinyint,
    @TimeZone smallint,
    @TimeZone smallint,
    @WorkDays smallint,
    @WorkDayStartHour smallint,
    @Config smallint,
    @Flags int,
    @Author int,
    @DefTheme nvarchar(64),
    @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),
    @AllowMUI bit,
    @TitleResource nvarchar(256),
    @DescriptionResource nvarchar(256),
    @AlternateMUCultures nvarchar(max),
    @OverwriteMUCultures bit,
    @TemplateVersion smallint,
    @UIVersion tinyint,
    @ClientTag smallint,
    @TimeCreated datetime = NULL,
    @TimeLastModified datetime = NULL,
    @RequestGuid uniqueidentifier = NULL OUTPUT
)```
@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 of the site definition that contains the site definition configuration used to provision the site. If the value is -1, then the existing value MUST NOT be 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 zero, the existing value MUST NOT be updated.

>Locale: The LCID associated with the site which is used to determine the current culture for locale settings. If the value is zero, the existing value MUST NOT be updated.

>Collation: The collation order of the site which indicates an additional sort 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 MUST be used when displaying time values for this site. If the value is -1, the existing value MUST NOT be updated.

>Time24: If set to 1, a 24-hour time format MUST be used when displaying time values for this site; otherwise, a 12-hour time format MUST be used.

>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. (For more information see [MS-WSSFO2], section 2.2.4.1 – Calendar View Options Type.)

>WorkDays: A set of Workdays Flags which specify the week days defined as the work week for this site. (For more information about Workdays Flags, see [MS-WSSFO2], section 2.2.2.13 – Workdays Flag.)

>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.

@Flags: A site property flags (see [MS-WSSFO2], section 2.2.10) value describing the configuration of this site. If the value is NULL, the existing value MUST NOT be updated.

@Author: The user identifier of the user who is listed as the creator of this site. If the value is NULL, the existing value MUST NOT be updated.

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

@AlternateCSSUrl: The URL for a custom cascading style sheet (CSS) file registered on the site for use in pages of the site.

@CustomizedCss: This contains a list of custom CSS files associated with this site.

@CustomJSUrl: The store-relative URL for a custom JScript file registered on the site for use in pages of the site.

@AlternateHeaderUrl: The store-relative 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 class identifier (CLSID) of the external security provider for this site. This MUST be NULL for sites using the default security implementation.

@MasterUrl: The store-relative URL of the master page for the specified site. If the value is NULL, the existing value is not updated.

@CustomMasterUrl: The store-relative URL for the custom master page for a given site. If the value is NULL, the existing value is not updated.

@SiteLogoUrl: The store-relative 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.

@AllowMUI: A bit which indicates whether the multilingual user interface (MUI) feature is enabled. The MUI feature is enabled when @AllowMUI is 1, otherwise the feature is not enabled.

@TitleResource: The resource token or resource identifier of the title for the site whose metadata is to be updated.

@DescriptionResource: The resource token or resource identifier of the description for the site whose metadata is to be updated.

@AlternateMUI Cultures: The string that contains the distinct language code identifier (LCID) for all the alternate language(s) of the site. Every element MUST be separated by semicolon. The general format is specified as follows.

(<Language Identifier>;<Language Identifier>;...<Language Identifier>)

@OverwriteMUI Cultures: A bit which specifies whether the changes made to user-specified text in the default language should automatically overwrite the existing translations made in all alternate languages. If the value is 1, the translations MUST be overwritten.
@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 MUST NOT be updated.

@UIVersion: A user interface (UI) version number that represents the user interface of the site.

@ClientTag: An integer that represents the application file cache version for files in the site.

@TimeCreated: The time that the site is created. This MUST be in UTC format. If the value is NULL, then the existing value MUST NOT be updated.

@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 MUST be updated to the system time of the back-end database server.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Successful completion.</td>
</tr>
<tr>
<td>3</td>
<td>The site specified by @WebId does not exist</td>
</tr>
<tr>
<td>1150</td>
<td>Failed to update, as the value specified by @Version does not match the version of the site metadata being updated.</td>
</tr>
</tbody>
</table>

Result Sets: MUST NOT return any result sets.

3.2.4.90 proc_GetFollowableLists

The proc_GetFollowableLists stored procedure returns a list of the followable lists in a content database, along with metadata about each list. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_GetFollowableLists (  @LastItemModifiedDate datetime,  @ListId uniqueidentifier,  @RowLimit int,  @RequestGuid uniqueidentifier = NULL OUTPUT );
```

@LastItemModifiedDate: A value to be used to filter which followable lists are returned. Only lists whose last item modified dates are less than or equal to this value MUST be returned.

@ListId: A value to be used to filter which followable lists are returned. For lists whose last item modified date is equal to @LastItemModifiedDate, only lists whose list identifier is greater that @ListId MUST to be returned.

@RowLimit: The number of rows returned MUST not exceed this value.

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

Return Code Values: An integer which MUST be 0.
3.2.4.90.1 Followed List Result Set

This result set returns the list of followable lists specified by the input parameters. The result set MUST return 1 row for each followable list. The result set MUST be sorted in descending order according to the Modified value and within that ascending order according to the ListID value. This result set is defined using T-SQL syntax, as follows:

```sql
SubscriptionId uniqueidentifier,
HostHeader nvarchar(128),
BitFlags int,
SiteId uniqueidentifier,
tp_WebId uniqueidentifier,
ListID uniqueidentifier,
Modified datetime,
LastDeleted datetime,
D.DirName nvarchar(256),
D.LeafName nvarchar(128);
```

**SubscriptionId**: MUST contain the GUID of the implementation-specific subscription feature for the site collection containing the list or NULL if the list or site collection does not exist.

**HostHeader**: For the site collection containing the list, this MUST contain the Host Header string associated with the site collection or NULL if the list or site collection does not exist.

**BitFlags**: MUST contain a site collection flags value describing the site collection containing the list or NULL if the list or site collection does not exist.

**SiteId**: MUST contain the site collection identifier of the site collection containing the list or NULL if the list does not exist.

**tp_WebId**: MUST contain the identifier of the site containing the list or NULL if the list does not exist.

**ListId**: MUST contain the identifier of the list or NULL if the list does not exist.

**Modified**: MUST contain the time in UTC (Coordinated Universal Time) specifying when an item in the list was last modified or NULL if the list does not exist.

**LastDeleted**: MUST contain the time in UTC (Coordinated Universal Time) specifying when an item in the list was last deleted, the creation time of the list if no items from the list have ever been deleted, or NULL if the list does not exist.

**D.DirName**: MUST contain the directory name of the location that contains the list or NULL if the list does not exist.

**D.LeafName**: MUST contain the leaf name of the location that contains the list or NULL if the list does not exist.

3.2.4.91 proc_GetLastItemModifiedDates

The proc_GetLastItemModifiedDates stored procedure takes as a parameter a list of lists and returns metadata about each list. The stored procedure is defined using T-SQL syntax, as follows:

```sql
PROCEDURE proc_GetLastItemModifiedDates (
    @LastItemModifiedTable tvpLastItemModified READONLY,
    @RequestGuid uniqueidentifier = NULL OUTPUT
```

[MS-WSSCCSP3] — v20120630
Windows SharePoint Services: Content Database Core List Schema and Site Provisioning Communications Version 3 Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
@LastItemModifiedTable: A type as defined in 2.2.7.1 specifying the list of
lists for which metadata is being requested.

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

Return Code Values: An integer which MUST be 0.

3.2.4.91.1 List Result Set

This result set returns the metadata about the list of lists specified by the input parameters. The
result set MUST return 1 row for each list about which metadata is requested. This result set is
defined using T-SQL syntax, as follows:

ArrayIndex        int,
WebId              uniqueidentifier,
ListId             uniqueidentifier,
Modified           datetime,
LastDeleted        datetime;

ArrayIndex: An integer value representing the list's index in the array.

WebId: MUST contain the identifier of the site containing the list or NULL if the list does not exist.

ListId: MUST contain the identifier of the list or NULL if the list does not exist.

Modified: MUST contain the time in UTC (Coordinated Universal Time) specifying when an item in
the list was last modified or NULL is the list does not exist.

LastDeleted: MUST contain the time in UTC (Coordinated Universal Time) specifying when an item
in the list was last deleted, the creation time of the list if no items from the list have ever been
deleted, or NULL if the list does not exist.

3.2.5 Timer Events

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

3.2.6 Other Local Events

No other local events impact the operation of this protocol.

3.3 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.

3.3.1 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 finished as part of a response for a higher level event.

- Configuration
- Site collections
- Sites
- Lists
- List items
- Documents
- Users
- Groups

3.3.2 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.

3.3.3 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).

3.3.4 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.

3.3.5 Timer Events

If the connection timeout event is triggered, the connection and the stored procedure call fails.
3.3.6 Other Local Events

No other local events impact the operation of this protocol.
4 Protocol Examples

4.1 Features

This section provides examples that show how to activate and deactivate a feature.

4.1.1 Activate a Feature at a Site

This scenario is initiated when a feature is activated for 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.

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 deactivate the specified feature by calling the proc_DeactivateFeature stored procedure.

2. 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 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',
   @DirParentId = 1,
   @AddMinorVersion = 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-8ea5-26ea7c1d1621}"><Folder TargetName="Item"/></Folder><FieldRefs><FieldRef ID="{c042a256-4a8e-8ea5-26ea7c1d1621}" Name="ContentType"/></FieldRefs><XmlDocuments><XmlDocument NamespaceURI="http://schemas.microsoft.com/sharepoint/v3/richtext" Required="TRUE">ShowInNewForm="TRUE"</XmlDocument></XmlDocuments></ContentType><ContentType ID="0x01200087D020766698947AC0E70B27EDEE926" Name="Folder" Group="Folder"/>

[MS-WSSCCSP3] — v20120630
Windows SharePoint Services: Content Database Core List Schema and Site Provisioning Communications
Version 3 Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
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_MapContentTypeToList` stored procedure with the following parameters to associate the "snowboarder" list content type with the "Ski Team" list:

```sql
EXEC proc_MapContentTypeToList
    @SiteId = N'59e2191d-fca4-4061-8a2c-7ed5b5fe9fedc',
    @WebId = N'ddf4f4f1-8c7-4f7-8-b41-a3d5836900da',
    @ListId = N'446e723a-989-4-763-9bd-2721b8b073d',
    @ContentTypeId = N'0x010030FB45D373D641489EE87FA33528FD40E078487DD373026C438946D85BC4F2788',
    @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:

```sql
EXEC proc_UpdateListFields
    @SiteId = N'59E2191D-FCA4-4061-8A2C-7ED5B5FE9FEDC',
    @WebId = N'DDF4F4F1-8C7-4F7-8-B41-A3D5836900DA',
    @ListId = N'446E723A-989-4763-9B7D-2721B8B073D',
    @Fields = N'12.0.0.6219.0.<FieldRef Name="ContentTypeId"/><FieldRef Name="Title" ColName="nvarchar1"/><FieldRef Name="_ModerationComments" ColName="ntext1"/><FieldRef Name="File_x0020_Type" ColName="nvarchar2"/>asics:sharepoint/v3" DisplayName="Contact Photo" Group="Core Contact and Calendar Columns" Type="URL" Format="Image" Sealed="TRUE" Sealed="TRUE" AllowDeletion="TRUE" Customization="" SourceId="http://schemas.microsoft.com/sharepoint/v3" DISPLAY_NAME="Contact Photo"
Group="Core Contact and Calendar Columns" Type="URL" Format="Image" Sealed="TRUE"
```

126 / 150

[MS-WSSCCSP3] — v20120630
Windows SharePoint Services: Content Database Core List Schema and Site Provisioning Communications Version 3 Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
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 enjoy besides snowboarding. Jonathan’s friend David asked to make the "Hobbies" column a Required field. 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. 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-7EBEFC9FEC0D',
    @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-b863-017e6dd2d47}" Name="Title" Required="TRUE" ShowInNewForm="TRUE" ShowInEditForm="TRUE" />
        <FieldRef ID="{203fa378-6ebd-4d9a-af16-22a4c1f4f461}" Name="Hobbies" Required="TRUE" Customization="" ReadOnly="FALSE" PITarget="MicrosoftWindowsSharePointServices" PIAttribute="ContentTypeID"><Default>Item</Default><CHOICES><CHOICE>Item</CHOICE><CHOICE>Folder</CHOICE><CHOICE>snowboarder</CHOICE></CHOICES></FieldRef></FieldRefs><XmlDocuments><XmlDocument NamespaceURI="http://schemas.microsoft.com/sharepoint/v3/contenttype/forms">PEZvcm1UZW1wbGF0ZXMgeG1sbnM9Imh0dHA6Ly9zY2hlbWFzLm1pY3Jvc29mdC5jb20vc2hhcmVwb2lud3J5d250ZW50
    dHlwZGJyb3Jtcy+TGlzdE5ldz5MaXU8m9yb29vbTmV3PjwvQXVzaXZl玩耍" /></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-7ed5bf9fecd',
    @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-7ed5fc9fedc',
   @WebId = 'dd1f4f41-8cf7-4f78-b41d-a3d5836900da',
   @ContentTypeId = 0x010030FB45D373D641489EE87FA33528FD4E,
   @FieldId = 'c04a2a56-787d-4a6f-8a8a-cf6ab767f12d';

EXEC `proc_MapFieldToContentType`
@SiteId = '59e2191d-fca4-4061-8a2c-7ed5fc9fedc',
@WebId = 'dd1f4f41-8cf7-4f78-b41d-a3d5836900da',
@ContentTypeId = 0x010030FB45D373D641489EE87FA33528FD4E,
@FieldId = 'fa564e0f-0c70-4ab9-b863-0177e6dd247';

EXEC `proc_MapFieldToContentType`
@SiteId = '59e2191d-fca4-4061-8a2c-7ed5fc9fedc',
@WebId = 'dd1f4f41-8cf7-4f78-b41d-a3d5836900da',
@ContentTypeId = 0x010030FB45D373D641489EE87FA33528FD4E,
@FieldId = '203fa378-6eb8-4ed9-a4f9-221a4c1fbf46';

EXEC `proc_MapFieldToContentType`
@SiteId = '59e2191d-fca4-4061-8a2c-7ed5fc9fedc',
@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 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.

<table>
<thead>
<tr>
<th>ContentTypeId</th>
<th>WebId</th>
<th>ListId</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x010030FB45D373D641489EE87FA33528FD4E0078487DD373026C438B946D85BC4F2788</td>
<td>DD1F4F41-8CF7-4F78-B41D-A3D5836900DA</td>
<td>44E6723A-9894-4763-9B7D-27210B80B73D</td>
</tr>
</tbody>
</table>

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',

---

[MS-WSSCCSP3] — v20120630
Windows SharePoint Services: Content Database Core List Schema and Site Provisioning Communications Version 3 Protocol Specification

Copyright © 2012 Microsoft Corporation.

Release: July 16, 2012
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:

```sql
EXEC proc_UpdateListFields
@SiteId = '59E2191D-FCA4-4061-8A2C-7EDB5FC9FECD',
@WebId = 'DD1F4F41-8CF7-4F78-B41D-A3D5836900DA',
@ListId = '44E6723A-9894-9BD7-2721B08B73D',
@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-9894-9BD7-2721B08B73D}" 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"',
@Version = 71;
```
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.

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_ListContentTypesInWebRecursive stored procedure.

2. The back-end database server returns result sets as listed in proc_ListContentTypesInWebRecursive.

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.
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. 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.
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 3.1.28.1 - [MS-WSSFO2], section 3.1.4.28.5.

8. The front-end Web server builds a transactional dynamic query in T-SQL syntax to update the definition (the display name in this scenario) of the field in the list.

   • The query begins a new transaction.
   • The query attempts to update the field definition of Fields 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.

   • The query begins a new transaction.
   • The query attempts to update the list content types on the specified list using the proc_UpdateListContentTypes stored procedure.
   • 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 3.1.5.19.1 - [MS-WSSFO2], section 3.1.5.19.3.

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 3.1.5.20.6.

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.

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 3.1.5.20.6.

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 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. In addition, another view may be created to display only those list items whose creator field 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.

4.4 Custom Actions

Bryan is a web manager for a company of door to door sales agents. On their Web site, this company has a list of potential customers with their addresses. Bryan wants his agents to be able to open up the mapped location for any customer from a list on their site. This can be achieved by associating a custom action to the site which will be visible to the sales agents depending on their role and permission. Because custom actions can either start URLs or JScript calls when triggered, this action when triggered can act upon a selected customer in the list and open their location in a mapping Web site using JScript.

For simplicity’s sake, this example assumes that:

- There’s JScript function already written that can take an item identifier and fetch its address column values. The name of the JScript function is MyJumpToMapFunction. Using that, this JScript can begin the mapping Web site with the retrieved address.

4.4.1 Add a Custom Action

To add a custom action for Bryan’s scenario, the proc_AddOrUpdateCustomAction stored procedure is called and the following actions occur in the specified sequence:
1. The protocol client calls the `proc_AddOrUpdateCustomAction` stored procedure with the following parameters:

   ```sql
   exec proc_AddOrUpdateCustomAction @Id='80788823-0DB5-428E-8FA5-0418DF7DC2CE',@ScopeId='CB17ECAC-BFA4-43FC-ACAA-AF616FEDDD43',@SiteId='8ECFF020-3AA4-4D60-A27A-A566D1537C36',@WebId='CB17ECAC-BFA4-43FC-ACAA-AF616FEDDD43',@FeatureId='00000000-0000-0000-0000-000000000000',
   @SolutionId=NULL,@SolutionWebId=NULL,@SolutionLevel=1,@ScopeType=3,@Properties=N'<?xml version="1.0" encoding="utf-16"?><Elements xmlns="http://schemas.microsoft.com/sharepoint/" xmlns:ac="http://schemas.microsoft.com/sharepoint/">
   ```

2. The protocol server returns a return code of 0, which indicates successful execution. The protocol server also returns a result set defined in 3.2.4.4.1 "Add or Update Custom Action Result Set", as follows:

   ```
<table>
<thead>
<tr>
<th>UnnamedColumn</th>
</tr>
</thead>
<tbody>
<tr>
<td>No row returned</td>
</tr>
</tbody>
</table>
   ```

   - The protocol client ignores the result set.

4.4.2 Retrieve a Custom Action

To display the custom action for Bryan’s scenario, the user interface would need to call the `proc_GetCustomActionsFromScope` stored procedure, resulting in the following actions to occur in the specified sequence:

1. The protocol client calls the protocol server with the following parameters for this stored procedure:

   ```sql
   exec proc_GetCustomActionsFromScope @ScopeId='CB17ECAC-BFA4-43FC-ACAA-AF616FEDDD43',@SiteId='8ECFF020-3AA4-4D60-A27A-A566D1537C36',@WebId='CB17ECAC-BFA4-43FC-ACAA-AF616FEDDD43',@SolutionLevel=1
   ```

2. The protocol server returns a return code of 0 which indicates successful execution. The protocol server also returns a result set, defined in 3.2.4.32.1 "Get Custom Actions From Scope Result Set", as follows:
The protocol client then uses this result set to display the custom actions on the user interface.

### 4.4.3 Delete a Custom Action

In the case where Bryan no longer desires the custom action in his scenario, the proc_DeleteCustomAction stored procedure can be called, resulting in the following actions to occur in the specified sequence:

1. The protocol client calls the proc_DeleteCustomAction stored procedure with the following parameters:

   ```
   exec proc_DeleteCustomAction @Id='223154a8-2671-4778-8a75-b189acaf96c0',@ScopeId='CB17ECA-C-BFA4-43FC-ACAAF616FE-DDD43',@SiteId='8ECFF020-3AA4-4D60-A27A-A566D15373C6',@WebId='CB17ECA-C-BFA4-43FC-ACAAF616FE-DDD43',@ScopeType=3
   ```

2. The protocol server returns a return code of 0, which indicates successful execution. The protocol server also returns 2 result sets as defined in 3.2.4.33.1 "Delete Custom Action Result Set One" and 3.2.4.33.2 "Delete Custom Action Result Set Two", as follows:

   **Delete Custom Action Result Set One:**

   ```
   {UnnamedColumn}
   ```

   | 1 |
Delete Custom Action Result Set Two:

<table>
<thead>
<tr>
<th>ScopeCount</th>
<th>ParentScopeCount</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- The protocol client ignores the result sets.

4.5 Metadata Information

This section provides an example of working with metadata.

4.5.1 proc_SetWebMetaInfo

- The metadata information for a given site is set using this stored procedure. An example of this is while creating a site.

Jonathan wants to create a site to store all his contact information. He chooses the create site option, with the URL specified as http://<servername>:<port>/Contacts. While the site is created, in order to set the metadata information for this site, this stored procedure is called.

```sql
proc_SetWebMetaInfo

    @ WebSiteId = "92F730F6-0702-4AF6-BE22-3770B17A7630",
    @ WebUrl = N"Contacts",
    @ MetaInfo = @wssp1,
    @Flags = @wssp2,
    @DefTheme = NULL,
    @IncrementSiteTimeStamp = 1,
    @MasterURL = NULL,
    @CustomMasterURL = NULL,
    @@WebId output;
```

Where @wssp1 contains the binary encoded value for the metainformation of the site (with value in this case as "vti_categories:VR\Travel Expense\\ Report Business Competition Goals/Objectives Ideas Miscellaneous Waiting VIP In\\ Process Planning Schedule  vti_defaultlanguage:SW|en-us vti_extenderversion:SR|14.0.0.0  vti_approvallevels:VR\Approved Rejected Pending\\ Review ")

And @wssp2 = 291. This is the integer representation of the following flags set –
<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00000001</td>
<td>This site allows display of implementation-specific user presence information in the front-end Web server.</td>
</tr>
<tr>
<td>0x00000002</td>
<td>This site allows display of implementation-specific enhanced user presence information in the front-end Web server.</td>
</tr>
<tr>
<td>0x00000020</td>
<td>The front-end Web server for this site displays the local navigation element.</td>
</tr>
<tr>
<td>0x00000100</td>
<td>This site has not yet been provisioned with a site template.</td>
</tr>
</tbody>
</table>

When site creation succeeds, the stored procedure returns a value of 0.
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 permissions, 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® SharePoint® Foundation 2013 Preview
- Microsoft® SQL Server® 2008 R2 SP1
- Microsoft® SQL Server® 2012

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

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

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

A
Abstract data model
  client 121
  server 27
Add or update custom action result set 38
All file URLs result set 77
AllListsAux table structure 24
Applicability 16
Attribute groups - overview 26
Attributes - overview 26
Audit mask result set 92

B
Back-end database server interface 27
Binary structures
  List Base Type Pattern 19
  List Identifier Packed Array 18
  ContentTypeId 17
  Usage Data Binary Field Structure 19
Bit fields - overview 17

C
Capability negotiation 16
Change tracking 144
ChildWebs result set 79
Client
  abstract data model 121
  front-end Web server interface 121
  initialization 122
  local events 123
  message processing 122
  overview 121
  sequencing rules 122
  timer events 122
  timers 122
Client-interface
  front-end Web server 121
Common data types
  overview 17
Complex types
  Feature Property Definitions 24
  Site Health Check Result Message 25
Content type descendants result set (section 3.2.4.46.1 82, section 3.2.4.62.1 97)
Content type is ghosted result set 75
Content type list usage result set 83
Content type result set 53
Content types and columns example 125
Content types result set 59
Creation and modification result set 66
Custom actions example 137

D
Daily usage result set 74
Data model - abstract
  client 121
  server 27
Data types
  common 17
  Upgrade Status simple type 17
  Upgrade Type simple type 17
Data types - simple
  Upgrade Status 17
  Upgrade Type 17
Defined content type order result set 58
Delete custom action result set one 71
Delete custom action result set two 71
Derived content types result set 85
Derived site content types result set 85
Domain group cache back-end database server update 94
Domain group cache back-end database server update result set 63
Domain group cache Front-End Web Server update result set (section 3.2.4.27.3 63, section 3.2.4.61.4 94)
Domain group cache versions result set (section 3.2.4.27.1 63, section 3.2.4.61.2 94)

E
Elements - overview 26
EstimatedSize result set 52
Event receivers result set (section 3.2.4.29.4 66, section 3.2.4.61.6 95)
Events
  local - client 123
  local - server 121
  timer - client 122
  timer - server 121
Examples
  content types and columns 125
  custom actions 137
  features 124
  metadata information 140
  views 137

F
Feature properties result set 56
Feature Property Definitions - complex type 24
Features example 124
Field definition result set 64
Fields - vendor-extensible 16
Flag structures - overview 17
Front-end Web server client interface 121

G
Get custom actions from scope result set 69
Get Web feature list result set one 68
Glossary 9
Groups - overview 26
I
Implementer - security considerations 142
Index of security parameters 142
Informative references 13
Initialization
client 122
server 32
Introduction 9
Invalid parameters result set 57

L
List Base Type Pattern binary structure 19
List child webs filtered result set 80
List Content Types result set 22
List count result set 48
List event receivers result set 49
List Identifier Packed Array binary structure 18
List metadata receivers result set 65
List metadata result set 48
List permissions result set 49
ListContentTypesInWeb result set 84
Lists result set 59
Lists using field result set 86
Local events
client 123
server 121

M
Map URL to list and review result set 90
Message processing
client 122
server 32
Messages
add or update custom action result set 38
all file URLs result set 77
AllListsAux table structure 24
attribute groups 26
attributes 26
audit mask result set 92
bit fields 17
ChildWebs result set 79
common data types 17
content type descendants result set (section 3.2.4.46.1 82, section 3.2.4.62.1 97)
content type is ghosted result set 75
content type list usage result set 83
content type result set 83
content types result set 59
creation and modification result set 66
daily usage result set 74
defined content type order result set 58
delete custom action result set one 71
delete custom action result set two 71
derived content types result set 85
derived site content types result set 85
domain group cache back-end database server update result set (section 3.2.4.27.2 63, section 3.2.4.61.3 94)
domain group cache Front-End Web Server update result set (section 3.2.4.27.3 63, section 3.2.4.61.4 94)
domain group cache versions result set (section 3.2.4.27.1 63, section 3.2.4.61.2 94)
elements 26
EstimatedSize result set 52
event receivers result set (section 3.2.4.29.4 66, section 3.2.4.61.6 95)
feature properties result set 56
Feature Property Definitions complex type 24
field definition result set 64
flag structures 17
get custom actions from scope result set 69
get Web feature list result set one 68
groups 26
invalid parameters result set 57
List Base Type Pattern binary structure 19
list child webs filtered result set 80
List Content Types result set 22
list count result set 48
list event receivers result set 49
List Identifier Packed Array binary structure 18
list metadata result set (section 3.2.4.13.2 48, section 3.2.4.29.1 65)
list permissions result set 49
ListContentTypesInWeb result set 84
lists result set 59
lists using field result set 86
map URL to list and view result set 90
monthly usage result set 73
namespaces 24
NULL unique permissions result set 66
object content type identifier result set (section 3.2.4.17 53, section 3.2.4.19.1 55)
parent site URL result set 60
related fields result set 66
simple types 24
Site Collection Upgrade Info result set 23
site event receivers result set 63
Site Health Check Result Message complex type 25
site metadata result set (section 3.2.4.27.4 63, section 3.2.4.61.5 94)
site props result set 60
site URL result set 94
SiteWebs result set (section 3.2.4.43.1 78, section 3.2.4.90.1 120, section 3.2.4.91.1 121)
tContentTypeId binary structure 17
transport 17
undefined content type order result set 58
unghosted list fields result set 87
unique permissions result set 65
Usage Data Binary Field Structure binary structure 19
user document security context result set 95
WebId result set (section 3.2.4.35.1 72, section 3.2.4.66.1 99, section 3.2.4.67.1 100)
WebsAndSubwebsDTM result set 50
Metadata information example 140

Methods
proc_AcquireSiteUpgradeSession 33
proc_ActivateFeature 33
proc_AddContentTypeToScope 35
proc_AddOrUpdateCustomAction 36
proc_AddToSiteUpgradeQueue 38
proc_CloneFeature 39
proc_DeactivateContentTypeInScope 40
proc_DeactivateFeature 42
proc_DeleteContentTypeInScope 42
proc_DeleteCustomAction 49
proc_DeleteCustomActionForFeature 51
proc_DeleteFieldTemplateInScope 44
proc_DropListField 45
proc_EnumListsWithMetadata 46
proc_EnumWebAndSubwebsDTM 50
proc_EstimateDocsSize 51
proc_FetchContentTypeInScope 52
proc_FetchNextSiteForUpgrade 53
proc_FixV2ContentTypeField 54
proc_GetContentTypeIdFromUrl 54
proc_GetCustomActionsFromScope 68
proc_GetFeatureProperties 55
proc_GetFolderContentTypeOrder 57
proc_GetFollowableLists 119
proc_GetLastItemModifiedDates 120
proc_GetListContentTypeOrder 57
proc_GetListContentTypeOrder 57
proc_GetListIdsToSync 59
proc_GetParentWebUrl 59
proc_GetSiteProps 60
proc_GetSiteUpgradeSession 62
proc_GetTpWebMetaData 62
proc_GetUnghostedBaseFieldTemplateInSite 64
proc_GetUniqueListMetaTdata 64
proc_GetWebExtendedMetaData 66
proc_GetWebFeatureList 67
proc_GetWebIdOfListId 72
proc_GetWebUsageData 72
proc_HeartbeatSiteUpgradeSession 74
proc_IsContentTypeGhosted 74
proc_IsContentTypeInUseInList 76
proc_IsFieldTemplateUsedInContentTypeTemplate 76
proc_ListAllFileUrls 77
proc_ListAllWebsOfSite 78
proc_ListChildWebs 79
proc_ListChildWebFiltered 80
proc_ListContentTypeInWebRecursive 84
proc_ListContentTypesInWebRecursive 84
proc_ListContentTypeInWeb 83
proc_ListDerivedContentTypes 85
proc_MakeViewDefaultForContentType 87
proc_MakeViewDefaultForList 88
proc_MakeViewMobileDefaultForList 88
proc_MakeContentTypeToList 88
proc_MapContentTypeToList 88
proc_MapContentTypeToList 88
proc_MapContentTypeToListAndView 90
proc_MapT2FieldToList 91
proc_MarkWebAsProvisioned 91
proc_MergeWeb 92
proc_MiniSproc 93
proc_ProvisionContentType 95
proc_ReleaseSiteUpgradeSession 97
proc_RenameListItemsContentType 98
proc.ResolveWebLinkitem 99
proc_RetrieveSiteHealthCheckResults 99
proc_RetrieveSiteUpgradeSession 100
proc_SetListFormToUrl 101
proc_SetSiteFlags 102
proc_SetSiteProps 103
proc_SetSiteProps 103
proc_SetTvView 104
proc_SetWebMetaTdata 105
proc_SetWebUsageData 106
proc_StoreUserTinfoListTdata 107
proc_UnmapContentTypeFromList 108
proc_UnmapFieldFromList 108
proc_UnmapFieldsFromContentType 109
proc_UpdateContentTypeInScope 109
proc_UpdateFeatureProperties 111
proc_UpdateFeatureVersion 112
proc_UpdateListContentTypeS 112
proc_UpdateListFields 113
proc_UpdateSiteHashKey 114
proc_UpdateSiteUpgradeSession 115
proc_UpdateTpWebMetaTdata 116
Monthly usage result set 73

N
Namespaces 24
Normative references 12
NULL unique permissions result set 66

O
Object content type identifier result set (section 3.2.4.17 53, section 3.2.4.19.1 55)
Overview (synopsis) 13

P
Parameters - security index 142
Parent site URL result set 60
Preconditions 16
Prerequisites 16
proc_AcquireSiteUpgradeSession method 33
proc_ActivateFeature method 33
proc_AddContentTypeToScope method 35
proc_AddOrUpdateCustomAction method 36
proc_AddToSiteUpgradeQueue method 38
proc_CloneFeature method 39
proc_CopyResourceDir method 39
proc_DeactivateContentTypeInScope method 40
proc_DeactivateFeature method 42
proc_DeleteContentTypeInScope method 42
proc_DeleteCustomAction method 69
get Web feature list result one  68
invalid parameters  57
list child webs filtered result set  80
List Content Types  22
list count  48
list event receivers  49
list metadata (section 3.2.4.13.2  48, section 3.2.4.19.1  65)
list permissions  49
ListContentTypesInWeb  84
lists  59
lists using field  86
map URL to list and view  90
NULL unique permissions  66
object content type identifier (section 3.2.4.17  53, section 3.2.4.19.1  55)
parent site URL  60
related fields  66
Site Collection Upgrade Info  23
site event  63
site metadata  63
site metadata result set  94
site props  60
site URL  94
SiteWebs (section 3.2.4.43.1  78, section 3.2.4.90.1  120, section 3.2.4.91.1  121)
undefined content type order  58
unghosted list fields  87
unique permissions  65
user document security context  95
WebId (section 3.2.4.35.1  72, section 3.2.4.66.1  99, section 3.2.4.67.1  100)
WebsAndSubwebsDTM  50
Result sets - monthly usage  73

S

Security
implementer considerations  142
parameter index  142
Sequencing rules
client  122
server  32
Server
abstract data model  27
back-end database interface  27
initialization  32
local events  121
message processing  32
overview  27
proc AcquireSiteUpgradeSession method  33
proc ActivateFeature method  33
proc AddContentTypeToScope method  35
proc AddOrUpdateCustomAction method  36
proc AddToSiteUpgradeQueue method  38
proc CloneFeature method  39
proc CopyResourceDir method  39
proc DeactivateContentTypeInScope method  40
proc DeactivateFeature method  42
proc DeleteContentTypeInScope method  42
proc DeleteCustomAction method  69
proc DeleteCustomActionForFeature method  71
proc DeleteFieldTemplateInScope method  44
proc DropListField method  45
proc EnumListsWithMetadata method  46
proc EnumWebAndSubwebsDTM method  50
proc EstimateDocsSize method  51
proc FetchContentTypeInScope method  52
proc FetchNextSiteForUpgrade method  53
proc FixV2ContentTypeField method  54
proc GetContentTypeIdFromUrl method  54
proc GetCustomActionsFromScope method  68
proc GetFeatureProperties method  55
proc GetFolderContentTypeOrder method  57
proc GetFollowableLists method  119
proc GetLastItemModifiedDates method  120
proc GetListContentTypes method  58
proc GetListIdsToSync method  59
proc GetParentWebUrl method  59
proc GetSiteProps method  60
proc GetSiteUpgradeSession method  62
proc GetToWebMetaData method  62
proc GetUnhostedBaseFieldTemplateInSite method  64
proc GetUniqueListMetaData method  64
proc GetWebExtendedMetaData method  66
proc GetWebFeatureList method  67
proc GetWebIdOfListId method  72
proc GetWebUsageData method  72
proc HeartbeatSiteUpgradeSession method  74
proc IncrementSiteClientTag method  75
proc IsContentTypeGhosted method  74
proc IsContentTypeInUseInList method  76
proc IsFieldTemplateUsedInContentTypeTemplate method  76
proc IsFieldTemplateUsedInContentTemplate method  77
proc IsListAllFileUrls method  77
proc IsListAllWebsOfSite method  78
proc IsListChildWebs method  79
proc IsListChildWebsFiltered method  80
proc IsContentTypeInUse method  82
proc IsContentTypeInWeb method  83
proc IsContentTypeInWebRecursive method  84
proc IsListDerivedContentTypes method  85
proc ListsUsingFieldTemplate method  86
proc ListUnhostedFieldTemplatesInList method  86
proc MakeViewDefaultForContentType method  87
proc MakeViewDefaultForList method  88
proc MakeViewMobileDefaultForList method  88
proc MapContentTypeToList method  89
proc MapFieldToList method  90
proc MapV2FieldToList method  91
proc mapWebAsProvisioned method  91
proc MergeWeb method  92
proc MiniSproc method  93
proc ProvisionContentType method  95
proc ReleaseSiteUpgradeSession method  97
proc RemoveFromSiteUpgradeQueue method  98
proc RenameListItemContentType method  98
proc ResolveWikiLinkItem method  99
proc RetrieveSiteHealthCheckResults method  99

149 / 150